
FastJAM: a Fast Joint Alignment Model for Images

Supplemental Material

Contents

This document contains the following:

- **§ A: Visual Comparison**
 - Additional qualitative comparisons against existing joint alignment methods (Figure 6).
- **§ B: Additional Visualizations**
 - Additional joint alignment results on the CUB dataset (Figure 7).
 - Full pairwise alignment grids for representative categories (Figure 8 and Figure 9)
- **§ C: Explaining the Colormap Visualization in More Detail**
 - Illustration of FastJAM’s canonical alignment using a fixed RGB colormap (Figure 10).
- **§ D: Runtime Analysis**
 - Scalability runtime assessment for increasing number of images ($N = 10$ to 100) (Figure 11).
- **§ E: Model Configuration and Training Setup**
 - Full architecture summary (Table 5).
 - Optimization settings, matcher configuration, and training procedure.
- **§ F: External Tools and Frameworks**
 - Summary of third-party tools and libraries used, including RoMa, LoFTR, Grounded-SAM, and `torch_geometric`.
- **§ G: Lie Group Parameterization**
 - Full explanation of the Lie group parameterization.

Additionally, key notational conventions used throughout the main paper and this document are summarized in Table 6.

A Additional Visual Comparisons

To qualitatively assess alignment quality, we compare FastJAM against existing joint alignment methods, including SpaceJAM and ASIC. As illustrated in Figure 6, FastJAM produces more coherent and natural-looking alignments, particularly in challenging cases involving pose variation. Unlike ASIC, which applies dense warping and often introduces distortions, FastJAM preserves global structure by relying on sparse keypoints and homographic transformations.

B Additional Visualizations

This section provides supplementary qualitative results that further demonstrate the alignment capabilities of FastJAM across various categories and settings.

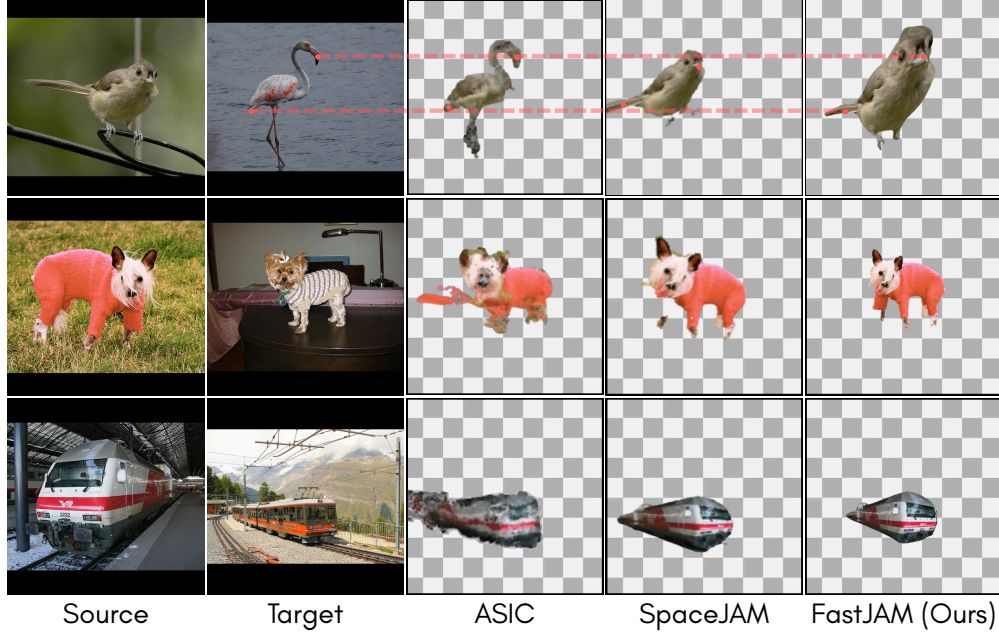


Figure 6: **Qualitative Comparison of Pairwise Alignment Methods.** Visual comparison of joint alignment results across ASIC [1], SpaceJAM [2], and FastJAM (ours) on several image pairs. ASIC, which relies on dense warping fields, often introduces spatial distortions and unrealistic deformations in low-texture or structured regions. In contrast, SpaceJAM and FastJAM apply global homographies, resulting in more coherent transformations. Notably, in the top row (bird), FastJAM produces the most geometrically consistent alignment, as evidenced by the parallel lines and precise correspondence of semantically meaningful points such as the beak and tail.

B.1 Additional Joint Alignment on CUB Dataset

Figure 7 presents further joint alignment results on additional classes from the CUB-200 dataset. For each class, the original input images are shown in the top row, while the bottom row displays their aligned counterparts. The outputs demonstrate FastJAM’s ability to handle fine-grained categories and produce visually coherent canonical views across varying poses and appearances.

B.2 Pairwise Alignment

We visualize full pairwise alignment grids for two representative categories: “aeroplane” and “horse.” As shown in Figure 8 and Figure 9, each grid displays how the source images (rows) are aligned to the target images (columns) using the estimated inverse-compositional warps. The diagonal entries, which correspond to self-alignments (i.e., identity transformations), are highlighted with a purple dashed-line frame. These grids illustrate FastJAM’s ability to produce consistent, symmetric mappings across image pairs and maintain semantic structure throughout the alignment process.

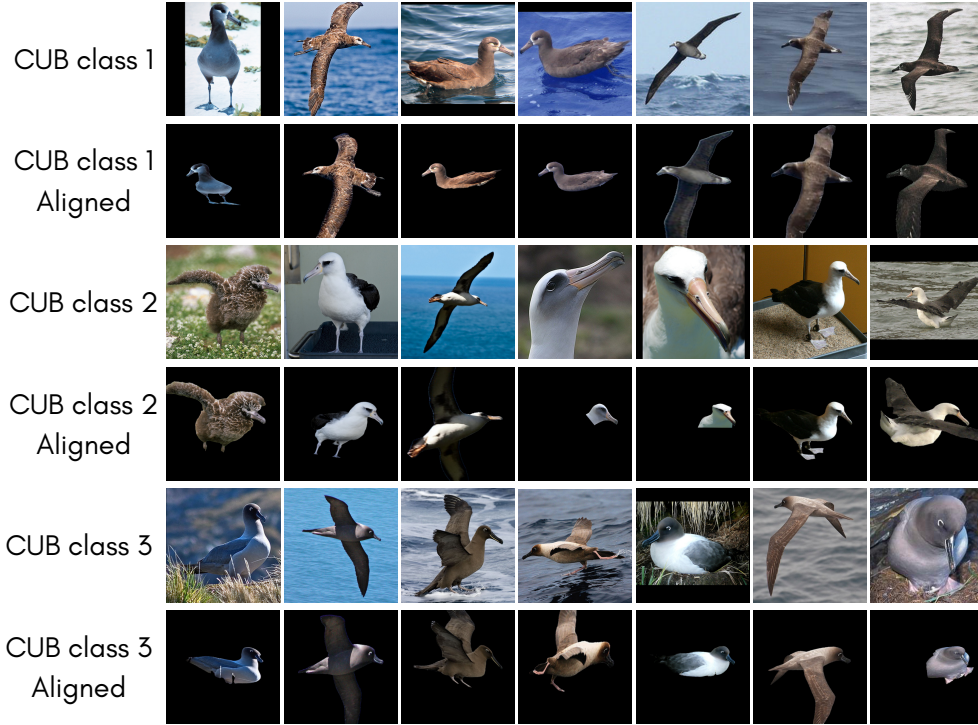


Figure 7: **Qualitative alignment results on CUB-200 classes.** Each pair of rows corresponds to a distinct semantic class from the CUB-200 [3] dataset. The top row in each pair shows the original, unaligned images; the bottom row shows the corresponding aligned images produced by FastJAM. The alignment process successfully maps semantically consistent parts (e.g., heads, wings, tails) to similar spatial locations across different instances within each class.

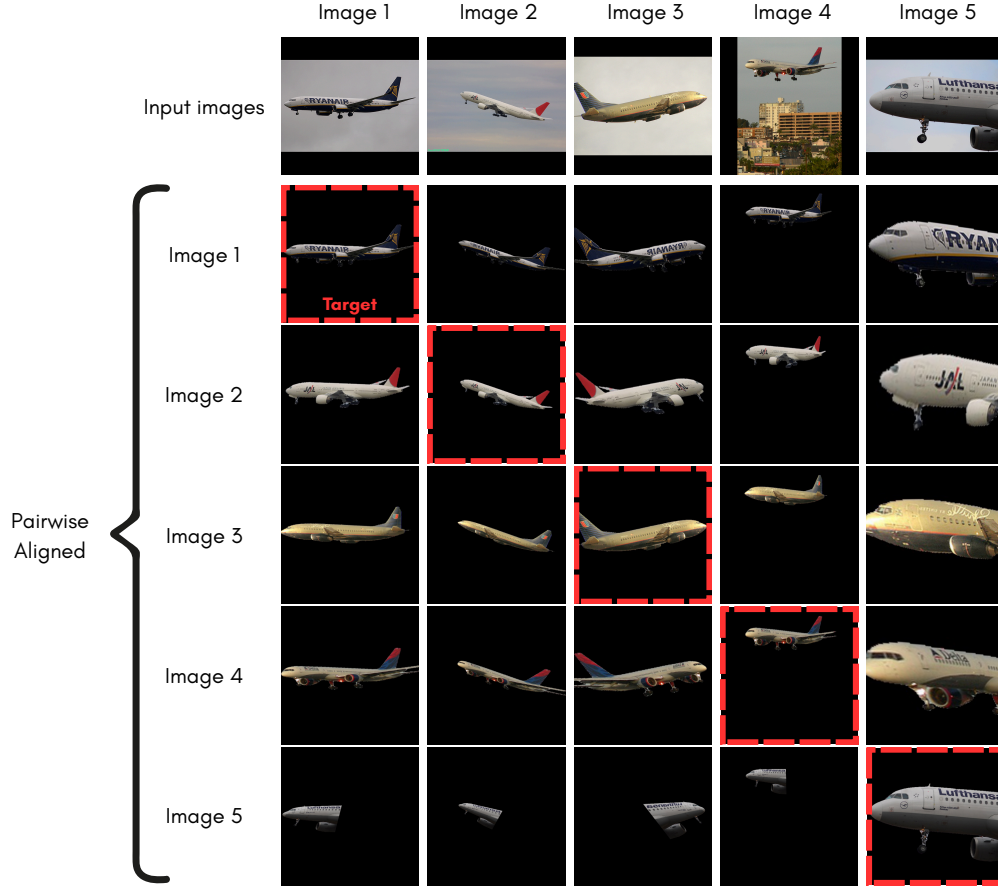


Figure 8: **Pairwise Alignment Grid – ‘Aeroplane’ Class.** Top row: five input images from the ‘aeroplane’ category. Below: a 5×5 alignment matrix, where each entry in column j displays the corresponding source image from row i , warped to align with target image j using the inverse-compositional transformation $T^{\theta_i} \circ T^{-\theta_j}$. Each row thus visualizes the same source image aligned to five different targets. Diagonal entries show the self-warped images (i.e., identity transformation), but with the canonical background rather than the original, and are marked with a **red** dashed-line frame. This layout highlights FastJAM’s ability to achieve coherent, semantically meaningful alignments across all image pairs.

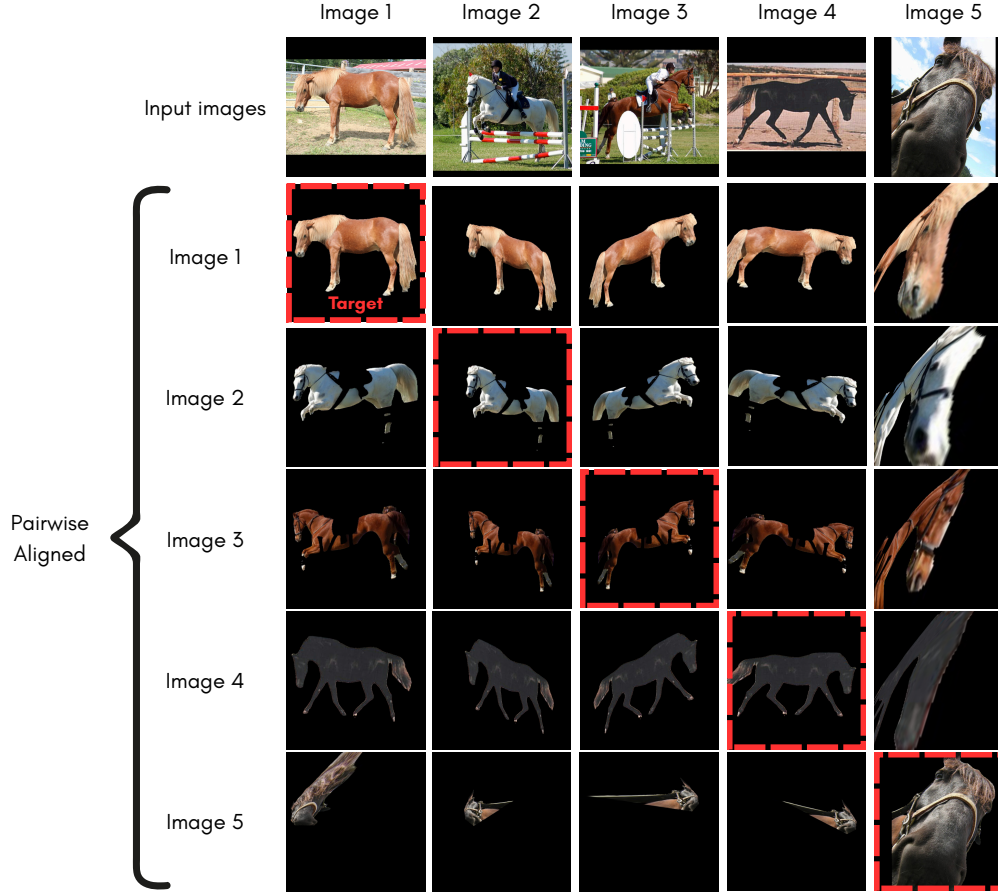


Figure 9: **Pairwise Alignment Grid – ‘Horse’ Class.** Top row: five input images from the ‘horse’ category. Below: a 5×5 grid showing the full pairwise alignment structure. Each image in row i , column j corresponds to the source image i aligned to the target image j using $T^{\theta_i} \circ T^{-\theta_j}$. The diagonal entries depict self-warping (identity), rendered with the canonical background rather than the original, and are marked with a **red** dashed-line frame. This visualization reveals consistent alignment behavior across the set, illustrating how FastJAM handles pose and appearance variation within a semantic class.

C Explaining the Colormap Visualization in More Detail

To provide an intuitive understanding of how FastJAM aligns images to a shared canonical space, we visualize the warped outputs using a predefined RGB colormap. As shown in Figure 10, each input image is first aligned to the canonical frame and blended with the colormap. The result is then inverse-warped back to the original image space, allowing us to visualize how semantic regions are mapped consistently across instances. This process highlights FastJAM’s ability to establish meaningful correspondences without relying on dense features or explicit templates.

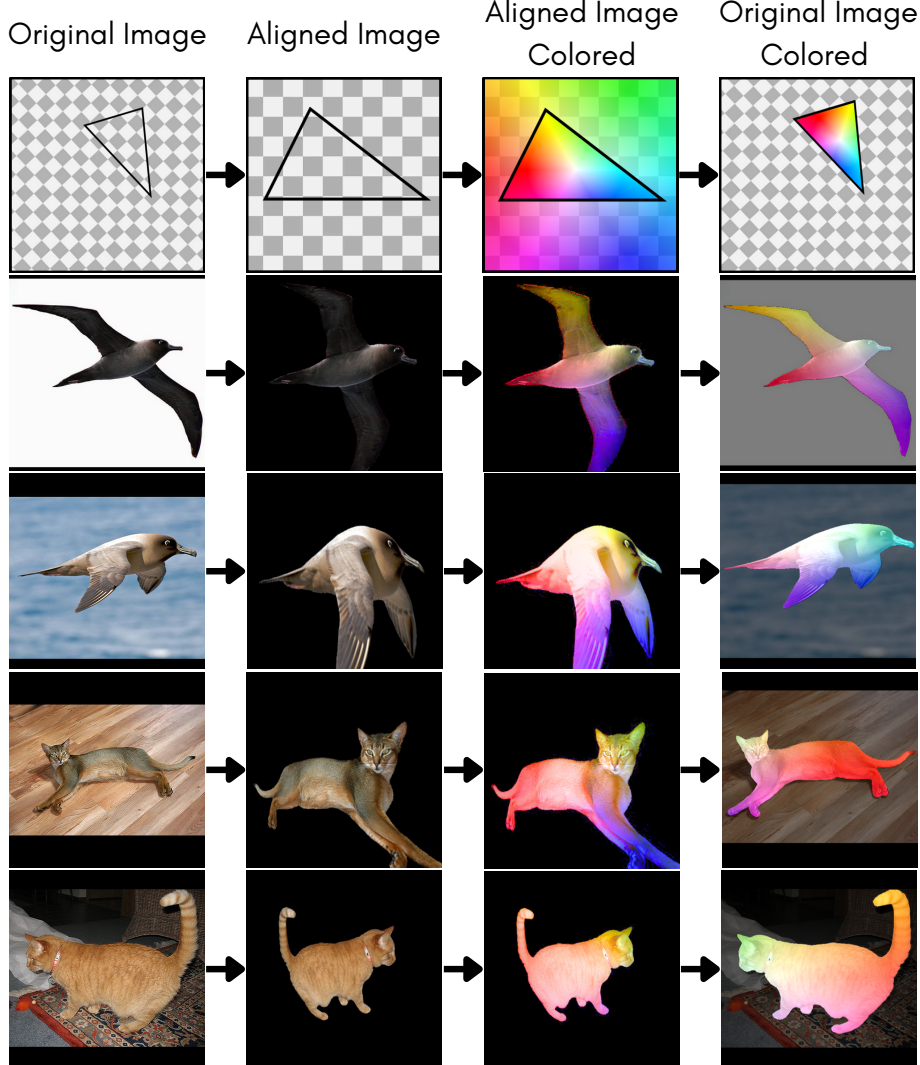


Figure 10: **Illustrated Canonical Space Visualization.** This figure provides an intuitive explanation of how FastJAM visualizes the canonical space \mathcal{C} using a predefined RGB colormap I_{colormap} . For each input image I_i (left column), we first apply the estimated homography to obtain its aligned version in canonical space: $I_i \circ T^{\theta_i}$. We then blend this aligned image with the colormap via averaging: $\frac{1}{2}(I_{\text{colormap}} + (I_i \circ T^{\theta_i}))$. Finally, we apply the inverse warp to visualize the blended canonical signal in the original image frame: $[\frac{1}{2}(I_{\text{colormap}} + (I_i \circ T^{\theta_i}))] \circ T^{-\theta_i}$. This provides a visual explanation of how semantically similar regions across instances are mapped to consistent spatial locations.

D Runtime Analysis

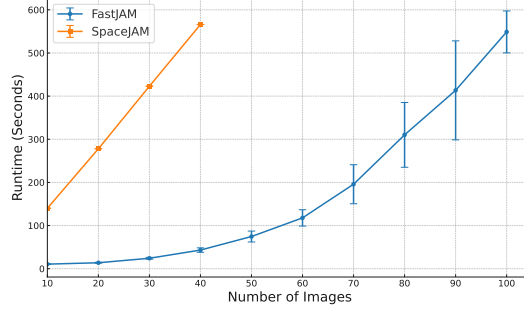


Figure 11: Runtime analysis between SpaceJAM and FastJAM over an increasing number of images. SpaceJAM PCA processing step runs out of RAM after 40 images.

As illustrated in Figure 11, FastJAM remains efficient across all tested sizes, completing alignment in under 450 seconds even for 100 images. In contrast, the current implementation of SpaceJAM runs out of RAM after $N = 40$ images due to the PCA preprocessing step.

E Model Configuration and Training Setup

We train the model for 600 epochs using the Adam optimizer with an initial learning rate of 5×10^{-3} , multiplied by 0.5 after 200 epochs without improvement. The loss function is based on the Geman-McClure formulation with a robustness parameter $\sigma = 0.25$, and no weight decay is applied. The feature extractor within the GNN uses 5 layers of hidden size 128, followed by a linear projection to an 8-dimensional homography parameter vector. A full summary of the model architecture and its 133,256 trainable parameters is provided in Table 5. To encourage geometric stability, the final projection layer is initialized to approximate the identity transformation, and transformations are parameterized using Lie algebra to ensure invertibility. Alignment is applied iteratively using a single pass of the inverse compositional (IC) spatial transformer network. During optimization, horizontal flips are checked every 100 epochs.

Table 5: GraphSAGE GNN Model Summary.

Layer (type:name)	Output Shape	Param
GraphSAGE GNN	–	–
convs.0.lin_l.weight	[128, 2]	256
convs.0.lin_l.bias	[128]	128
convs.0.lin_r.weight	[128, 2]	256
convs.1.lin_l.weight	[128, 128]	16,384
convs.1.lin_l.bias	[128]	128
convs.1.lin_r.weight	[128, 128]	16,384
convs.2.lin_l.weight	[128, 128]	16,384
convs.2.lin_l.bias	[128]	128
convs.2.lin_r.weight	[128, 128]	16,384
convs.3.lin_l.weight	[128, 128]	16,384
convs.3.lin_l.bias	[128]	128
convs.3.lin_r.weight	[128, 128]	16,384
convs.4.lin_l.weight	[128, 128]	16,384
convs.4.lin_l.bias	[128]	128
convs.4.lin_r.weight	[128, 128]	16,384
fc.weight	[8, 128]	1,024
fc.bias	[8]	8
Total	–	133,256

For correspondence estimation, we employ the RoMa matcher at a fixed image resolution of 560×560 for both coarse and upsampled stages. A maximum of 10 keypoints is retained per image, filtered by

non-maximum suppression (NMS) using a radius of 0.054 in normalized coordinates—corresponding to a 30×30 pixel window in the original image space. This ensures spatial coverage while avoiding redundant detections.

Table 6: Summary of Notation

Symbol	Description
$\mathcal{I} = (I_i)_{i=1}^N$	Set of input images
M_i	Number of keypoints in image I_i
$X_i = \{x_i^{(1)}, \dots, x_i^{(M_i)}\}$	Keypoints in image I_i , $x_i^{(m)} \in [-1, 1]^2$
\mathcal{M}_{ij}	Set of matched keypoints between I_i and I_j
\mathcal{T}	Family of parametric transformations (e.g., homographies)
$T^{\theta_i} \in \mathcal{T}$	Transformation for image I_i , parameterized by θ_i
$\mathcal{G} = (\mathcal{V}, \mathcal{E})$	Keypoint graph with intra- and inter-image edges
$f(\mathcal{G})$	GNN-based function that predicts $\{\theta_i\}_{i=1}^N$ from a graph
$I_i \circ T^{\theta_i}$	Image I_i warped by transformation T^{θ_i}

F External Tools and Frameworks

We gratefully acknowledge the use of several open-source libraries and resources in this project. Our GNN implementation is based on `torch_geometric`, primarily using the GraphSAGE [4] architecture, along with other variants for comparison. We used Weights & Biases for experiment tracking and visualization. For keypoint matching, we built upon the official implementation of RoMa [5], and we also incorporated components from the LoFTR framework [6]. Object-centric masks were obtained using Grounded-SAM, which combines Grounding DINO [7, 8] and the Segment Anything Model (SAM) [9]. We thank the authors of all these works for making their code and models publicly available.

G Lie Group Parameterization

A homography has 8 degrees of freedom and corresponds to an equivalence class of invertible matrices, where a representative with unit determinant can be used. Now consider

$$\mathfrak{sl}(3) = \left\{ \Theta = \begin{bmatrix} \theta_1 & \theta_2 & \theta_3 \\ \theta_4 & \theta_5 & \theta_6 \\ \theta_7 & \theta_8 & -(\theta_1 + \theta_5) \end{bmatrix} \right\} \text{ and } \text{SL}(3) = \{ \mathbf{H} \in \mathbb{R}^{3 \times 3} : \det \mathbf{H} = 1 \}. \quad (1)$$

The space $\mathfrak{sl}(3)$ is the Lie algebra of trace-zero matrices. The matrix exponential maps $\mathfrak{sl}(3)$ into $\text{SL}(3)$, yielding a smooth eight-parameter representation of homographies. Our network predicts $(\theta_i)_{i=1}^N$, where each $\theta_i \in \mathbb{R}^8$ is mapped to $\Theta_i \in \mathfrak{sl}(3)$ as shown above, and the homography is obtained via $T^{\theta_i} = \mathbf{H}_i = \exp(\Theta_i)$. This construction guarantees $\det T^{\theta_i} = 1$. In particular, $\theta_i = \mathbf{0}_{8 \times 1}$ yields the identity matrix, and $T^{-\theta_i}$ is the inverse of T^{θ_i} .

References

- [1] Kamal Gupta, Varun Jampani, Carlos Esteves, Abhinav Shrivastava, Ameet Makadia, Noah Snavely, and Abhishek Kar. ASIC: Aligning sparse in-the-wild image collections. In *ICCV*, 2023. [2](#)
- [2] Nir Barel, Ron Shapira Weber, Nir Mualel, Shahaf E Finder, and Oren Freifeld. Spacejam: a lightweight and regularization-free method for fast joint alignment of images. In *European Conference on Computer Vision*, pages 180–197. Springer, 2024. [2](#)
- [3] Catherine Wah, Steve Branson, Peter Welinder, Pietro Perona, and Serge Belongie. The caltech-ucsd birds-200-2011 dataset, 2011. [3](#)
- [4] Will Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. *Advances in neural information processing systems*, 30, 2017. [8](#)
- [5] Johan Edstedt, Qiyu Sun, Georg Bökman, Mårten Wadenbäck, and Michael Felsberg. Roma: Robust dense feature matching. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 19790–19800, 2024. [8](#)
- [6] Jiaming Sun, Zehong Shen, Yuang Wang, Hujun Bao, and Xiaowei Zhou. Loftr: Detector-free local feature matching with transformers. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 8922–8931, 2021. [8](#)
- [7] Tianhe Ren, Shilong Liu, Ailing Zeng, Jing Lin, Kunchang Li, He Cao, Jiayu Chen, Xinyu Huang, Yukang Chen, Feng Yan, Zhaoyang Zeng, Hao Zhang, Feng Li, Jie Yang, Hongyang Li, Qing Jiang, and Lei Zhang. Grounded sam: Assembling open-world models for diverse visual tasks, 2024. [8](#)
- [8] Shilong Liu, Zhaoyang Zeng, Tianhe Ren, Feng Li, Hao Zhang, Jie Yang, Chunyuan Li, Jianwei Yang, Hang Su, Jun Zhu, et al. Grounding dino: Marrying dino with grounded pre-training for open-set object detection. *arXiv preprint arXiv:2303.05499*, 2023. [8](#)
- [9] Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C. Berg, Wan-Yen Lo, Piotr Dollár, and Ross Girshick. Segment anything. *arXiv:2304.02643*, 2023. [8](#)