SpaceJAM: a Lightweight and Regularization-free Method for Fast Joint Alignment of Images

Nir Barel* 🟢, Ron Shapira Weber* 🟢, Nir Mualem🟢, Shahaf E. Finder🟢, and Oren Freifeld🟢

The Department of Computer Science, Ben-Gurion University of the Negev, Israel
{banir,ronsha,nirmu,finders}@post.bgu.ac.il, orenfr@cs.bgu.ac.il

Abstract. The unsupervised task of Joint Alignment (JA) of images is beset by challenges such as high complexity, geometric distortions, and convergence to poor local or even global optima. Although Vision Transformers (ViT) have recently provided valuable features for JA, they fall short of fully addressing these issues. Consequently, researchers frequently depend on expensive models and numerous regularization terms, resulting in long training times and challenging hyperparameter tuning. We introduce the Spatial Joint Alignment Model (SpaceJAM), a novel approach that addresses the JA task with efficiency and simplicity. SpaceJAM leverages a compact architecture with only ∼16K trainable parameters and uniquely operates without the need for regularization or atlas maintenance. Evaluations on SPair-71K and CUB datasets demonstrate that SpaceJAM matches the alignment capabilities of existing methods while significantly reducing computational demands and achieving at least a 10x speedup. SpaceJAM sets a new standard for rapid and effective image alignment, making the process more accessible and efficient. Our code is available at: https://bgu-cs-vil.github.io/SpaceJAM/.

Keywords: Joint Alignment · Congealing · Regularization-free · STN

1 Introduction

Joint alignment (JA) of images is the task of geometrically transforming an image collection in a way that optimizes some criterion of mutual similarity. JA is useful for reducing uninformative intra-dateset (or intra-class) variability, thereby facilitating more accurate analysis and interpretation across various applications, ranging from medical imaging to automated recognition systems. Unfortunately, achieving a successful JA is fraught with challenges, including poor local optima or even poor *global* optima (*i.e.*, trivial non-useful solutions; *e.g.*, shrinking all images to zero size), intra-class variations in pose/orientation/illumination, complex geometric deformations, visual clutter, and more. A JA method typically consists
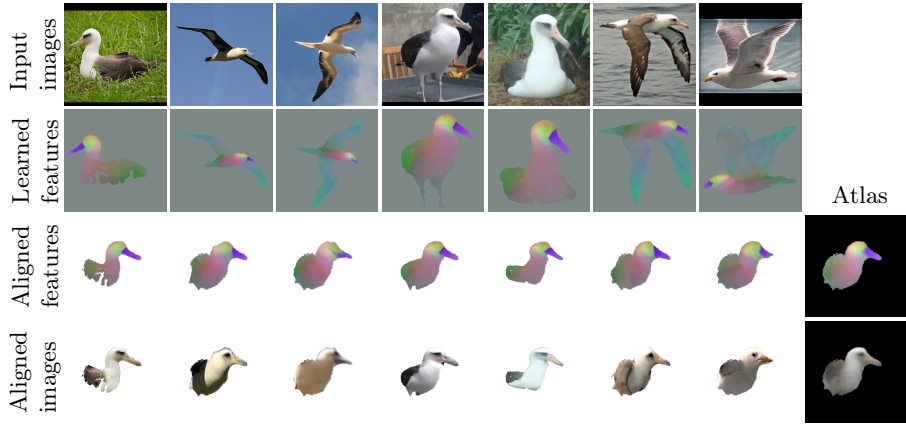
---

*Contributed equally.

Fig. 1: **SpaceJAM joint alignment** Our framework jointly aligns a set of images of an object category in only a few minutes. Top-to-bottom: 1) input images; 2) learned low-dimensional representations; 3) aligned features; 4) aligned images. The last column depicts the average representation (atlas) obtained after training.

of four components: 1) features to be aligned; 2) a criterion of joint similarity; 3) an optimization process; 4) warping the images via either a parametric transformation family (*e.g.*, homographies) or dense pixel-to-pixel correspondences.

Traditionally, JA has relied on classical methods such as congealing [9, 10, 32, 45], which gradually aligns one image towards the rest, or alignment to a centroid which uses a reference image or a (possibly-latent) template. Other methods have utilized feature-based approaches, such as SIFT [41], to establish correspondences between images or key-points. While effective in certain domains (*e.g.*, medical scans), those classical approaches often struggle with more diverse image collections.

With the integration of Deep Learning (DL) into JA (*e.g.*, Chelly *et al.* [8]), especially via the use of Vision Transformers (ViT) [12] features, the field has witnessed a leap forward. ViTs, DINO specifically [7], offer rich semantic features that partially mitigate some of the challenges that hindered traditional JA methods. However, many issues persist even when using ViT features. This is arguably what led recent works [20,49] to over-rely on high-capacity-but-expensive models, as well as extensive regularization strategies. The latter, besides adding to the computational burden, require regularization hyperparameter tuning. Together, that approach led to methods that are slow and often too brittle.

We take a different approach. While we are happy to use DINO features, *we argue that feeding such features to high-capacity models is not, in fact, needed for the JA task and that rather than using such models here, it is better to explore what the best ways are to formulate, and then solve, this task.* With this in mind, this paper shows how to effectively, and efficiently, solve the JA task. To that end, we introduce the Spatial Joint Alignment Model (SpaceJAM), a regularization-free JA approach that does not require explicitly maintaining

an atlas. See also Figure 1. SpaceJAM utilizes a lightweight recurrent Spatial Transformer Network (STN) [26], a Lie-algebraic framework, and a novel loss function. The proposed approach not only ensures fast JA (*e.g.*, at least a 10x speedup in comparison to contemporary methods [20, 49]) but also maintains the geometric integrity of the aligned images. We evaluate our method on the SPair-71K and CUB datasets, and show performance either better than, or on par with, state-of-the-art models but with significantly reduced computational demands, marking a significant advancement in the field of JA.

To summarize, our key contributions are as follows: 1) we introduce SpaceJAM, a lightweight method for JA of in-the-wild images which is much faster than contemporary approaches; 2) a novel inverse-compositional loss function that obviates the need of using regularization terms and does not require maintaining an atlas; 3) an efficient solution for handling reflections (also known as flips).

2   Related Work

Classical JA methods focused on techniques that leverage geometric transformations and hand-crafted features. Congealing, a seminal concept introduced in [32, 45], refers to gradually aligning an image set by iteratively aligning one image towards the others, to minimize a global cost function, typically the entropy or least-squares of the raw pixel values [9, 10, 32] or SIFT descriptors [24, 38, 61]. Other approaches use clustering to simultaneously partition the data and align all class members to their respective mean [39,44] or via template matching [14,18,27]. The success of those classical JA methods, however, was hindered by the quality of the to-be-aligned features. Another classical approach seeks to model an image set as a low-rank linear subspace [21, 30, 69], the most noticeable work being RASL [53], whose main problem lies in its reliance on a good initial alignment.

Deep learning.   The advent of deep learning has significantly expanded the capabilities of image alignment techniques. Huang *et al*. applied the classical congealing algorithm to the features of a Convolutional Neural Net (CNN) [23]. The Spatial Transformer Net (STN) [26] introduced a differentiable module that can be integrated into larger neural nets to predict and apply spatial transformations to input images or feature maps, enabling end-to-end learning of the alignment process. Since their inception, STNs have proven an invaluable tool for the JA task and were used in both the congealing framework [3], explicit Atlas building [11, 35, 62, 66], joint clustering and alignment tasks [40, 47], and for a JA step before building moving-camera background models [8, 13]. Coupled with Generative Adversarial Networks (GAN) [19], STNs were also shown to produce high-fidelity class-category canonical spaces (or atlases) for natural images [48,52]. However, GAN-based methods requires large amount of training data.

Semantic correspondence through self-supervision.   Self-supervised learning (SSL) approaches have recently gained traction for the task of correspondence

Table 1: A comparison with recent JA methods (SPair's 'Cat' category [46]).

Method	# Params	# Losses	#HP	Atlas-free learning	#epochs	Time	PCK@0.10
NeuCongealing [49]	28.7M	8	8	✗	8K	1:17:02	53.3
ASIC [20]	7.9M	4	5	✗	20K	1:06:48	54.8
SpaceJAM (Ours)	**0.016M**	**1**	**0**	✓	**0.7K**	**0:05:58**	**60.8**

discovery between images. By leveraging the representations learned by ViTs (such as DINO [7]) or text-to-image diffusion models [58], several works were able to use these features for the task of **pairwise** image correspondence and alignment [28, 29, 43, 64, 68]. While useful in the pairwise case, finding *joint dense correspondence* between N images is intractable for a large N, as the complexity is $O(N^2)$. Additionally, the Nearest Neighbor (NN) algorithm requires a comparison between every pixel in the source image and all of the pixels in all of the other images. Finally, methods such as [64, 68] require multiple t steps for the diffusion model, which further contributes to the computation burden. In contrast, our proposed JA method produces such a mapping without performing NN searches.

JA using DINO features. DINO features offer robust and semantically meaningful representations for many computer-vision tasks, JA included. The Neural Congealing algorithm [49] utilizes a test-time training framework to build an atlas for a given class (*e.g.*, birds) by matching DINO features via rigid and non-rigid warps, using a ResNet-based STN for each [22]. We remark that despite its name, Neural Congealing [49] is more related to atlas-based methods than to congealing methods. Such ambivalent terminology is prevalent, unfortunately, as sometimes people mistakenly refer to the JA *task* itself as congealing, regardless whether the JA *method* is congealing-based or atlas-based.

ASIC [20] uses DINO features to perform dense warping to map every pixel in the input to a canonical space using a U-net architecture [57]. However, both [49] and [20] grapple with computational overhead, stability issues, and the necessity for extensive regularization to prevent model collapse or trivial solutions. Additionally, ASIC's dense warping approach is prone to incoherent global alignment, often resulting in fragmented or an inconsistent alignment. In contrast, SpaceJAM reaches competitive results much faster and with orders of magnitude fewer trainable parameters. See, *e.g.*, Table 1, as well as the supplementary material (**SupMat**) for additional running times.

In summary, while classical methods have provided a strong foundation for image alignment, DL has significantly enhanced the ability to handle complex and varied alignment tasks. The move towards SSL and semantic correspondence methods further illustrates the evolving landscape of the image alignment field, highlighting ongoing challenges and the need for innovative solutions to address computational efficiency, coherence, and optimization stability. These needs motivated our proposed framework, whose overview is depicted in Figure 2.
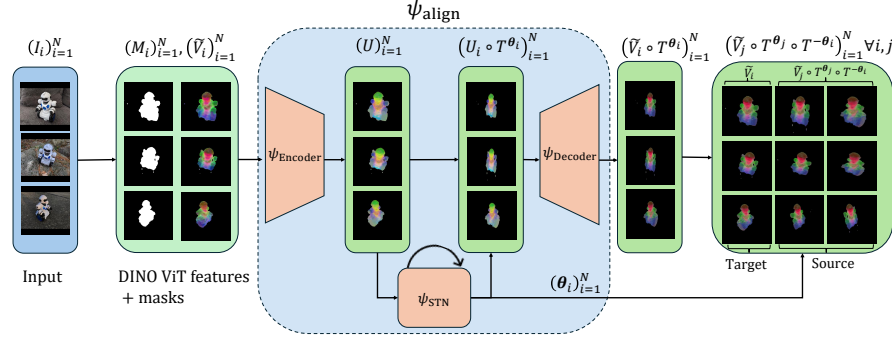
Fig. 2: Framework overview. Given a set of images, $(I_i)_{i=1}^N$, their DINO-ViT representations and coarse masks , $(V_i, M_i)_{i=1}^N$, SpaceJAM learns an inverse-compositional pairwise alignment between each image pair, and consequently, features, $(U_i)_{i=1}^N$, in a shared semantic space, where they are warped (according to learned warping parameters, $(\boldsymbol{\theta}_i)_{i=1}^N$) to produce their aligned versions, $(U_i \circ T^{\boldsymbol{\theta}_i})_{i=1}^N$. Pairwise alignment of I_j to a target image I_i, is achieved by warping to the shared space and then wapring the result by the inverse transformation of the target image, yielding $I_j \circ T^{\boldsymbol{\theta}_j} \circ T^{-\boldsymbol{\theta}_i}$.

3 Background: Typical Challenges in Joint Alignment

Let $\mathcal{T}$ be a family of spatial transformations, from $\mathbb{R}^2$ to $\mathbb{R}^2$. For concreteness, our notation below assumes $\mathcal{T}$ is parametric (as is the case, *e.g.*, for affine transformations or homographies) but most of the discussion holds for the nonparametric case as well. Let $T^{\boldsymbol{\theta}} \in \mathcal{T}$ denote the generic element of $\mathcal{T}$ where $\boldsymbol{\theta}$ is the corresponding parameter vector. Let (I_i, I_j) be an image pair. The **pairwise alignment** problem is

$$\underset{T^{\boldsymbol{\theta}_{ji}} \in \mathcal{T}}{\arg\min} D(I_i, I_j \circ T^{\boldsymbol{\theta}_{ji}}) \tag{1}$$

where $\circ$ is function composition, $T^{\boldsymbol{\theta}_{ji}}$ is the transformation warping I_j towards I_i, and D is a discrepancy measure, based on raw pixels or image features.

Joint alignment. Given N images, $(I_i)_{i=1}^N$, the underlying notion behind the JA problem (see, *e.g.*, [17]) is that for each observed image I_i there is a latent transformation, $T^{\boldsymbol{\theta}_i} \in \mathcal{T}$, such that $(I_i \circ T^{\boldsymbol{\theta}_i})_{i=1}^N$ are essentially different realizations of some shared canonical representation. In other words, the JA solution is the set of warping parameters, $(\boldsymbol{\theta}_i)_{i=1}^N$, that warps the entire image ensemble to some shared representation, usually referred to as an *atlas* (or *prototypical image*). Thus, the JA problem is often formulated as finding a latent atlas, I_μ, together with the aforementioned parameters:

$$\underset{I_\mu, (T^{\boldsymbol{\theta}_i})_{i=1}^N \in \mathcal{T}}{\arg\min} \sum_{i=1}^N D(I_\mu, I_i \circ T^{\boldsymbol{\theta}_i}). \tag{2}$$

JA Approach #1: Alignment to a Shared Atlas. One JA approach relies on building a shared atlas, I_μ, that minimizes Equation 2. The atlas could be

a learnable parameter [47] or the average of the warped images [8, 14, 27, 53], $I_\mu = \frac{1}{N} \sum_{i=1}^{N} I_i \circ T^{\theta_i}$. A key challenge in that approach is that since the $(\theta_i)_{i=1}^{N}$ are unknown, so is I_μ and thus the latter must be found during the optimization process together with $(\theta_i)_{i=1}^{N}$. Note that the optimization may also be amortized via the training of a neural net [13].

Unfortunately, Equation 2 can be undesirably minimized by, *e.g.*, (i) removing all images from the domain of I_μ or (ii) severely shrinking or stretching the images. Both (i) and (ii) represent poor *global* minima: the nonnegative loss becomes zero, yet the solution is useless. Additionally, the process heavily relies on a good initial alignment of the set and is prone to converge to poor local minima, which could be visually seen as a blurred representation of the set. See Erez *et al.* [13] for a detailed discussion on problems associated with that approach. One possible remedy to the issues above is to use a regularization over the predicted transformations (not to be confused with regularization over the network's weights). That is, the JA problem becomes:

$$\arg\min_{I_\mu, (T^{\theta_i})_{i=1}^{N} \in \mathcal{T}} \sum\nolimits_{i=1}^{N} D(I_\mu, I_i \circ T^{\theta_i}) + \mathcal{R}(T^{\theta_i}; \lambda), \qquad (3)$$

where the regularization term, $T^{\theta_i} \mapsto \mathcal{R}(T^{\theta_i}; \lambda)$ is parameterized by *hyperparameters* (HP), λ. For instance, one may hope, against hope, that penalizing large deviations from the identity transformation would guide the optimization process to a more plausible atlas. However, *there is a major problem here whose severity is often swept under the rug*: the need to determine good values for λ.

JA Approach #2: Congealing. The discussion above leads to another JA approach: congealing [32, 45]. The congealing algorithm seeks to minimize some JA criterion (*e.g.*, in its original formulation it was entropy) by iteratively warping each single image towards the other images but only after those images were already warped using the current estimates of their respective transformations. *One appealing property of congealing is that it defines a notion of central tendency* of the data without having to explicitly build an atlas or to rely on a reference image [32]. Congealing avoids some of the pitfalls of aligning to a shared atlas, though sometimes regularization is still used. It was later shown that least squares congealing (LSC) has several advantages over the entropy-based algorithm in terms of optimization and convergence [9]. Finally, the clever Inverse-Compositional LSC (IC-LSC) method was introduced to handle outlier images and objects being warped outside of the image domain when applying a single warp to the entire set [10]. Although not DL-based, IC-LSC is, in some sense, the closest to our approach.

Regularization-based DL Approaches for Joint Alignment. The aforementioned challenges in solving the JA problem still exist even when considering rich semantically-meaningful representations such as DINO [7]. Contemporary methods handle these issues by employing extensive regularization on the predicted warps and/or learned atlas. Consider ASIC [20], for instance, which seeks to densely map every input pixel to a canonical shared space. Dense mapping
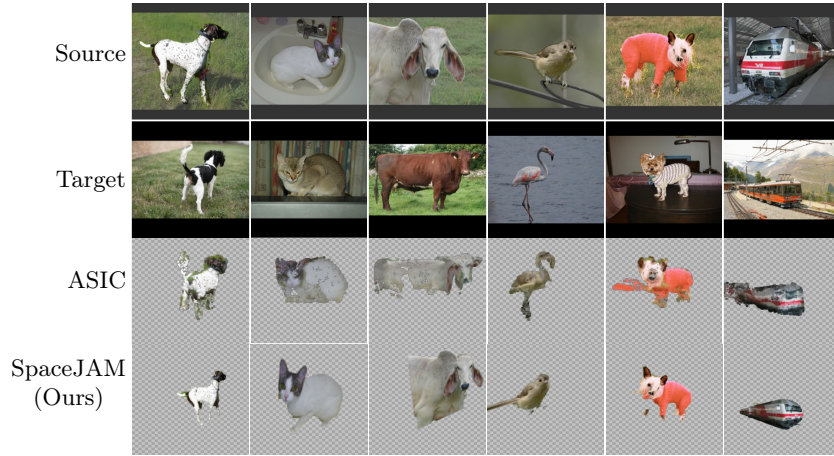
Fig. 3: Global alignment vs. dense warping evaluated on several classes of the SPair-71K dataset [46]. A source image (1st row) is mapped to the target image (2nd row) via either dense warping (3rd row) achieved by ASIC [20] (as presented in their paper) or a parametric alignment by SpaceJAM (ours, 4th row). Dense mapping is prone to produce incoherent results (please zoom in to better see that effect) and heavily relies on regularization. In comparison, our regularization-free method produces geometrically-coherent results and is also much faster.

strategies are inherently susceptible to breaking the object geometry (as shown in Figure 3). To mitigate this problem, the authors of [20] enforce the following regularization and auxiliary losses: (1) an equivariance loss to handle geometric variations, (2) total-variation regularization to encourage smooth mappings, (3) consistent part alignment to map semantically similar parts, combined with additional two losses that are not directly related to the object geometry, for a total of 5 regularization terms (while [49] has 6 reg. terms and 8 HPs). Each loss term requires its own HP, λ. *The reason that the crucial dependency on regularization terms, let alone so many of them, is much worse than one might expect is threefold.* First, the optimal choice of λ is usually dataset-specific and this renders such methods either too brittle or limited. Second, even on a single dataset, determining a good value typically requires some supervision – at odds with the unsupervised nature of the JA task. Third, the need to have several (sometimes many) loss terms, can complicate and prolong the optimization process. Thus, it is unsurprising that current DL-based JA methods such as [20, 49] must use expensive architectures and are slow.

4   Method

Given a collection of images, $(I_i)_{i=1}^N$, along with their DINO representations, $(V_i)_{i=1}^N$, of an object (*e.g.*, a particular bird) or an object class (*e.g.*, multiple birds), our goal is to jointly align the images in a semantically-meaningful manner.

Before describing the proposed method we emphasize that all of our design choices below were made with the goal of making the JA process easier, faster, and more stable. Together, those choices have a surprisingly strong synergistic effect, yielding a lightweight JA method that is *much* faster and its number of trainable parameters is $< 1\%$ of the numbers in contemporary methods.

4.1 Preprocessing

The goal of our preprocessing procedure is to extract object masks and reduce the dimensionality of the DINO features. Similarly to [20, 49], and to keep the comparison with those methods fair, we use [2] to produce the saliency-based **masks** for the image collection. Let M_i denote the binary mask for I_i. Next, we apply **dimensionality reduction** to the DINO features. Let $V_i \in \mathbb{R}^{d \times H \times W}$ where d is typically high (*e.g.*, $d = 384$) and (H, W) are the (height,width) of the DINO representation. Our dimensionality reduction has two steps.

To reduce the computational burden, we apply Principal Component Analysis (PCA) using the first K principal components, where $K < d$ (in our experiments, $K = 25$). Let $\hat{V}_i \in \mathbb{R}^{K \times H \times W}$ denote the PCA representation of V_i, and let $\tilde{V}_i = \hat{V}_i \cdot M_i$ denote its masked version.

In the second step, we train a fully convolutional, low-capacity autoencoder (only $\sim$3K parameters) to reduce the number of channels to 3. We chose 3 since it is low enough to serve as an information bottleneck that removes redundant details and since it allows for interpretability when visualizing the encoded features using RGB colors. The AE is trained to reconstruct $\tilde{V}_i$ using the following loss,

$$\mathcal{L}_{\text{AE}} = \sum_{i=1}^{N} \left\| \tilde{V}_i - \psi_{\text{dec}}(\psi_{\text{enc}}(\tilde{V}_i)) \right\|_{\ell_2}^2. \tag{4}$$

As we explain later, the obtained latent representation of the AE, $U_i \triangleq \psi_{\text{enc}}(\tilde{V}_i) \in \mathbb{R}^{3 \times H \times W}$, will serve a purpose in our alignment network. We find that the combination of PCA and AE to reduce the input's number of channels is highly effective from both computational and representational points of view.

4.2 Alignment Network

Our alignment network, ψ_{align}, is based on an STN [26]. Below we provide a brief overview of that STN and how we handle common pitfalls in STN training. Let ψ_{STN} denote the STN and, for now, let X denote its generic input. The core of the STN is the so-called **localization network**, denoted by ψ_{loc}, that predicts transformation parameters from X; *i.e.*, $\boldsymbol{\theta} = \psi_{\text{loc}}(X)$. The output of the entire STN is $(\boldsymbol{\theta}, X \circ T^{\boldsymbol{\theta}}) = \psi_{\text{STN}}(X)$. That is, the output is both $\boldsymbol{\theta}$ and the warped image obtained by applying $T^{\boldsymbol{\theta}}$ to X. STNs have been shown to be an effective tool for various tasks, albeit difficult to optimize [35, 63]. The issue stems from a few reasons, which we address in a theoretically-grounded manner.

The first difficulty is in predicting large transformations. To address this issue, the Inverse-Compositional STN (IC-STN) [35] was proposed, which, based on

the classical Inverse-Compositional Lucas & Kanade algorithm [42], predicts a cascade of smaller warps and composes them. In our case, we do so by recursively feeding the STN several times with its own output (see **SupMat** for details). Empirically, 5 recurrences suffice (a higher number yielded a very small gain) so this is what we used in our experiments. Since the recurrences share the same STN, the number of trainable parameters does not increase with the number of recurrences. As we apply several transformations in cascade, each predicted transformation can be relatively small. A fine nuance is that while usually each STN call involves interpolation (due to the resulting irregular grid), here whenever we compose transformations we perform only the last interpolation; this improves the quality of the resulting image as fewer interpolation artifacts occur. Our entire model, ψ_{align}, consists of the AE and 5 recurrences of ψ_{STN}.

4.3   Lie Algebras, Lie groups, and the Matrix Exponential

Another difficulty is that, without special care, an STN might predict non-invertibile transformation matrices, which could hinder the optimization process; *e.g.*, [63] showed that restricting an affine STN to invertible affine transformations allows for more robust and stable training. Thus, our alignment network, ψ_{align}, uses a cascade of transformations that are members of a Lie group. This formulation is more stable and robust compared to the vanilla STN. We represent spatial transformations as elements of Lie groups via a Lie-algebraic parameterization, as we now explain. Consider these 6 spaces of 3-by-3 real matrices:

$$\text{se}(2) = \left\{ \boldsymbol{A} : \boldsymbol{A} = \begin{bmatrix} 0 & \theta_2 & \theta_3 \\ -\theta_2 & 0 & \theta_6 \\ 0 & 0 & 0 \end{bmatrix} \right\} , \ \text{SE}(2) = \left\{ \boldsymbol{T} : \boldsymbol{T} = \begin{bmatrix} T_1 & T_2 & T_3 \\ T_4 & T_5 & T_6 \\ 0 & 0 & 1 \end{bmatrix} \& \begin{bmatrix} T_1 & T_2 \\ T_4 & T_5 \end{bmatrix} \in \text{SO}(2) \right\} ,$$

$$\text{aff}(2) = \left\{ \boldsymbol{A} : \boldsymbol{A} = \begin{bmatrix} \theta_1 & \theta_2 & \theta_3 \\ \theta_4 & \theta_5 & \theta_6 \\ 0 & 0 & 0 \end{bmatrix} \right\} , \ \text{Aff}(2) = \left\{ \boldsymbol{T} : \boldsymbol{T} = \begin{bmatrix} T_1 & T_2 & T_3 \\ T_4 & T_5 & T_6 \\ 0 & 0 & 1 \end{bmatrix} \& \det \boldsymbol{T} > 0 \right\} ,$$

$$\text{sl}(3) = \left\{ \boldsymbol{A} : \boldsymbol{A} = \begin{bmatrix} \theta_1 & \theta_2 & \theta_3 \\ \theta_4 & \theta_5 & \theta_6 \\ \theta_7 & \theta_8 & -(\theta_1+\theta_5) \end{bmatrix} \right\} , \ \text{SL}(3) = \left\{ \boldsymbol{H} : \boldsymbol{H} = \begin{bmatrix} H_1 & H_2 & H_3 \\ H_4 & H_5 & H_6 \\ H_7 & H_8 & H_9 \end{bmatrix} \& \det \boldsymbol{H} = 1 \right\} .$$

The shortcuts SE (or se) Aff (or aff), SL (or sl), and SO stand for "Special Euclidean", "Affine", "Special Linear", and "Special Orthogonal", respectively. The following are well-known mathematical facts, widely used in computer vision (see, *e.g.*, [8, 13, 36, 37]): 1) se(2), aff(2), and sl(3) are *linear* spaces, called (matrix) **Lie algebras**. 2) SE(2), Aff(2), and SL(3) are *nonlinear* spaces. Moreover, they are (matrix) **Lie groups**; namely, each of these spaces is both a group (the binary operator being matrix multiplication) and a smooth manifold, and the group operations (multiplication; inversion) are smooth. 3) Each element of each of those groups is an **invertible matrix**. 4) **Nesting property:** se(2) is a linear subspace of aff(2) and the latter is a linear subspace of sl(3). Likewise, SE(2) is a matrix subgroup of Aff(2) and the latter is a matrix subgroup of SL(3). 5) SE(2) is the group of rigid-body transformations in $\mathbb{R}^2$. Aff(3) is the group of (orientation-preserving) **invertible affine transformations** in $\mathbb{R}^2$. SL(3), the group of volume-preserving linear transformations in $\mathbb{R}^3$, can also be identified with the group of **homographies** in $\mathbb{R}^2$ (this is why we denoted

its generic element by $\boldsymbol{H}$). 6) The **matrix exponential** maps $se(2) \to SE(2)$, $aff(2) \to Aff(2)$, and $sl(3) \to SL(3)$.

The matrix exponential provides a differentiable map from a linear space (the algebra) into its corresponding nonlinear space (the group). This implies that, using the Lie-algebraic parametrization, an STN [26] can be easily restricted to yield only elements of the Lie group of interest [63]. This is useful since while optimization over nonlinear manifolds can be hard (*e.g.*, even a simple gradient-descent step might bring us outside the manifold), the Lie-algebraic parametrization, together with the matrix exponential, lets us perform gradient-based optimization in a linear space (for other methods for optimization on manifolds, see Boumal's textbook [5]) In fact, [63] showed that an STN restricted to *invertible* affine transformations was more stable and yielded better results than the commonly-used unrestricted STN. In our case, there is also an additional reason why we use Lie groups and this is the fact that our approach relies directly on Lie groups' closure under inversion and composition (as detailed below).

4.4   Loss Function

Recall that, for image I_i, we let V_i, U_i, and M_i denote its DINO representation, an encoded 3-channel representation, and mask map, respectively. We train ψ_{align} to find the JA of $(I_i)_{i=1}^{N}$ in a forward-inverse compositional manner. Part of the motivation behind our loss is to avoid the drawbacks of explicitly warping the images to a shared space (see § 3). In particular, we avoid using regularization terms over the transformations. Our loss consists of two conceptual steps: (i) for each image, I_i, in batch k – denoted by $B_k = (I_i)_{i=1}^{N_b}$ (where N_b is the # of images in the batch) – predict the forward warping parameters, $\boldsymbol{\theta}_i$, using U_i as the recurrent STN's input, and then (ii) compose the forward warp with the inverse warp of every other image in the batch, to build the following loss:

$$\mathcal{L}_{\text{IC}} = \sum_{i=1}^{N_b} \sum_{j:j\neq i} \left\| \tilde{V}_j - (\tilde{V}_i \circ T^{\boldsymbol{\theta}_i} \circ T^{-\boldsymbol{\theta}_j}) \right\|_{\ell_2}^2 . \tag{5}$$

Minimizing this loss implicitly maps each image to the shared space, since

$$\tilde{V}_j \approx \tilde{V}_i \circ T^{\boldsymbol{\theta}_i} \circ T^{-\boldsymbol{\theta}_j} \Leftrightarrow \tilde{V}_j \circ T^{\boldsymbol{\theta}_j} \approx \tilde{V}_i \circ T^{\boldsymbol{\theta}_i} . \tag{6}$$

The loss in Equation 5 relies on Lie groups' closure under inversion (*i.e.*, the matrix associated with $T^{\boldsymbol{\theta}}$ is invertible) and composition. Notably, the proposed loss effectively frees us from having to use regularization. This is because, by construction, a "bad" predicted transformation would directly increase the loss as the corresponding warped image would fail to "explain away" the other images, unless all the predicted transformations would be "bad" in *exactly* the same way (which is unlikely and indeed never happens in practice).

Training is done via standard DL gradient-based optimization, except we also use a Lie-algebraic curriculum learning to ease the optimization further. Meaning, exploiting the nested structure of the Lie algebras, we start the training

with SE(2) and end with homographies. See **SupMat** for details. After training, SpaceJAM jointly aligns the entire image collection using its forward pass.

Reflections. The outline of our proposed solution for handling reflections (*i.e.*, flips), whose full details are in our **SupMat**, is as follows. During training, we dynamically change the selected flip configuration and train on all configurations simultaneously (this yields smooth and robust training) but save computations and increase stability by *calculating gradients only for the best flip configuration.*

Atlas Building. Upon learning the JA, the method can also quickly generate an atlas as follows. Given the images $(I_i)_{i=1}^{N}$, we predict the forward warping parameters via SpaceJAM's forward pass and obtain the aligned images via $(I_i \circ T^{\boldsymbol{\theta}_i})_{i=1}^{N}$. The atlas can be taken as the sample mean of either the warped features, $\frac{1}{N}\sum_{i=1}^{N} V_i \circ T^{\boldsymbol{\theta}_i}$, or the warped PCA-related versions, $\frac{1}{N}\sum_{i=1}^{N} \tilde{V}_i \circ T^{\boldsymbol{\theta}_i}$, Either way results in a semantically-meaningful atlas; see, Figure 4.

4.5   Implementation

We use simple architectures. ψ_{loc} is a 2-layer CNN with $\sim$13K trainable parameters and ψ_{AE} is a fully-convolutional AE with $\sim$3K trainable parameters, so the total is only 16K. The final layer predicts 8 parameters (the number of degrees of freedom in a homography) in total. We predict a cascade of 5 warps in total, where ψ_{align} is shared throughout the process (akin to standard IC-STN [35]). SpaceJAM is implemented in `PyTorch` [51] and optimized for 300 epochs for ψ_{AE} and then an additional 400 for the entire framework ($\psi_{\mathrm{AE}}+\psi_{\mathrm{align}}$), using the Adam optimizer [31] and a StepLR scheduler ($\gamma = 0.9$, every 50 epochs).

5   Results

We evaluate our method on several real-world benchmarks, containing object categories with different illuminations, visual clutter, and occlusions, and compare it with several JA methods. The **Datasets** include **SPair-71k** [46], which consists of 1,800 images from 18 categories (*i.e.*, bird). As is common in the weakly-supervised test-time optimization scenario, we train (in a weakly-supervised manner; *i.e.* only the categories are known, but the latent alignments are not) SpaceJAM on each of the SPair-71k test sets independently and report results for each category. For **CUB-200** [65], we 1) report results on its first 3 categories, consisting of 25-30 images each when comparing with [20] and 2) the average of 14 subsets, following the protocol in [49, 52]. For a fair comparison, in the quantitative comparison our method, like [20, 49], used DINOv1 (ViT-S/8) [7]. For completeness and future comparisons, we also report our results with DINOv2 [50].

5.1   Qualitative Results

Recall that SpaceJAM is trained in an inverse-compositional framework which first aligns an image to a shared space and then warps it to all other images in the batch using the inverse transform of the target. This allows us to compute (after

Fig. 4: Pairwise and joint alignment using SpaceJAM. The input images (1th row) overlayed by the learned features (2nd row) which are used to predict the warping parameters. The 3rd row shows source-to-target alignment under severe conditions and the 4th row shows the jointly aligned features and the category atlas (right column).

the training is done) a shared atlas without explicitly maintaining one during training. Examples of our method's JA and resulting atlas can be seen in Figure 1 and Figure 4. Figure 3 provides a visual comparison between SpaceJAM and ASIC in terms of pairwise alignment (for AISC, we use the visual results appearing in [20]). Evidently, SpaceJAM achieves high fidelity and geometrically-coherent alignment under challenging conditions. Figure 4 shows how the learned low-dimensional features ($U_i \in \mathbb{R}^{3 \times H \times W}$; second row) allow us to perform both pairwise alignment (3rd row) and JA (4th row) in an interpretable manner, where semantically-related parts (*e.g.*, faces) share the same color across images. The visual results indicate that SpaceJAM can align diverse collections of images efficiently and accurately (see the **SupMat** for more visual results).

5.2   Quantitative Evaluation

We assess our method on the semantic point correspondence task using the SPair-71k [46] and CUB-200 [65] datasets, which are widely recognized benchmarks in both pairwise and JA domains. The performance is quantified by the PCK-Transfer metric, which calculates the proportion of keypoints accurately aligned within a threshold of $\alpha \cdot \max(h, w)$ relative to the true position. Following established protocols [49], $\alpha = 0.1$ (denoted as PCK@0.10) for both datasets, with (h, w) representing the dimensions of the object's bounding box. We perform pairwise alignment using Equation 6.

In line with the comparative framework established by [20], we categorize existing methods into 3 groups: (1) *Strong supervision*, which relies on manually annotated keypoints; (2) *GAN supervision*, using category-specific GANs and; (3) *Weak supervision*, using category-level supervision without explicit keypoints.

Table 2: SPair-71k Dataset results: Per-class and overall mean PCK@0.10 evaluated on the test set. The best result among *weakly-supervised* methods is boldfaced (the runner-up is underlined). ($\star$) indicates that the method used a reference image. Missing values ($-$) indicate unreported results (we, like [20], found that running the code from [49] usually did not converge successfully).

Supervision	Method	Aero	Bike	Bird	Boat	Bottle	Bus	Car	Cat	Chair	Cow	Dog	Horse	Motor	Person	Plant	Sheep	Train	TV	All
Strong Supervision	SCorrSAN [25]	57.1	40.3	78.3	38.1	51.8	57.8	47.1	67.9	25.2	71.3	63.9	49.3	45.3	49.8	48.8	40.3	77.7	69.7	55.3
GAN supervision	GANgealing [52]	-	37.5	-	-	-	-	-	67.0	-	-	23.1	-	-	-	-	-	-	57.9	-
Weak supervision (train/test)	CNNGeo [54]	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 [60]	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 [55]	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
	NCNet [56]	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
	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	32.2	35.9	26.3
	PMD [34]	26.2	18.5	48.6	15.3	**38.0**	21.7	17.3	51.6	13.7	34.3	25.4	18.0	20.0	24.9	15.7	16.3	31.4	_38.1_	26.5
	PSCNet-SE [29]	28.3	17.7	45.1	15.1	37.5	30.1	_27.5_	47.4	14.6	32.5	26.4	17.7	24.9	24.5	19.9	16.9	34.2	37.9	27.0
Weak supervision (test-time optimization)	VGG+MLS [1]	29.5	22.7	61.9	**26.5**	20.6	25.4	14.1	23.7	14.2	27.6	30.0	29.1	24.7	27.4	19.1	19.3	24.4	22.6	27.4
	DINO+MLS [1,6]	49.7	20.9	63.9	19.1	32.5	27.6	22.4	48.9	14.0	36.9	39.0	30.1	21.7	41.1	17.1	18.1	35.9	21.4	31.1
	DINO+NN [2]	_57.2_	24.1	_67.4_	24.5	26.8	29.0	27.1	52.1	_15.7_	_42.4_	_43.3_	30.1	23.2	40.7	16.6	24.1	31.0	24.9	33.3
	NeuCongeal [49]	-	**29.1***	-	-	-	-	-	53.3	-	-	35.2	-	-	-	-	-	-	-	-
	ASIC [20]	**57.9**	_25.2_	**68.1**	_24.7_	35.4	28.4	**30.9**	54.8	**21.6**	**45.0**	**47.2**	_39.9_	26.2	**48.8**	14.5	_24.5_	_49.0_	24.6	_36.9_
	SpaceJAM (DINOv1, ViT-S)	43.4	22.0	34.5	18.8	32.4	**36.3**	24.1	**60.8**	06.5	37.1	35.9	**46.7**	**53.6**	_46.9_	**36.0**	**24.6**	**75.0**	**46.0**	**37.8**
	SpaceJAM (DINOv2, ViT-B)	52.9	48.9	48.1	25.1	45.8	53.7	47.9	61.2	18.7	40.5	51.5	49.8	58.3	56.3	31.8	42.7	59.9	59.7	42.0
	SpaceJAM (DINOv2, ViT-L)	53.6	53.4	45.4	47.5	71.0	66.8	59.5	62.7	19.8	59.2	56.2	63.8	62.1	59.6	33.1	48.1	68.0	61.6	45.7

A subset of these methods adopts a train/test paradigm, using a large amount of data. Unlike those works, SpaceJAM involves training a compact model and leverages a test-time optimization scheme. The baseline scores, obtained from [20], include: 1) DINO nearest neighbor (DINO+NN; [2]); 2) Neural Congealing [49], which builds a shared atlas; and 3) ASIC [20] which densely maps pixels to a canonical grid (both are weakly-supervised JA methods).

Table 2 shows PCK@0.1 for all classes of the SPair-71k dataset [46]. We report the average of 3 runs for SpaceJAM. Our method performs best on average for all object classes (**All**) in the *weak supervision* category, using a much lighter model that trains 10x faster. It also performs best and second-best in 9 out of 18 classes. The results are consistent for both rigid objects (*i.e.*, 'Train') and ones with extreme variations ('Cat') with the exception of the 'Chair' category, where it was unable to overcome the initial coarse masks provided by [2]. Table 3 shows the results on the CUB-200 dataset [65] as reported in [20], where our method is the runner-up after [20]. That dataset consists of only birds and offers less variety than SPair-71K. When comparing to [49,52], on 14 subsets, our method outperforms both. Please see our **SupMat** for further discussion of the results.

5.3 Complexity

SpaceJAM is a lightweight model, consisting of only 16K (*i.e.*, 0.016M) trainable parameters. We compare our method to [20,49] using the official implementations. As Table 1 shows, SpaceJAM's # of trainable parameters is *much* lower than those of [49] and [20] which are 28.7M and 7.9M, respectively. Table 1 also shows that SpaceJAM achieves convergence within only 300 AE pretraining epochs and 400 AE+STN epochs, translating to a training time of ≈ 6 minutes on a single RTX4090, which starkly contrasts with the *hours* required by the competing methods. Despite its compact design, SpaceJAM attains a PCK@0.1 score of 60.8 on the 'Cats' dataset, outperforming both competitors. Evidently, SpaceJAM's

Table 3: A comparison on CUB-200.

Method	CUB-200 (3 cate.)	Method	CUB-200 (Subsets)
VGG+MLS [1]	25.8	-	-
DINO+MLS [1,6]	67.0	-	-
DINO+NN [2]	68.3	GANgealing [52]	56.8
ASIC [20]	**75.9**	NeuCongeal [49]	63.6
SpaceJAM	69.6	SpaceJAM	**69.9**

Table 4: Ablation Study.

Ablation	CUB-200 (3 cate.)	SPair-71k (3 cate.)
Alignment to atlas	58.5	54.6
No AE pre-training	54.9	56.6
No Curiculum	70.8	57.2
No Lie Groups	65.6	57.1
STN	61.6	43.5
ICSTN-3	72.6	57.6
Complete model	**73.3**	**58.1**

efficiency does not hurt performance, as it not only accelerates the training process but also enhances alignment accuracy.

5.4   Ablation Study & Limitations

Ablation. We conducted an ablation study for SpaceJAM (using DINOv2 [50]) on subsets of the CUB-200 and SPair-71k datasets and evaluated them using PCK@0.1, as shown in Table 4. Due to the low-capacity of the models used in SpaceJAM, the AE pre-training proves essential before the JA. Predicting a cascade of 5 small transformations shows significant improvement over a single large one (STN) or three (ICSTN-3), when the # of trainable parameters is the same. An alignment to an atlas, even when using the robust Inverse Consistency Averaging Error (ICAE [67]) performs worse than our inverse-compositional framework. Table 4 also shows that using the Lie groups is crucial and that the curriculum learning's effect, while positive, is less significant.

   Limitations.   SpaceJAM, while being an JA effective method, was not designed for pairwise dense correspondences. Thus, it was not optimized for maximizing the per-pixel agreement between an image pair. That said, § 5 shows that the speed gains in speed are drastic and that the method is much faster than contemporary JA methods [20, 49] while its dense mapping score (a standard evaluation metric for both pairwise and JA tasks) is still comparable (and in fact, in some cases better). This trade-off also comes into play in the choice of the transformation family. As with other STNs [26,59,63], it is possible to incorporate more expressive transformations such as diffeomorphisms (as in, *e.g.*, [15,16,63]) which are richer than homographies. Lastly, our performance also partially relies on the quality of the initial masks, a limitation we share with [20,49].

6   Conclusion

SpaceJAM is a lightweight method that effectively performs JA with neither regularization terms nor building an explicit atlas during the process. It demonstrates excellent performance on benchmarks like SPair-71K and CUB, on par with, or outperforming, existing methods despite having far less trainable parameters and a much shorter running time. While here we mentioned only 3 matrix groups, both our method and code support additional ones (*e.g.*, the similarity group).

SpaceJAM: a Lightweight and Regularization-free Method for Fast Joint Alignment of Images Supplemental Material

Nir Barel* ◉, Ron Shapira Weber* ◉, Nir Mualem◉, Shahaf E. Finder◉, and Oren Freifeld◉

The Department of Computer Science, Ben-Gurion University of the Negev, Israel
{banir,ronsha,nirmu,finders}@post.bgu.ac.il, orenfr@cs.bgu.ac.il

1 Handling flips

As detailed in the main manuscript, flips require special care. Let (I_i, I_j) be a source and target images respectively (for simplicity, we drop the DINO ViT and learned features notion and use this notion). In this particular case, our loss function reduces to

$$\mathcal{L}_{\text{IC}} = \left\| I_j - (I_i \circ T^{\boldsymbol{\theta}_i} \circ T^{-\boldsymbol{\theta}_j}) \right\|_{\ell_2}^2 . \tag{1}$$

To incorporate flips efficiently, we consider only horizontal flips (since vertical flips could be reached through a horizontal flip + rotation) and compute the gradient only between the best matching pair. Particularly, let F^{k_i} be the kth flip configuration applied to the ith image, where $k \in C$ such that C holds the possible configuration (in our case, 2). The objective function is now

$$\mathcal{L}_{\text{IC}} = \sum_{i=1}^{N} \sum_{k_i=1}^{|C|} \min_{k_j \in C} \left\| I_j \circ F^{k_j} - ((I_i \circ T^{\boldsymbol{\theta}_i}) \circ F^{k_i}) \circ T^{-\boldsymbol{\theta}_j} \right\|_{\ell_2}^2 , \tag{2}$$

where (k_i, k_j) are the flips considered for the image pair (I_i, I_j).

2 Curriculum learning

To incorporate the Lie-algebraic curriculum learning during training, we gradually add more complex transformation modules, starting from SE(2) and later "release" more of the transformation parameters, to obtain (invertible) affine transformations and finally homographies. Figure 1 illustrate the process, where additional transformation parameters are "released", as illustrated by the warped images above the training timeline.
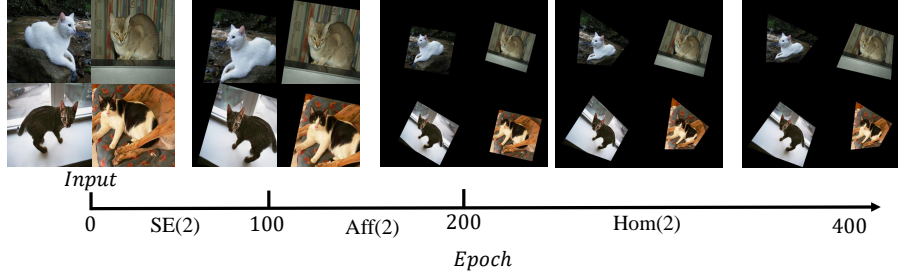
Fig. 1: Lie Algebric curriculum learning. The notation SE(2) between the epochs $(0, 100)$ states that during that interval, the training is restricted to SE(2). At epoch 100, more transformation parameters are "released" to allow for affine transformations.

3 Inverse-Compositional STN (IC-STN) [35]

The IC-STN [35], based on the classical Inverse-Compositional Lucas & Kanade algorithm [42], predicts a cascade of smaller warps and composes them. In our case, we do so by recursively feeding the STN several times with its own output. In effect,

$$(\boldsymbol{\theta}, X \circ T^{\boldsymbol{\theta}}) = \psi_{\text{STN}}(X) \tag{3}$$

$$(\boldsymbol{\theta}', X \circ T^{\boldsymbol{\theta}} \circ T^{\boldsymbol{\theta}'}) = \psi_{\text{STN}}(X \circ T^{\boldsymbol{\theta}}) \tag{4}$$

$$(\boldsymbol{\theta}'', X \circ T^{\boldsymbol{\theta}} \circ T^{\boldsymbol{\theta}'} \circ T^{\boldsymbol{\theta}''}) = \psi_{\text{STN}}(X \circ T^{\boldsymbol{\theta}} \circ T^{\boldsymbol{\theta}'}) \tag{5}$$

and so on.

4 An Additional runtime comparison

We provide an additional runtime comparison with Neural Congealing [49] and ASIC [20] on the 'Dog' and 'Bike' datasets [46]. The results are presented in Table 1.

Table 1: A comparison with recent JA methods and evaluation on 3 SPair-71K categories [46].

Method	# Params	# Losses	#HP	Atlas-free learning	#epochs	Cat	Time Bike	Dog
NeuCongealing [49]	28.7M	8	8	✗	8K	1:17:02	1:12:55	1:25:28
ASIC [20]	7.9M	4	5	✗	20K	1:06:48	1:07:40	1:06:11
SpaceJAM (Ours)	**0.016M**	**1**	**0**	✓	**0.7K**	**0:05:58**	**0:06:11**	**00:05:43**

5   Architectures

A detailed overview of the Autoencoder (AE) and Spatial Transformer Network (STN) architectures and the number of trainable parameters. Together they form our alignment module - ψ_{align}.

Table 2: Autoencoder Model Summary. GFMN = GlobalFeatureMapNormalizer.

Layer (type:depth-idx)	Output Shape	Param
Autoencoder	[1, 25, 256, 256]	–
Encoder: 1-1	[1, 3, 256, 256]	–
Sequential: 2-1	[1, 3, 256, 256]	–
Conv2d: 3-1	[1, 32, 256, 256]	832
ReLU: 3-2	[1, 32, 256, 256]	–
BatchNorm2d: 3-3	[1, 32, 256, 256]	64
Conv2d: 3-4	[1, 16, 256, 256]	528
ReLU: 3-5	[1, 16, 256, 256]	–
BatchNorm2d: 3-6	[1, 16, 256, 256]	32
Conv2d: 3-7	[1, 3, 256, 256]	51
GFMN: 3-8	[1, 3, 256, 256]	–
Decoder: 1-2	[1, 25, 256, 256]	–
Sequential: 2-2	[1, 25, 256, 256]	–
Conv2d: 3-9	[1, 16, 256, 256]	64
ReLU: 3-10	[1, 16, 256, 256]	–
BatchNorm2d: 3-11	[1, 16, 256, 256]	32
Conv2d: 3-12	[1, 32, 256, 256]	544
ReLU: 3-13	[1, 32, 256, 256]	–
BatchNorm2d: 3-14	[1, 32, 256, 256]	64
Conv2d: 3-15	[1, 25, 256, 256]	825

Table 3: STN Model Summary.

Layer (type:depth-idx)	Output Shape	Param
Conv2d-1	[1, 10, 250, 250]	1,480
AdaptiveMaxPool2d-2	[1, 10, 32, 32]	0
ReLU-3	[1, 10, 32, 32]	0
Conv2d-4	[1, 5, 28, 28]	1,255
AdaptiveMaxPool2d-5	[1, 5, 8, 8]	0
ReLU-6	[1, 5, 8, 8]	0
Linear-1	[1, 1, 32]	10,272
ReLU-2	[1, 1, 32]	0
Linear-3	[1, 1, 9]	297

We also evaluate the effect of the STN size on the resulting alignment. Figure 2 shows the average PCK@0.1 of 5 runs for the 3 subsets of the CUB200 datasets [65]. We increase the number of trainable parameters by using the same STN architecture with larger convolutional blocks in terms of # kernels and their size. Notably, the performance effectively saturates at as early as ∼15K parameters. Increasing the model further even to 24M parameters, does not results in additional gains.

6   Further Discussion of the Results

A natural question arises – why do different models perform better in some classes and worse in others? For instance, consider the 'Dogs' class of the SPair dataset. In the results presented in Table 2 in the main paper, ASIC outperforms the proposed method on that class by approximately 11 points, suggesting superior alignment. However, the visual comparison in Figure 3 reveals that dense-correspondence methods like ASIC often result in incoherent alignment. The warped images display artifacts such as holes, and the dog faces become unrecognizable. This discrepancy arises because benchmarks like SPair and CUB-200 focus on the sparse correspondence of hand-picked key points rather than measuring global
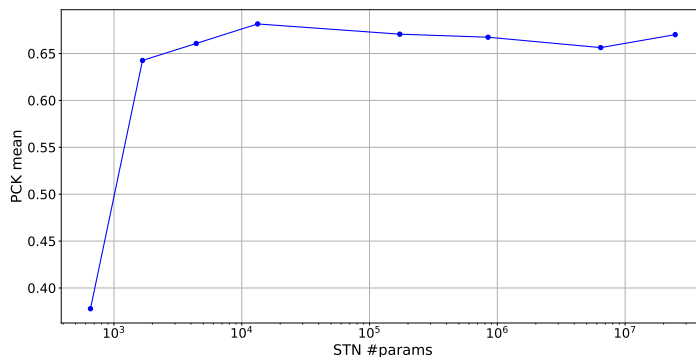
Fig. 2: Average PCK@0.1 score as a function of # trainable parameters of the STN (the x-axis is log-scaled). The model reaches saturation around ∼ 15K parameters.

(a) DINO+NN   **(b)** ASIC   **(c)** SpaceJAM (ours)

Fig. 3: Geometric fidelity of transformed images (DINO+NN and ASIC results were obtained from [20]).

alignment. In fact, the basic DINO-NN outperforms SpaceJAM on the same 'Dog' class, but yields significantly poorer visual results, as illustrated in Figure 3. This highlights the limitations of the DINO-NN approach. Additionally, our method outperforms ASIC in more than half of the classes while requiring 100x fewer parameters and achieving a 10x reduction in training time. Finally, the variance in results can also be attributed to the small set size (20-30 images) compared to the diverse poses, illuminations, and occlusions present in each set.

7   Additional Visualizations

7.1   Additional joint alignment results

More visual results of SpaceJAM's joint alignment (JA) are presented below (Figures 2-6) for the SPair-71K [46] and Samurai ('robot') [4] datasets. The figures show, from top-to-bottom: 1) input images; 2) DINO ViT features (first 3 PCs); 3) learned features 4) aligned features, and 5) aligned images. The aligned features and images are masked by the intersection of the coarse input mask and the median mask of the set (both after alignment). The atlas of the set appears at the bottom right.

Fig. 4: Joint alignment results - "train".

Fig. 5: Joint alignment results - "cat".

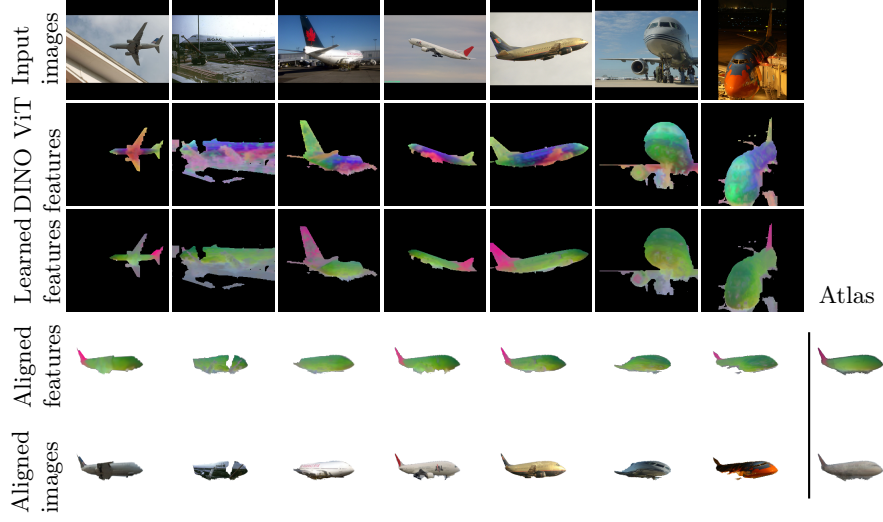Fig. 6: Joint alignment results - "robot".

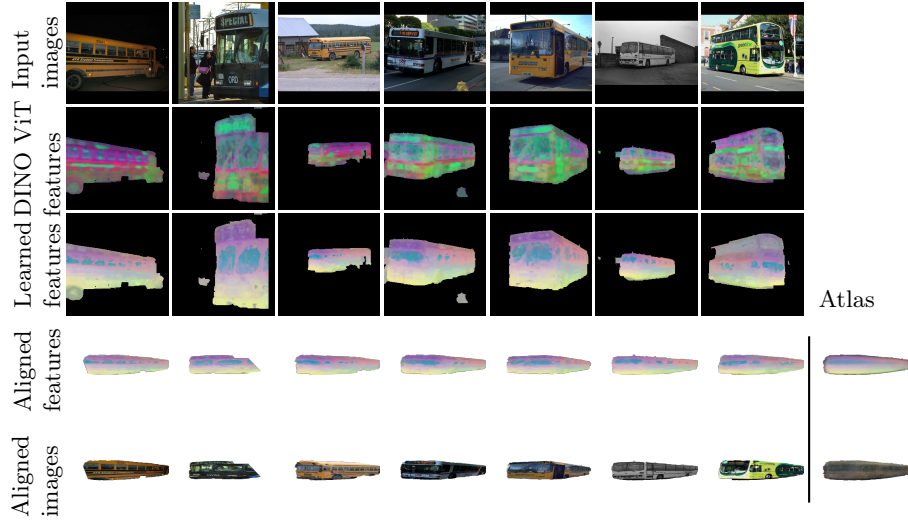**Fig. 7: Joint alignment results - "plane".**

Fig. 8: Joint alignment results - "bus".

7.2   Additional pairwise alignment results

More visual results of SpaceJAM's pairwise alignment are presented below. The figures show, from top-to-bottom: 1) input images; 2) learned features overlay; 3-7) Source-to-target pairwise alignment, where the image in the red square is aligned to all other images.
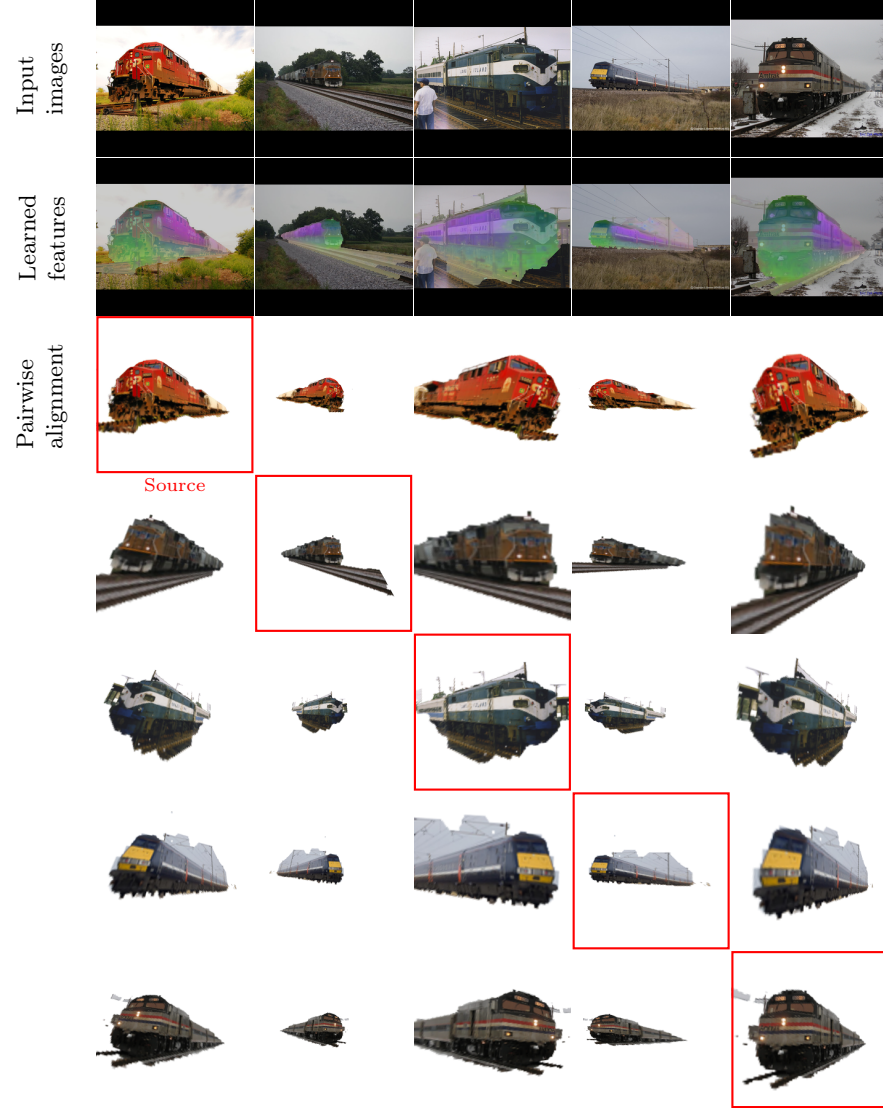

Fig. 9: Pairwise alignment results - "train".

Fig. 10: Pairwise alignment results - "bus".

References

1. Aberman, K., Liao, J., Shi, M., Lischinski, D., Chen, B., Cohen-Or, D.: Neural best-buddies: Sparse cross-domain correspondence. ACM TOG (2018)
2. Amir, S., Gandelsman, Y., Bagon, S., Dekel, T.: Deep vit features as dense visual descriptors. In: ECCV Workshops (2022)
3. Annunziata, R., Sagonas, C., Cali, J.: Jointly aligning millions of images with deep penalised reconstruction congealing. In: ICCV (2019)

4. Boss, M., Engelhardt, A., Kar, A., Li, Y., Sun, D., Barron, J., Lensch, H., Jampani, V.: Samurai: Shape and material from unconstrained real-world arbitrary image collections. Advances in Neural Information Processing Systems **35**, 26389–26403 (2022)
5. Boumal, N.: An introduction to optimization on smooth manifolds. Cambridge University Press (2023)
6. Caron, M., Misra, I., Mairal, J., Goyal, P., Bojanowski, P., Joulin, A.: Unsupervised learning of visual features by contrasting cluster assignments. NeurIPS (2020)
7. Caron, M., Touvron, H., Misra, I., Jégou, H., Mairal, J., Bojanowski, P., Joulin, A.: Emerging properties in self-supervised vision transformers. In: ICCV (2021)
8. Chelly, I., Winter, V., Litvak, D., Rosen, D., Freifeld, O.: JA-POLS: a moving-camera background model via joint alignment and partially-overlapping local subspaces. In: CVPR (2020)
9. Cox, M., Sridharan, S., Lucey, S., Cohn, J.: Least squares congealing for unsupervised alignment of images. In: CVPR (2008)
10. Cox, M., Sridharan, S., Lucey, S., Cohn, J.: Least-squares congealing for large numbers of images. In: ICCV. IEEE (2009)
11. Dalca, A., Rakic, M., Guttag, J., Sabuncu, M.: Learning conditional deformable templates with convolutional networks. NeurIPS (2019)
12. Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., et al.: An image is worth 16x16 words: Transformers for image recognition at scale. arXiv preprint arXiv:2010.11929 (2020)
13. Erez, G., Weber, R.S., Freifeld, O.: A deep moving-camera background model. In: ECCV. Springer (2022)
14. Felzenszwalb, P.F., Schwartz, J.D.: Hierarchical matching of deformable shapes. In: CVPR. pp. 1–8. IEEE (2007)
15. Freifeld, O., Hauberg, S., Batmanghelich, K., Fisher III, J.W.: Highly-expressive spaces of well-behaved transformations: Keeping it simple. In: ICCV (2015)
16. Freifeld, O., Hauberg, S., Batmanghelich, K., Fisher III, J.W.: Transformations based on continuous piecewise-affine velocity fields. IEEE TPAMI (2017)
17. Frey, B.J., Jojic, N.: Estimating mixture models of images and inferring spatial transformations using the em algorithm. In: CVPR. IEEE (1999)
18. Gavrila, D.M.: Multi-feature hierarchical template matching using distance transforms. In: ICPR. IEEE (1998)
19. Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., Bengio, Y.: Generative adversarial nets. NeurIPS (2014)
20. Gupta, K., Jampani, V., Esteves, C., Shrivastava, A., Makadia, A., Snavely, N., Kar, A.: Asic: Aligning sparse in-the-wild image collections. In: ICCV (2023)
21. He, J., Zhang, D., Balzano, L., Tao, T.: Iterative grassmannian optimization for robust image alignment. Image and Vision Computing (2014)
22. He, K., Zhang, X., Ren, S., Sun, J.: Identity mappings in deep residual networks. In: ECCV. Springer (2016)
23. Huang, G., Mattar, M., Lee, H., Learned-Miller, E.G.: Learning to align from scratch. In: NeurIPS (2012)
24. Huang, G.B., Jain, V., Learned-Miller, E.: Unsupervised joint alignment of complex images. In: ICCV. IEEE (2007)
25. Huang, S., Yang, L., He, B., Zhang, S., He, X., Shrivastava, A.: Learning semantic correspondence with sparse annotations. In: ECCV. pp. 267–284. Springer (2022)
26. Jaderberg, M., Simonyan, K., Zisserman, A., et al.: Spatial transformer networks. In: NeurIPS (2015)

27. Jain, A.K., Zhong, Y., Lakshmanan, S.: Object matching using deformable templates. IEEE TPAMI (1996)
28. Jeon, S., Kim, S., Min, D., Sohn, K.: Parn: Pyramidal affine regression networks for dense semantic correspondence. In: ECCV (2018)
29. Jeon, S., Kim, S., Min, D., Sohn, K.: Pyramidal semantic correspondence networks. IEEE TPAMI (2021)
30. Kemelmacher-Shlizerman, I., Seitz, S.M.: Collection flow. In: CVPR. IEEE (2012)
31. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. CoRR (2014), http://arxiv.org/abs/1412.6980
32. Learned-Miller, E.G.: Data driven image models through continuous joint alignment. IEEE TPAMI (2006)
33. Lee, J., Kim, D., Ponce, J., Ham, B.: Sfnet: Learning object-aware semantic correspondence. In: CVPR (2019)
34. Li, X., Fan, D.P., Yang, F., Luo, A., Cheng, H., Liu, Z.: Probabilistic model distillation for semantic correspondence. In: CVPR (2021)
35. Lin, C.H., Lucey, S.: Inverse compositional spatial transformer networks. In: CVPR (2017)
36. Lin, D., Grimson, E., Fisher III, J.: Learning visual flows: A Lie algebraic approach. In: CVPR (2009)
37. Lin, D., Grimson, E., Fisher III, J.: Modeling and estimating persistent motion with geometric flows. In: CVPR (2010)
38. Lin, W.Y., Liu, L., Matsushita, Y., Low, K.L., Liu, S.: Aligning images in the wild. In: CVPR. IEEE (2012)
39. Liu, X., Tong, Y., Wheeler, F.W.: Simultaneous alignment and clustering for an image ensemble. In: ICCV. IEEE (2009)
40. Loiseau, R., Monnier, T., Aubry, M., Landrieu, L.: Representing shape collections with alignment-aware linear models. In: 3DV. IEEE (2021)
41. Lowe, D.G.: Object recognition from local scale-invariant features. In: ICCV. IEEE (1999)
42. Lucas, B.D., Kanade, T.: An iterative image registration technique with an application to stereo vision. In: IJCAI (1981)
43. Mariotti, O., Mac Aodha, O., Bilen, H.: Improving semantic correspondence with viewpoint-guided spherical maps. arXiv preprint arXiv:2312.13216 (2023)
44. Mattar, M.A., Hanson, A.R., Learned-Miller, E.G.: Unsupervised joint alignment and clustering using bayesian nonparametrics. arXiv preprint arXiv:1210.4892 (2012)
45. Miller, E.G., Matsakis, N.E., Viola, P.A.: Learning from one example through shared densities on transforms. In: CVPR. IEEE (2000)
46. Min, J., Lee, J., Ponce, J., Cho, M.: Spair-71k: A large-scale benchmark for semantic correspondence. arXiv preprint arXiv:1908.10543 (2019)
47. Monnier, T., Groueix, T., Aubry, M.: Deep transformation-invariant clustering. NeurIPS (2020)
48. Mu, J., De Mello, S., Yu, Z., Vasconcelos, N., Wang, X., Kautz, J., Liu, S.: Coordgan: Self-supervised dense correspondences emerge from gans. In: CVPR (2022)
49. Ofri-Amar, D., Geyer, M., Kasten, Y., Dekel, T.: Neural congealing: Aligning images to a joint semantic atlas. In: CVPR (2023)
50. Oquab, M., Darcet, T., Moutakanni, T., Vo, H., Szafraniec, M., Khalidov, V., Fernandez, P., Haziza, D., Massa, F., El-Nouby, A., et al.: Dinov2: Learning robust visual features without supervision. arXiv preprint arXiv:2304.07193 (2023)

51. Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., et al.: Pytorch: An imperative style, high-performance deep learning library. NeurIPS (2019)
52. Peebles, W., Zhu, J.Y., Zhang, R., Torralba, A., Efros, A.A., Shechtman, E.: Gan-supervised dense visual alignment. In: CVPR (2022)
53. Peng, Y., Ganesh, A., Wright, J., Xu, W., Ma, Y.: Rasl: Robust alignment by sparse and low-rank decomposition for linearly correlated images. IEEE TPAMI (2012)
54. Rocco, I., Arandjelovic, R., Sivic, J.: Convolutional neural network architecture for geometric matching. In: CVPR (2017)
55. Rocco, I., Arandjelović, R., Sivic, J.: End-to-end weakly-supervised semantic alignment. In: CVPR (2018)
56. Rocco, I., Cimpoi, M., Arandjelović, R., Torii, A., Pajdla, T., Sivic, J.: Ncnet: Neighbourhood consensus networks for estimating image correspondences. IEEE TPAMI pp. 1020–1034 (2020)
57. Ronneberger, O., Fischer, P., Brox, T.: U-net: Convolutional networks for biomedical image segmentation. In: MICCAI. Springer (2015)
58. Saharia, C., Chan, W., Saxena, S., Li, L., Whang, J., Denton, E.L., Ghasemipour, K., Gontijo Lopes, R., Karagol Ayan, B., Salimans, T., et al.: Photorealistic text-to-image diffusion models with deep language understanding. NeurIPS (2022)
59. Schwöbel, P., Warburg, F.R., Jørgensen, M., Madsen, K.H., Hauberg, S.: Probabilistic spatial transformer networks. In: UAI (2022)
60. Seo, P.H., Lee, J., Jung, D., Han, B., Cho, M.: Attentive semantic alignment with offset-aware correlation kernels. In: ECCV (2018)
61. Shokrollahi Yancheshmeh, F., Chen, K., Kamarainen, J.K.: Unsupervised visual alignment with similarity graphs. In: CVPR (2015)
62. Sinclair, M., Schuh, A., Hahn, K., Petersen, K., Bai, Y., Batten, J., Schaap, M., Glocker, B.: Atlas-istn: joint segmentation, registration and atlas construction with image-and-spatial transformer networks. Medical Image Analysis (2022)
63. Skafte Detlefsen, N., Freifeld, O., Hauberg, S.: Deep diffeomorphic transformer networks. In: CVPR (2018)
64. Tang, L., Jia, M., Wang, Q., Phoo, C.P., Hariharan, B.: Emergent correspondence from image diffusion. NeurIPS (2024)
65. Wah, C., Branson, S., Welinder, P., Perona, P., Belongie, S.: The caltech-ucsd birds-200-2011 dataset (2011)
66. Weber, R.S., Eyal, M., Skafte Detlefsen, N., Shriki, O., Freifeld, O.: Diffeomorphic temporal alignment nets. In: NeurIPS (2019)
67. Weber, R.S., Freifeld, O.: Regularization-free diffeomorphic temporal alignment nets. In: ICML. PMLR (2023)
68. Zhang, J., Herrmann, C., Hur, J., Polania Cabrera, L., Jampani, V., Sun, D., Yang, M.H.: A tale of two features: Stable diffusion complements dino for zero-shot semantic correspondence. NeurIPS (2024)
69. Zhang, X., Wang, D., Zhou, Z., Ma, Y.: Robust low-rank tensor recovery with rectification and alignment. IEEE TPAMI (2019)