Bachelor Thesis

An analysis of image alignment methods for image collections with large pose variation

Al-Baraa Saeed Saeed Mohammed

Institute of Computer Science II – Visual Computing
University of Bonn

First Reviewer: Prof. Dr. Florian Bernard
Second Reviewer: Prof. Dr. Thomas Schult

March 14, 2025

Summary

In 2023, the neural congealing framework was introduced by Ofri-Amar et al. (2023), representing a significant advancement beyond conventional image alignment methodologies. Neural congealing not only performs alignment but also semantically co-aligns a set of images by identifying the shared content before to the start of training. The framework employs a pre-trained DINO-ViT model to co-segment the shared content and extract corresponding DINO-ViT features from the inputted images. A spatial transformer network is subsequently utilized to congeal the images, their saliency masks, and DINO-ViT features, while concurrently constructing a joint atlas that encapsulates all co-aligned images. This joint atlas serves as a foundation for downstream tasks such as edit propagation. The framework is equipped with a general configuration and supports additional settings tailored for joint image alignment, such as the inclusion of horizontal flips to optimize alignment precision.

In this thesis, the proposed framework will be tested and evaluated using a diverse range of image sets, with its performance assessed through both quantitative and qualitative metrics. These results will be compared. The selected image sets are designed to test the framework's robustness in addressing various forms of misalignment, ranging from minor 2D plane rotations to significant 3D plane rotations, as well as cases involving multiple objects within a single image. The objective is to investigate the limitations of the framework and explore aspects of its performance that were not addressed in the original research paper.

Furthermore, an alternative approach for initializing the jointly aligned atlas will be evaluated. This approach replies image clustering techniques to partition the original image set into smaller subsets that show higher intra-group similarity compared to the full image set. By reducing misalignment within each subset before the training, this method aims to improve the overall alignment performance of the framework. The effectiveness of this clustering-based initialization will be compared against the standard approach, which initializes the atlas by computing the mean of all key features. This comparison will provide insights into whether clustering can serve as a viable strategy to enhance the robustness and accuracy of the framework in cases with substantial variability in image content.

Acknowledgments

I want to thank whoever helped me write this thesis in a way or another.

Contents

1 Introduction

Image alignment is a fundamental problem in computer vision, focused on minimizing variability within a set of images to map them to a shared reference frame. It is a key step in many downstream tasks, including image registration, object recognition, medical imaging, and even artistic applications such as style transfer. Achieving consistent alignment is critical for tasks that require accurate feature correspondence across images. However, the inherent variability in real-world datasets—such as differences in pose, orientation, scale, lighting, and clutter—makes this a challenging problem. The concept of Congealing was first introduced by Learned-Miller (2006) in 2006 as an algorithm specifically designed to address this challenge. The original Congealing method primarily relied on affine transformations to align images by iteratively minimizing an entropy-based loss function. By reducing inter-image variability, this method significantly increased structural similarity among the images. While this algorithm demonstrated effectiveness in controlled scenarios, such as the alignment of handwritten digits, its applicability was constrained by its simplicity. It struggled to handle datasets characterized by significant semantic variability, non-affine transformations, or cluttered backgrounds.

The motivation for advancing beyond traditional Congealing methods comes from the growing need for robust image alignment techniques that can handle increasingly complex datasets. As modern datasets often include diverse objects, intricate structures, and varying transformations, traditional methods fall short in achieving precise alignment. For example, in medical imaging, slight misalignment can lead to inaccuracies in diagnosis, while in autonomous driving, the failure to align road scenes effectively can impact object detection systems. Similarly, in applications such as digitizing historical documents or aligning archaeological imagery, high variability in features due to wear or damage further complicates the alignment process. These real-world challenges necessitate more sophisticated methods that can leverage the semantic content of images rather than relying solely on low-level transformations.

Recent developments in deep learning and feature extraction have significantly extended the scope of Congealing methods. In 2023, Ofri-Amar et al. (2023) introduced the neural congealing framework, a substantial improvement over traditional techniques. This framework integrates a semantic understanding of image content, enabling it to tackle alignment problems involving

diverse and complex datasets. At its core, the neural congealing framework employs a pre-trained DINO-ViT model to co-segment shared content across a set of images and extract meaningful DINO-ViT features. These features, along with initial saliency masks, serve as the basis for constructing a shared semantic atlas. During training, the framework utilizes a spatial transformer network (STN) to iteratively align images, saliency masks, and extracted features while simultaneously refining the joint atlas. The resulting joint atlas is a comprehensive representation of the aligned image set and is useful for downstream applications such as edit propagation or image synthesis. Additionally, the framework includes flexible settings such as horizontal flipping to account for symmetry, further enhancing its versatility.

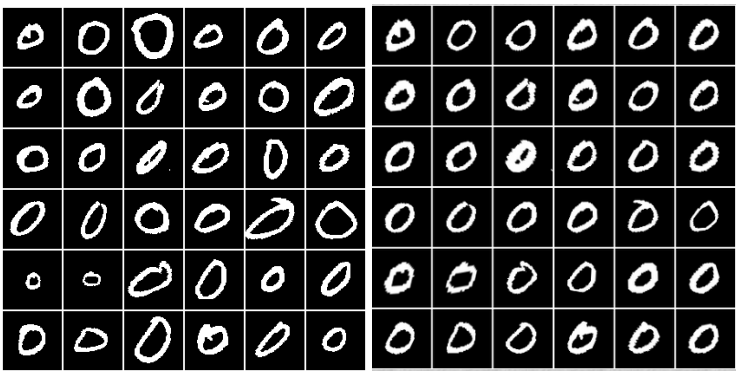
The neural congealing framework offers significant advantages over traditional methods, yet there remain critical gaps that need to be addressed. Its reliance on averaging features for atlas initialization, for instance, may not always yield optimal results. This approach treats all features equally, including noise or irrelevant content, which can hinder alignment accuracy. Furthermore, the framework has primarily been tested on specific datasets, and its robustness across a broader range of image sets is not yet fully understood. Key challenges include alignment under 2D plane rotations, 3D transformations, and scenarios involving multiple objects or significant clutter within individual images.

This thesis is motivated by the need to bridge these gaps and improve the performance and robustness of the neural congealing framework. By conducting rigorous evaluations under diverse and challenging conditions, this work aims to provide a deeper understanding of the framework's strengths and limitations. Specifically, the study will investigate the framework's ability to align datasets with high intra-class variability, rotational misalignments, and complex transformations. Both quantitative metrics, such as similarity scores, and qualitative assessments will be used to measure its performance. The findings will provide valuable insights into the effectiveness of the framework.

In addition to evaluating the existing framework, this thesis introduces and evaluates an alternative strategy for initializing the jointly aligned atlas. The proposed approach utilizes image clustering techniques to partition the original image set into smaller, more homogeneous subsets, thereby reducing internal variability before alignment begins. This clustering-based strategy is expected to mitigate some of the limitations of the standard feature-averaging method, which often struggles with datasets containing significant noise or variability. By aligning these subsets independently and combining their atlases, the framework can potentially achieve higher alignment accuracy and consistency. Through a systematic comparison between the standard and clustering-based initialization methods, this work seeks to answer a critical question: Can clustering improve the initialization process and enhance the overall alignment outcomes? By addressing this question, the thesis aims to contribute to the broader understanding and development of advanced image alignment methods.

2 Related Work

2.1 Congealing



(a) The handwritten zeros dataset before applying the congealing algorithm

(b) The handwritten zeros dataset after applying the congealing algorithm

Figure 2.1: The handwritten zeros dataset, consisting of only two features or colors, as presented before and after congealing. Learned-Miller (2006)

In the work of Learned-Miller (2006), the concept of congealing was introduced as an algorithm designed to align a set of images by iteratively applying a continuous set of transformations. The ultimate goal of this approach was to reduce variability across the image set and align the images to a shared, canonical form, guided by a predefined similarity metric. Specifically, the transformations applied included rigid transformations (such as translation and rotation) and affine transformations (including scaling and shearing).

To demonstrate the effectiveness of the proposed method, Learned-Miller (2006) selected a dataset of handwritten digit images. In this dataset, each image consisted of binary pixel values, with pixels taking on only two states: 0 (black) or 1 (white). The congealing process worked by stacking corresponding pixels from all images at the same spatial location across the dataset. These stacks were evaluated using a cross-entropy loss function, which measured the variability in pixel values within each stack. The algorithm then optimized the transformation parameters to

minimize the entropy across all stacks simultaneously. This iterative process gradually reduced inter-image variability, effectively aligning the images such that corresponding pixels across the dataset exhibited similar features as much as possible (an example of this process is shown in figure 2.1).

By leveraging the entropy-based loss function, the congealing algorithm achieved robust alignment for binary datasets like handwritten digits. However, the method had notable limitations. While effective for relatively simple cases with low semantic complexity and binary pixel features, the algorithm struggled with more complex datasets involving grayscale or color images, higher-dimensional transformations, or semantically diverse content.

2.2 K-Means Clustering

K-means clustering is a popular partitional clustering algorithm that groups unlabeled data into k clusters, aiming to maximize intra-cluster similarity and minimize inter-cluster similarity. Developed by researchers such as Steinhaus, Lloyd, MacQueen, and Jancey, it is widely used for data partitioning. The algorithm iteratively assigns data points to clusters based on proximity to the centroids and updates the centroids until the clusters stabilize. Ikotun et al., 2023

Mathematically, given a dataset $X = \{x_i\}_{i=1}^{n}$ of d-dimensional points, K-means partitions the data into k clusters $C = \{c_j\}_{j=1}^{k}$, where each cluster C_k minimizes the squared error:

$$J(C) = \sum_{k=1}^{K} \sum_{i=1}^{n} \|x_i - c_k\|^2$$

The algorithm starts by randomly selecting k centroids, assigning each point to the closest centroid, and then recalculating the centroids as the mean of the points in each cluster. This process repeats until the cluster assignments no longer change. Ikotun et al., 2023

The basic steps are:

1. Initialize k centroids.
2. Assign each data point to the nearest centroid.
3. Recompute the centroids.
4. Repeat until convergence.

While the algorithm is efficient, the final clusters can depend on the initial centroid positions, so multiple runs with different initializations are often recommended. Ikotun et al., 2023

3 Methodology

This chapter, in addition to explaining the Neural Congealing framework in detail, delves into how the boundaries of the Neural Congealing framework will be tested and explores potential optimizations for atlas initialization through K-clustering. The primary focus is to evaluate the framework's ability to perform under a wide range of conditions by testing it across diverse datasets with varying levels of complexity, scale, and noise. By incorporating datasets with different characteristics—such as images with varying object categories, challenging misalignment, and different degrees of background clutter—the study aims to assess the framework's robustness in handling real-world data variability. This includes testing the framework's performance under severe transformations like rotations, symmetries, and noise, to determine its capacity to effectively align semantically common content pre-processing steps.

In parallel, the chapter investigates the use of K-means clustering as an optimization strategy for atlas initialization. By partitioning features into distinct clusters, K-means clustering helps establish a more structured and context-sensitive starting point for the atlas, potentially improving the alignment process by reducing computational complexity and enhancing the accuracy of the initial alignment. This clustering-based approach will be compared to the primary atlas initialization technique, which is feature averaging, to evaluate whether K-means clustering offers better performance, particularly in handling datasets with diverse or noisy features.

Both the framework testing and the atlas optimization strategies will be analyzed through a combination of qualitative and quantitative metrics. These will include alignment accuracy, and the ability to handle different types of data perturbations. The goal is to provide an evaluation of the Neural Congealing framework's weaknesses, and potential for future improvement. By investigating both the testing scenarios and the impact of a different initialization technique, this chapter aims to further the understanding of how to optimize the framework's performance.

3.1 Background

3.1.1 Neural Congealing

In the work of Ofri-Amar et al. (2023), the Neural Congealing framework was introduced to semantically align different images from various domains without requiring additional input beyond the images themselves. The framework leverages a pre-trained DINO-ViT to identify shared semantic content across the images and to extract features for alignment. The congealing process is achieved using a Spatial Transformer Network (STN), which learns affine transformations for each image. Simultaneously, the framework constructs a joint atlas from the aligned images, designed to be as sparse as possible. (Ofri-Amar et al., 2023)

Architecture

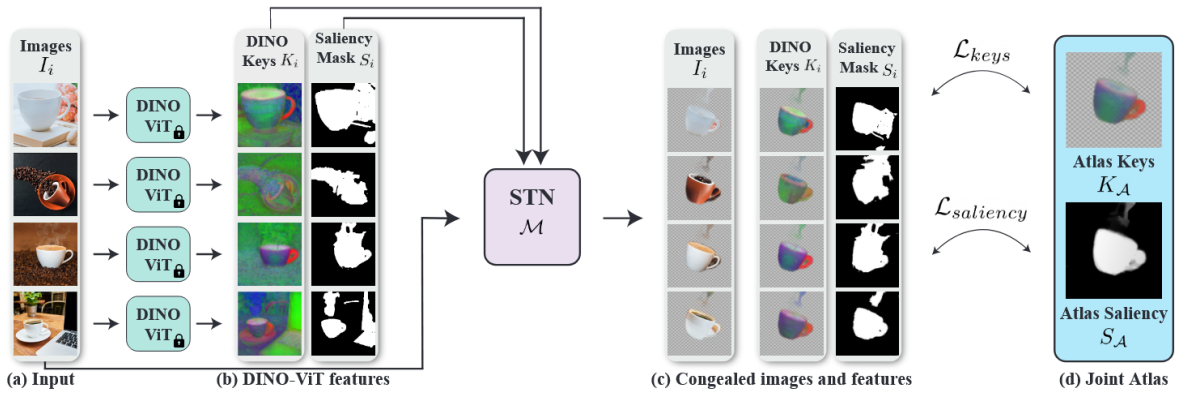


Figure 3.1: Neural Congealing Architecture, adapted from Ofri-Amar et al., 2023.

The Neural Congealing architecture (figure 3.1) is composed of three main components:

Pre-trained DINO-ViT: This component processes the input images to generate co-segmentation maps, referred to as saliency masks, with values ranging from 0 to 1. It also extracts features from the layers of the Visual Transformer (ViT), creating local descriptors for each image.(Ofri-Amar et al., 2023)

Spatial Transformer Network (STN): The DINO-ViT outputs are passed to the STN, which aligns the images and their corresponding DINO-ViT features according to the saliency masks. The STN consists of two sub-networks: one for learning rigid transformations and another for non-rigid transformations. These transformations are combined to generate a sampling grid, which is used to produce the aligned (congealed) images.(Ofri-Amar et al., 2023)

Joint Atlas Learning: Simultaneously, the framework learns a shared joint atlas to which the input images are aligned. This process refines the saliency masks, ensuring that the atlas remains

centered and semantically coherent. (Ofri-Amar et al., 2023)

Joint Image Alignment in Detail

The explanations in this section are grounded in the work of Ofri-Amar et al. (2023). Given a set of input images $I_{i_{i=1}}^{N}$, the method utilizes a pre-trained and fixed DINO-ViT model to extract key features from each image. These features, obtained from the model's final layer (a tunable hyperparameter), are represented as $K_i \in \mathbb{R}^{H \times W \times D}$, where H and W correspond to the image dimensions, and D denotes the embedding dimension of the feature vectors.

Given an input image set $\{I_i\}_{i=1}^{N}$ and a pre-trained (and fixed) DINO-ViT model, the approach extracts key features for each input image from the final layer of the DINO-ViT. Notably, the extraction layer is a hyperparameter that can be adjusted to achieve optimal results. The extracted DINO keys are denoted as

$$K_i \in \mathbb{R}^{H \times W \times D},$$

where H and W correspond to the spatial dimensions of the input images, and D is the embedding dimension of the feature vectors.

Subsequently, these key descriptors K_i are used to compute an initial per-image saliency mask S_i. These masks are generated during the co-segmentation phase as described in **Co-segmentation** and are computed once per image before the training process. These saliency masks serve as the input for the **Spatial Transformer**. However, these masks are preliminary and imprecise, capturing all regions with sufficiently high attention scores when processed by the DINO-ViT.

Ofri-Amar et al. (2023) introduce a learnable DINO-ViT keys atlas, denoted as

$$K_{\mathcal{A}} \in \mathbb{R}^{H_{\mathcal{A}} \times W_{\mathcal{A}} \times D},$$

where $H_{\mathcal{A}}$, $W_{\mathcal{A}}$, and D represent the height, width, and embedding dimension of the atlas, respectively. This atlas is optimized to encapsulate the semantically aligned common content across the input image set. Similarly, a learnable saliency atlas,

$$S_{\mathcal{A}} \in \mathbb{R}^{H_{\mathcal{A}} \times W_{\mathcal{A}}},$$

is defined to capture the aligned saliency information. Where $S_{\mathcal{A}}$ has pixel values ranging between 0 and 1.

To model the mapping of the **Spatial Transformer**, the authors define a 2D mapping $\mathcal{M}$, which, for a given input image I_i and a point $\mathbf{x}_{\mathcal{A}} \in \mathcal{A}$, maps to the corresponding point $\mathbf{x}_i \in I_i$ in the

input image:

$$\mathbf{x}_i = \mathcal{M}(I_i, \mathbf{x}_{\mathcal{A}}) \tag{3.1}$$

To account for both affine and non-affine transformations (i.e., transformations that preserve (affine) or distort (non-affine) the object's shape), the mapping $\mathcal{M}$ is modeled as a composition of two transformations, $\mathcal{M}_r \circ \mathcal{M}_f$, where $\mathcal{M}_r$ represents the rigid transformation and $\mathcal{M}_f$ the non-rigid transformation:

$$\mathcal{M}_r(I_i, x) = \mathbf{sR}x + \mathbf{t} \tag{3.2}$$

Here, $\mathbf{R} \in SO(2)$, $\mathbf{t} \in \mathbb{R}^2$, and $\mathbf{s} \in \mathbb{R}^+$ correspond to 2D rotation, translation, and global scale, respectively.

The non-rigid transformation $\mathcal{M}_f$ is defined as:

$$\mathcal{M}_f(I, x) = x + \mathbf{w} \tag{3.3}$$

where $\mathbf{w}$ is a per-pixel 2D offset.

Both $\mathcal{M}_r$ and $\mathcal{M}_f$ are implemented as **Spatial Transformers**, where the output of the STN implementing $\mathcal{M}_r$ is subsequently fed into the STN implementing $\mathcal{M}_f$. The results are then used to compute the final sampling grid for aligning K_i and S_i.

Objective Functions

The process of learning the joint atlas $\mathcal{A} = (K_{\mathcal{A}}, S_{\mathcal{A}})$ and the mapping $\mathcal{M} = (\mathcal{M}_r \circ \mathcal{M}_f)$ is described. All loss functions and explanations in this section are derived from or based on the work of Ofri-Amar et al. (2023).

The total loss function is composed of the following components:

$$\mathcal{L} = \mathcal{L}_{keys} + \lambda_s \mathcal{L}_{saliency} + \lambda_r \mathcal{L}_{reg_{\mathcal{M}}} + \lambda_a \mathcal{L}_{reg_{\mathcal{A}}} \tag{3.4}$$

In this equation, λ_s, λ_a, and λ_r are fixed weighting factors that can be adjusted during training as hyperparameters to control the relative importance of each loss term.

Semantic Loss $\mathcal{L}$keys. The semantic loss primarily focuses on aligning the keys atlas $K_{\mathcal{A}}$ with each of the congealed images K_i. It is defined as:

$$\begin{aligned} \mathcal{L}_{keys} &= \sum_{i=1}^{N} \frac{1}{N \cdot \Sigma_{S_{\mathcal{A}}}} \sum_{x_{\mathcal{A}}} S_{\mathcal{A}}(x_{\mathcal{A}}) \cdot \mathcal{D}(K_i(x_i), K_{\mathcal{A}}(x_{\mathcal{A}})) \\ \Sigma_{S_{\mathcal{A}}} &= \sum_{x_{\mathcal{A}}} S_{\mathcal{A}}(x_{\mathcal{A}}) \end{aligned} \tag{3.5}$$

Here, $K_{\mathcal{A}}(x_{\mathcal{A}})$ represents the atlas feature at location $x_{\mathcal{A}}$, while $K_i(x_i)$ is the corresponding feature in I_i. The mapping $\mathcal{M}$ ensures that:

$$K_i(x_i) = K_i(\mathcal{M}(I_i, x_{\mathcal{A}})), \tag{3.6}$$

where x_i is derived by mapping the atlas coordinate $x_{\mathcal{A}}$ through $\mathcal{M}$.

The rationale behind this loss is to maximize the similarity between $K_i(x_i)$ and its corresponding $K_{\mathcal{A}}(x_{\mathcal{A}})$, as the atlas encapsulates only the shared semantic information. A strong alignment between the features of K_i and $K_{\mathcal{A}}$ implicitly promotes similarity among all K_i.

During training, the features of K_i remain fixed and are not updated. As a result, minimizing the loss, or the difference between $K_i(x_i)$ and $K_{\mathcal{A}}(x_{\mathcal{A}})$, can only be achieved by modifying the mapping $\mathcal{M}$, which adjusts the relationship between $x_{\mathcal{A}}$ and x_i, or by directly refining the atlas features $K_{\mathcal{A}}(x_{\mathcal{A}})$. This ensures that the alignment and consistency between the congealed images are improved.

Saliency loss $\mathcal{L}_{saliency}$. The saliency loss aims to capture common salient regions across the saliency masks S_i and incorporate them into the saliency atlas $S_{\mathcal{A}}$, following a similar goal as the semantic loss.

$$\mathcal{L}_{saliency} = \frac{1}{N \cdot N_{\mathcal{A}}} \sum_{i=1}^{N} \sum_{x_{\mathcal{A}}} \rho_{0.7}(S_i(x_i), S_{\mathcal{A}}(x_{\mathcal{A}})) \tag{3.7}$$

$$\rho_\delta(a, b) = \begin{cases} \frac{1}{2}(a-b)^2, & \text{if } |a-b| < \delta \\ \delta \cdot \left(|a-b| - \frac{1}{2}\delta\right) & \text{otherwise} \end{cases} \tag{3.8}$$

Here, $N_{\mathcal{A}}$ is the number of pixels in $S_{\mathcal{A}}$, and $\rho_\delta(a, b)$ is the Huber loss. Since the saliency masks S_i are generated during the **co-segmentation** pre-processing step, they may contain noise and irrelevant content. The Huber loss is used to robustly handle these potential discrepancies.

This loss can be interpreted as a "voting loss," where each S_i contributes to identifying regions in the atlas $S_{\mathcal{A}}$ that are considered salient.

STN regularization $\mathcal{L}_{\mathbf{reg}_{\mathcal{M}}}$. his loss term is crucial for regularizing both the rigid and non-rigid mapping networks, preventing them from having excessive freedom in transforming the input images, which could lead to unrealistic, overly distorted images. The regularization loss is composed of several components:

$$\mathcal{L}_{reg_{\mathcal{M}}} = \lambda_{s_1}\mathcal{L}_{scale} + \lambda_{s_2}\mathcal{L}_{mag} + \mathcal{L}_{smooth} \tag{3.9}$$

where λ_{s_1} and λ_{s_2} are hyperparameters that control the relative impact of each component.

$\mathcal{L}_{scale}$ regularizes the rigid mapping $\mathcal{M}_r$ by:

$$\mathcal{L}_{scale} = \frac{1}{N}\sum_{i=1}^{N}|1 - s_i|^2 \tag{3.10}$$

where s_i is the scale parameter output by the rigid STN $\mathcal{M}_r$. Minimizing $\mathcal{L}_{scale}$ encourages the model to keep the scale of the input image close to 1.

$\mathcal{L}_{mag}$ regularizes the non-rigid mapping $\mathcal{M}_f$ by:

$$\mathcal{L}_{mag} = \frac{1}{N \cdot N_{\mathcal{A}}}\sum_{i=1}^{N}\sum_{x_{\mathcal{A}}}\|w_i\|_2^2 \tag{3.11}$$

This helps to minimize the amount of per-pixel 2D offset in the non-rigid transformation.

$\mathcal{L}_{smooth}$ regularizes both the rigid and non-rigid mappings by defining the loss over their shared output:

$$\mathcal{L}_{smooth} = \frac{1}{N \cdot N_{\mathcal{A}}}\sum_{i=1}^{N}\sum_{x_{\mathcal{A}}}\left(\left\|J^TJ\right\|_F + \left\|(J^TJ)^{-1}\right\|_F\right) \tag{3.12}$$

where

$$J = \frac{1}{\Delta}\begin{bmatrix} j_1 & j_2 \end{bmatrix} \in \mathbb{R}^{2\times 2} \tag{3.13}$$

with

$$j_1 = \mathcal{M}\left(I_i, x_{\mathcal{A}} + \Delta \cdot \begin{bmatrix}1\\0\end{bmatrix}\right) - \mathcal{M}(I_i, x_{\mathcal{A}}), \tag{3.14}$$

$$j_2 = \mathcal{M}\left(I_i, x_{\mathcal{A}} + \Delta \cdot \begin{bmatrix}0\\1\end{bmatrix}\right) - \mathcal{M}(I_i, x_{\mathcal{A}}). \tag{3.15}$$

Here, J is the Jacobian matrix of $\mathcal{M}$ at $x_{\mathcal{A}}$. The terms j_1 and j_2 capture the rate of change of $\mathcal{M}$ in the x- and y-directions. Since $\mathcal{M}(I_i, x_{\mathcal{A}})$ maps to a point $x_i \in I_i$, large gradients indicate greater distortion in the mapping $\mathcal{M}$ for the congealed K_i and S_i.

The parameter Δ controls the locality for the gradient calculation: $\Delta = 1$ uses the local neighborhood of atlas pixels, while a larger Δ allows for analyzing distortion over a wider patch of the atlas.

The loss is split into two components, $\mathcal{L}_{smooth_{local}}$ and $\mathcal{L}_{smooth_{global}}$, corresponding to $\Delta = 1$ and $\Delta = 20$, respectively.

Minimizing the Frobenius norm of the matrix $J^T J$ encourages reducing gradient values and consequently minimizing distortion in $\mathcal{M}$ as a whole.

Atlas regularization $\mathcal{L}_{\mathbf{reg}_\mathcal{A}}$. This loss controls the centering and sparsity of $K_\mathcal{A}$ and $S_\mathcal{A}$:

$$\mathcal{L}_{reg_\mathcal{A}} = \mathcal{L}_{center} + \lambda_p \mathcal{L}_{sparsity} \tag{3.16}$$

where λ_p is the weighting parameter for the sparsity.
Without this loss, the atlas would lack control over its distribution, potentially causing the relevant content to drift to arbitrary positions.
The loss $\mathcal{L}_{center}$ constrains the shared content to the center of the atlas:

$$\mathcal{L}_{center} = \left\| \frac{1}{\Sigma_{S_\mathcal{A}}} \cdot \sum_{x_\mathcal{A}} S_\mathcal{A}(x_\mathcal{A}) \cdot x_\mathcal{A} \right\|_2^2 \tag{3.17}$$

where $x_\mathcal{A}$ is normalized such that $x_\mathcal{A} \in [-1, 1]^2$.
Note that, due to the use of $S_\mathcal{A}(x_\mathcal{A})$ as a weighting factor in $\mathcal{L}_{keys}$ (see Eq. 3.5), a separate centering loss for $K_\mathcal{A}(x_\mathcal{A})$ is not needed.

The sparsity loss, $\mathcal{L}_{sparsity}$, ensures that the atlases only contain the most common content across the input images. Without it, the atlases could capture irrelevant content that isn't necessary for congealing the images:

$$\mathcal{L}_{sparsity} = \mathcal{L}^{S_\mathcal{A}}_{sparsity} + \lambda_k \mathcal{L}^{K_\mathcal{A}}_{sparsity} \tag{3.18}$$

The sparsity for $S_\mathcal{A}$ is regularized using L1 regularization along with Ψ_0, an approximation of L0 regularization, to penalize non-zero elements:

$$\mathcal{L}^{S_\mathcal{A}}_{sparsity} = \gamma \|S_\mathcal{A}\|_1 + \Psi_0(S_\mathcal{A}) \tag{3.19}$$

For $K_\mathcal{A}$, the sparsity is enforced on the non-salient parts (i.e., regions not containing common semantic content) by applying L1 regularization:

$$\mathcal{L}^{K_\mathcal{A}}_{sparsity} = (1 - S_\mathcal{A}) \cdot \|K_\mathcal{A}\|_1 \tag{3.20}$$

3.1.2 The DINO Architecture

The DINO framework, introduced by Caron et al. (2021), is a self-supervised learning approach that combines self-distillation with Vision Transformers (ViTs). Unlike traditional methods that rely on a fixed teacher model, DINO uses two identical models—student and teacher—that learn simultaneously. This self-supervised method does not require labeled data but instead uses augmentations of input images (e.g., crops and patches) to teach both models about invariant features in the data.

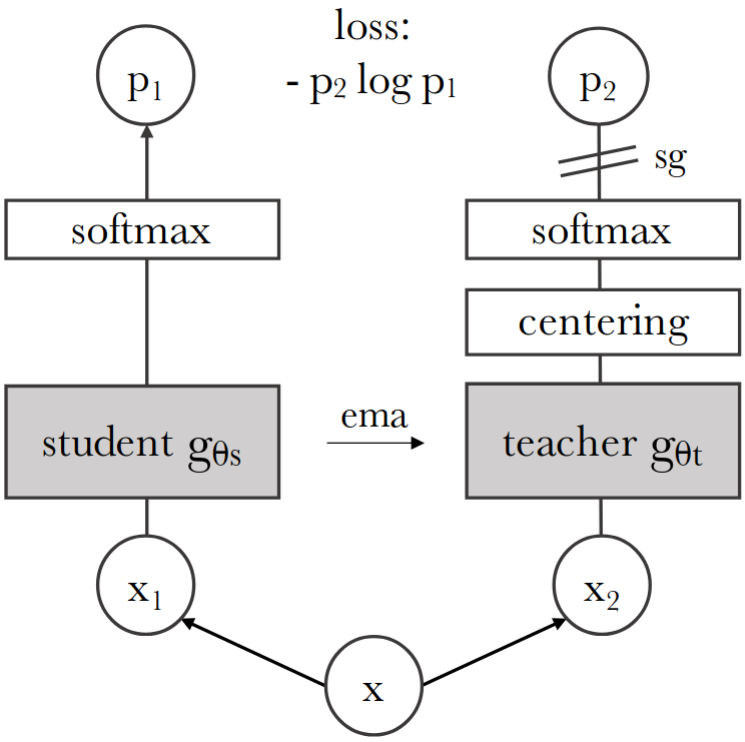


Figure 3.2: DINO architecture. Taken from Caron et al. (2021)

At the beginning, the input images x are augmented. DINO follows the work of Zhang et al. (2019) and applies color jittering, Gaussian blur, and solarization (i.e., inverting all pixel values above a threshold). This process generates a set of views for each image, including both global and local modes. Global modes are crops capturing a wider portion of the object, while local modes are smaller crops from these global ones. These crops x_1 and x_2 are passed to the student and teacher networks, respectively, with the student receiving the local crops and the teacher receiving the global ones.

The outputs are then fed into a softmax layer to obtain a distribution. The network is trained to

align both the teacher and student distributions by applying the cross-entropy loss:

$$\mathcal{L}(x) = -\sum_{x} p_1(x) \log(p_2(x)).$$

The intuition behind this is that if both networks can align their predictions, even though the student only sees small crops of a more global view, the network has learned robust representations of the object.

Notably, the flow of gradients to the teacher network is blocked. Instead, the teacher network parameters are updated via an exponential moving average (EMA) of the student's parameters, following the update rule:

$$\theta_t \leftarrow \lambda\theta_t + (1 - \lambda)\theta_s,$$

with λ following a cosine schedule from 0.996 to 1 during training Caron et al., 2021.

Additionally, the outputs of the teacher network are centered before being passed to the softmax layer. This centering layer maintains a moving average of the ViT's output and subtracts it from the logits entering the softmax. This is necessary to prevent collapse, where the network might learn trivial or degenerate solutions (e.g., a uniform distribution) that fail to capture meaningful information.

3.1.3 Co-segmentation

Co-Segmentation aims to jointly segment common content among all images within a given set. The pre-trained DINO-ViT divides the images into non-overlapping batches and generates local batch descriptors from the calculated DINO-Vit features for each batch. To calculate DINO-Vit features, each image is tokenized after being split into non-overlapping batches, with a [CLS] token assigned. These batches are then embedded and passed through Transformer layers. At each layer, the batches are associated with various features, including queries, keys, values, and tokens. The DINO-ViT features extracted from each layer are then grouped to form the local descriptors. These extracted features vary depending on the layers from which they are extracted, with features from lower layers focusing more on positional information, while those from higher layers emphasize semantic features. The Co-Segmentation process involves two phases: clustering to detect shared content and voting to determine which shared content is salient. Initially, the saliency masks of each image and their local batch descriptors are obtained. These local descriptors are then clustered using the K-means clustering method. Subsequently, voting is conducted by setting a threshold and calculating the saliency mean of all batches belonging to one image within a cluster. If the sum of these means exceeds the threshold, the shared content is considered salient. Finally, a binary segmentation algorithm called Grabcut is applied to refine the binary co-segmentation masks identified as salient. (Caron et al., 2021)

3.1.4 Spatial Transformer Network

The Spatial Transformer Network (STN) is a fully differentiable layer that does not require any supervised training beforehand. It is utilized to register images by taking the feature map of the images and learning a transformation that minimises the difference between the images according to a set of similarity metrics. The STN architecture consists of three main parts: the localization network, grid generator, and sampler. The localization network functions as a regressor, taking the feature map as input and outputting the parameters of the transformation it optimises to minimise the difference between the transformed moving image and the fixed image. The learned transformations are then passed to the grid sampler, which generates the grid of the transformed moving image by inverting the transformations to determine the mapping of pixels from the transformed moving image to the original moving image. Once the grid is generated, it is passed to the sampler, which samples the features of every pixel on the grid by bilinearly interpolating the corresponding pixel on the moving image. (Chen et al., 2023)

The STN Implementation in Neural Congealing

In the framework, both STNs utilize a ResNet backbone. The rigid STN, known as the STN_{sim}, outputs four logits (o_1, o_2,o_3,o_4) (where a logit is denoated as $logit(p) = log(p/1-p)$) that are used to obtain the rigid transformations as shown in equation (3.21), equation (3.22) and equation (3.23).

$$\theta = \pi \tanh(o_1)\ ,\ R = \begin{pmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{pmatrix} \tag{3.21}$$

$$s = \exp(o_2) \tag{3.22}$$

$$t = \begin{pmatrix} o_3 \\ o_4 \end{pmatrix} \tag{3.23}$$

The non-rigid STN, referred to as STN_{flow}, outputs a 16x16 feature grid, which is then converted to a 16x16 coarse flow field and subsequently upsampled by a factor of 8. Finally, the rigid transformations are applied to the 128x128 flow field to form the sampling grid. (Ofri-Amar et al., 2023)

3.2 Testing The Boundaries

This section details the approach taken to test the Neural Congealing framework's ability to handle various forms of visual misalignment, ranging from simple rotations to more complex types of noise and clutter. The framework is designed to align semantically common content across

diverse input images, even under challenging conditions. The testing will focus on evaluating how well Neural Congealing manages alignment when subjected to multiple types of transformations and disruptions. Specifically, it will be tested on its ability to address rotational misalignment in both 2D and 3D data, requiring it to align images and objects that may appear at different angles and orientations.

Beyond rotation, the framework will be challenged with datasets containing complex forms of visual noise, such as images with multiple objects that vary in size, or position. These scenarios introduce significant misalignment, where the framework must effectively distinguish and align relevant features within cluttered or noisy visual contexts. In particular, Neural Congealing must identify semantically meaningful content despite background clutter or the presence of multiple distracting objects, as the framework is designed to focus on shared content under severe variations in appearance, pose, and background conditions. Testing will evaluate the framework's performance in these conditions through both qualitative and quantitative metrics. These results will provide insights into Neural Congealing's effectiveness, helping to identify its limitations.

3.3 Atlas Initialization

In the Neural Congealing framework, atlas initialization plays a vital role in aligning semantically common content across diverse input images. The primary method for initializing the atlas is feature averaging, while an alternative method feature labeling through K-means clustering is tested and compared to the primary approach. Each method offers a distinct strategy for organizing and refining the atlas to enhance alignment.

The feature averaging method serves as the framework's primary atlas initialization approach. It initializes the atlas by calculating the mean of all features extracted from the input image set. This process creates a basic 2D grid that captures the central tendency of the features derived from the pre-trained DINO-ViT model. The feature averaging method provides a quick and straightforward way to generate an initial atlas. However, it may struggle with datasets that exhibit significant variations in appearance, pose, or background clutter, as averaging can obscure important details. Despite these limitations, feature averaging remains the baseline method for testing the framework's alignment performance without introducing additional complexity, as noted by Neural Congealing's developers Ofri-Amar et al., 2023.

In contrast, the alternative method feature labeling through K-means clustering is tested as a potential improvement over feature averaging. This method involves partitioning the features extracted from the input images into distinct groups using K-means clustering as a pre-processing step. Each cluster of features is treated as a separate subset, and a joint atlas is generated for each cluster. By clustering features into groups based on their similarity, K-clustering helps the framework to initialize more refined and context-sensitive atlases.

3.4 Qualitative and Quantitative Results

Both qualitative and quantitative results are essential for evaluating the performance of the framework and understanding its limitations. Qualitative results provide visual comparisons between datasets before and after alignment, highlighting the visual impact of the process. On the other hand, quantitative results rely on specific metrics to measure the similarity.

In the original work of Ofri-Amar et al. (2023), the PCK metric was employed. However, this metric relies on labeled images, which is not feasible for all datasets used in testing. Many datasets lacked the necessary labels, making it impossible to apply the PCK metric universally. To address this, an alternative metric, the **Global Cosine Similarity Metric**, was used.

Global Cosine Similarity Metric evaluates the alignment between the datasets before and after the alignment by calculating the mean Given a set of feature vectors $K_1(x_1), K_2(x_2), \ldots, K_N(x_N) \in \mathbb{R}^d$, the pairwise cosine similarity is computed using:

$$\mathcal{D}(K_i(x_i), K_j(x_j)) = \frac{K_i(x_i) \cdot K_j(x_j)}{\|K_i(x_i)\| \|K_j(x_j)\|} \tag{3.24}$$

Let $\mathbf{C}$ be the cosine similarity matrix for all features, where:

$$\mathbf{C}[i, j] = \mathcal{D}(K_i(x_i), K_j(x_j)) \quad \forall i, j \in \{1, 2, \ldots, N\}, i \neq j \tag{3.25}$$

To compute the global cosine similarity score, we take the mean of the upper triangular matrix (excluding the diagonal) of $\mathbf{C}$:

$$\mathcal{L}_{global} = \frac{1}{|U|} \sum_{(i,j) \in U} \mathbf{C}[i, j] \tag{3.26}$$

where U represents the indices of the upper triangular elements of $\mathbf{C}$ (i.e., $i < j$).

This score is computed both for the features before and after alignment to quantify the improvement.

4 Experiments

4.1 Testing The Boundaries

This section presents results aimed at identifying the failure cases of the Neural Congealing framework on datasets with challenging misalignment. These include datasets where the same object is subjected to 2D rotations across all angles and 3D rotations introducing out-of-plane variability. Additionally, the evaluation explores how the framework struggles with datasets containing multiple objects in varying poses, and its ability to align multiple objects within a single image.

4.1.1 2D Rotation

This section analyzes the performance of the Neural Congealing framework on four distinct datasets, each depicted in Figure 4.1. These datasets, each containing a single object rotated by multiples of 5 degrees (e.g., 20° or 30°). Each dataset consists of 12 images, showing the object at different rotation angles.

The analysis first presents qualitative results of two of the four datasets, visually demonstrating the framework's ability to align objects across varying degrees of rotational misalignment. This is followed by quantitative results of all the four datasets, which provide a numerical evaluation of the framework's performance across the datasets. The results highlight scenarios where the framework successfully aligns all images to a single pose, aligns them to multiple poses, or fails completely to jointly align even two images.

Bicycle

The qualitative results on the bicycle dataset highlight the framework's effectiveness across varying degrees of rotational misalignment, ranging from 120 degrees to 360 degrees.

figure 4.2 shows For misalignment of 120 degrees or less, the framework can correctly jointly align the dataset to a single canonical pose.

(a) Bicycle (b) Wood Chair (c) Cat Face (d) SUV Car

Figure 4.1: Examples of the datasets used for testing the Neural Congealing framework. Each image represents a different object category.

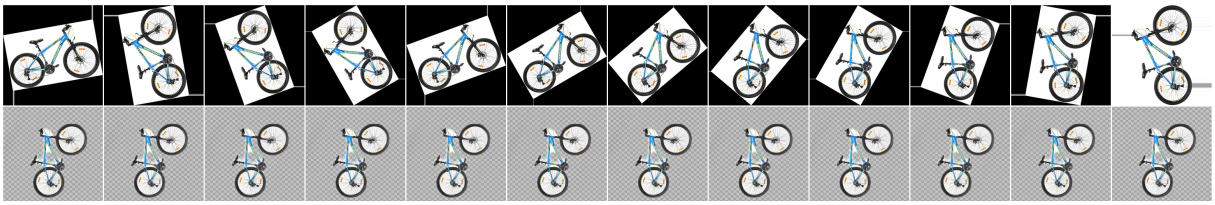


Figure 4.2: The bicycle dataset with 120 degrees misalignment was correctly jointly aligned

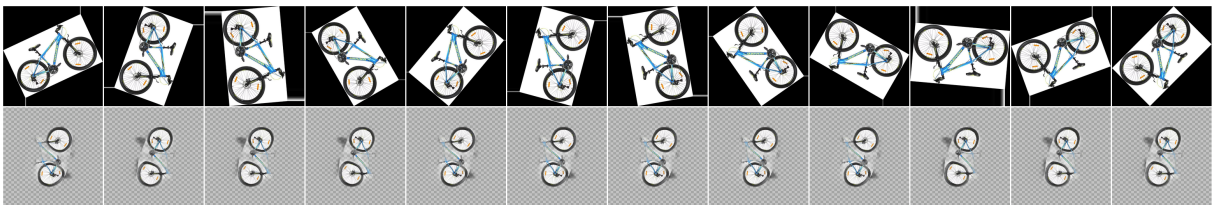


Figure 4.3: The bicycle dataset with 300 degrees is only aligned to multiple canonical poses and not to one canonical pose.

figure 4.3 shows that At 300 degrees, the framework can only align the dataset to multiple poses rather than a single pose. figure 4.4 illustrates that with 360 degrees or more of rotational

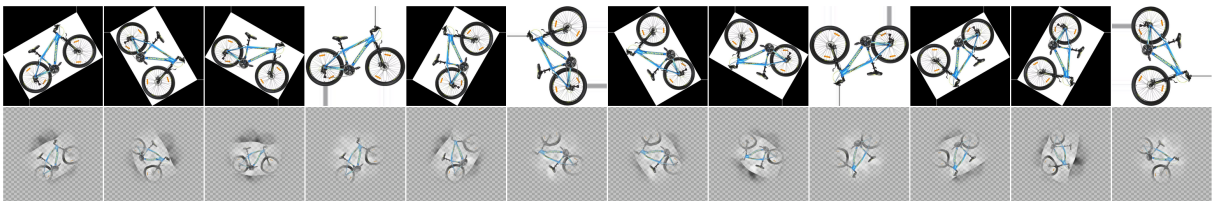


Figure 4.4: The bicycle dataset with 360 degrees can't be aligned to even multiple canonical poses anymore.

misalignment, the framework is unable to align the dataset into even multiple canonical poses.

Wood Chair

The qualitative results on the bicycle dataset highlight the framework's effectiveness across varying degrees of rotational misalignment, ranging from 120 degrees to 360 degrees. These results should follow up on the qualitative results of the bicycle dataset. figure 4.5 shows for

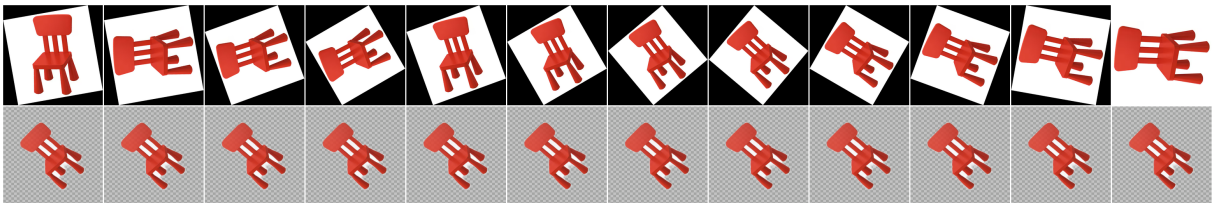


Figure 4.5: The wood chair dataset with 120 degrees misalignment was correctly jointly aligned

misalignment of 120 degrees or less, the framework can correctly jointly align the dataset to a single pose. figure 4.6 shows that At 300 degrees, the framework can only align the dataset to

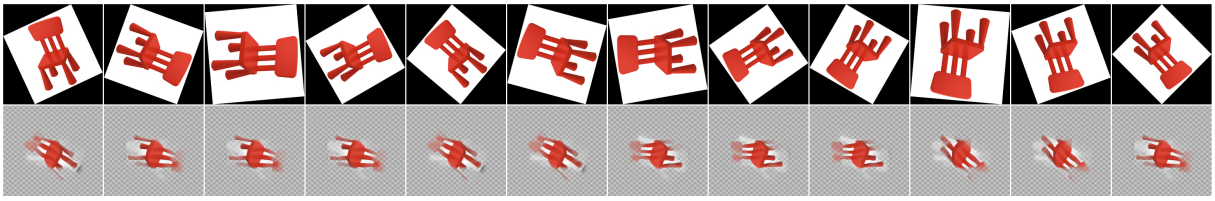


Figure 4.6: The wood chair dataset with 300 degrees is only aligned to multiple canonical poses and not to one canonical pose.

multiple poses rather than a single pose. which is a similar result to figure 4.3.

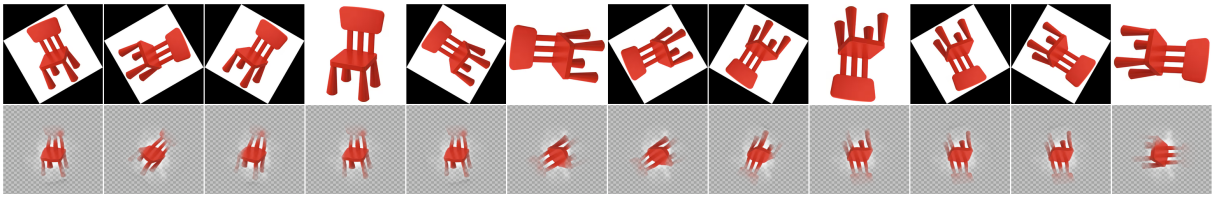


Figure 4.7: The bicycle dataset with 360 degrees can't be aligned to even multiple canonical poses anymore.

figure 4.7 shows that With 360 degrees or more, the framework fails to align the dataset even to multiple canonical poses, indicating its limitations with misalignment of 360 degrees or more.

Quantitative Results

The quantitative results for all four datasets are presented in four plots that show the similarity between the datasets before and after alignment. These results are illustrated in figure 4.8.

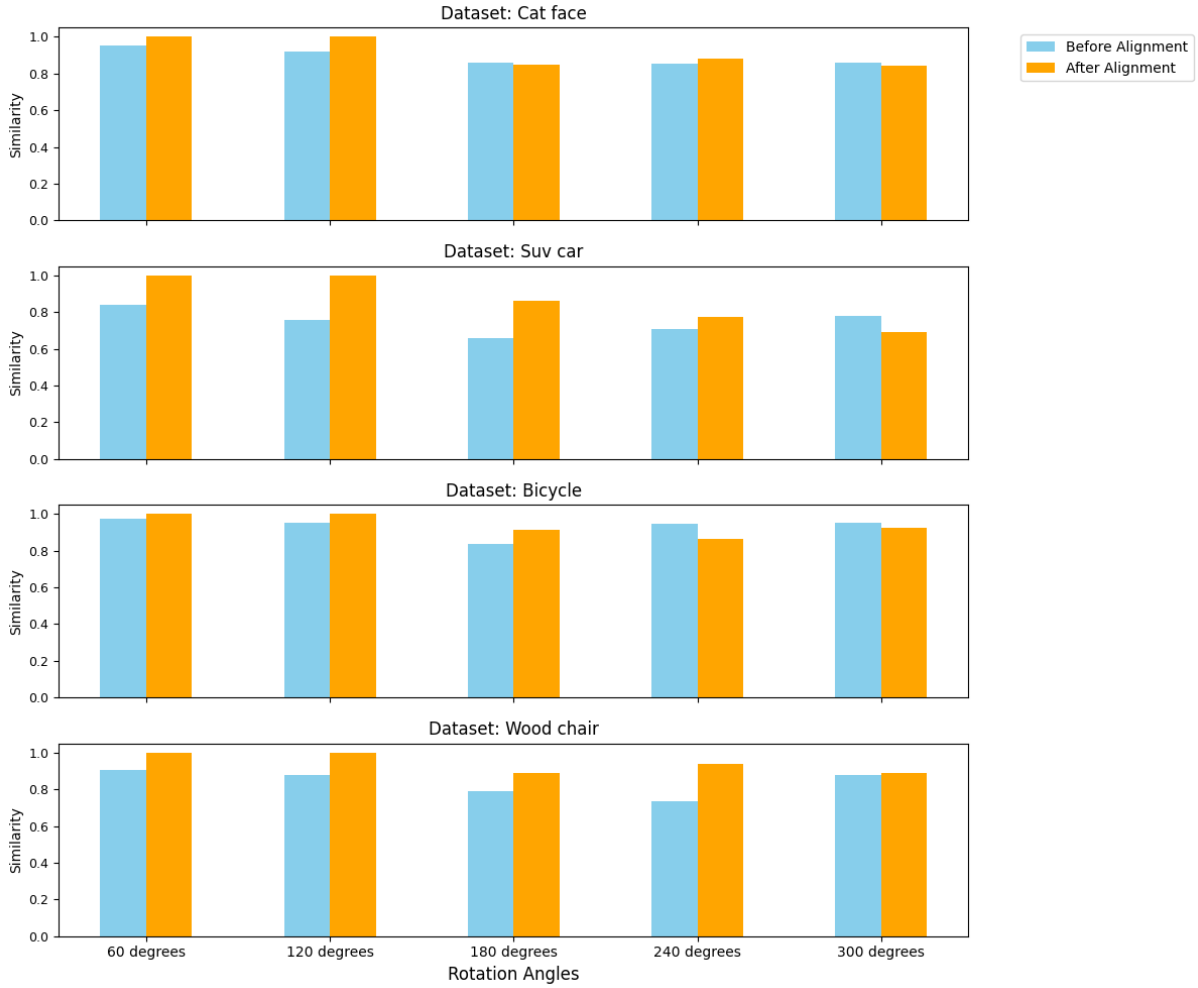


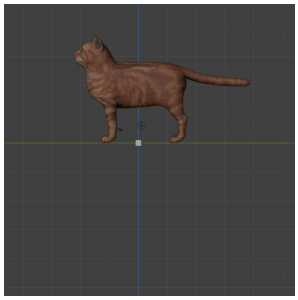
Figure 4.8: The similarity between all images of a dataset before and after the framework jointly aligns the dataset.

The quantitative results demonstrate that the framework is capable of jointly aligning datasets with rotational misalignment up to 120 degrees. However, when the rotational misalignment exceeds 180 degrees, the framework not only fails to jointly align the dataset but also produces an atlas with similarity that are sometimes lower than those of the original dataset before alignment.

4.1.2 3D Rotation

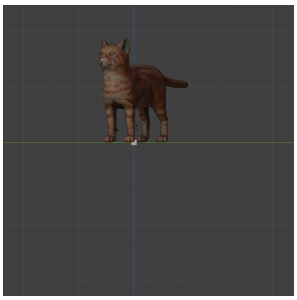
This section evaluates the Neural Congealing framework on a dataset comprising 12 images of a single cat, each rotated along the Y and Z axes at varying angles. The results reveal the framework's difficulty in handling significant rotations along these axes, demonstrating its limitations in addressing 3D rotational misalignment. These findings highlight a critical shortcoming in the framework's ability to effectively align images with 3D rotational misalignment. In summary, the framework struggles significantly with 2D rotational misalignment exceeding

180 degrees, where flipping an image becomes necessary for accurate joint alignment.



(a) Original cat mesh unrotated

(b) The cat mesh is rotated along the y axis



(c) The cat mesh is rotated along the z axis

Figure 4.9: The cat mesh rotated along the y and z axis

figure 4.9 illustrates the cat before rotation and after being rotated along both the Y and Z axes.

Y Axis

The evaluation demonstrates the framework's ability in handling rotational misalignment along the Y axis. figure 4.10 shows With even a 36-degree misalignment, no meaningful alignment is

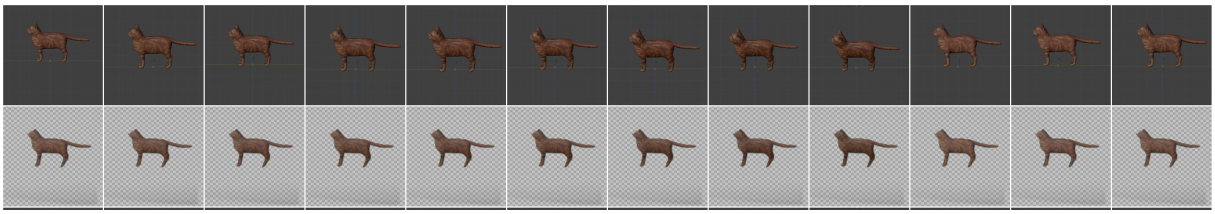


Figure 4.10: With 36 degrees misalignment on the y axis no alignment is really carried just cropping

achieved; instead, the images are merely cropped. figure 4.11 shows when the misalignment

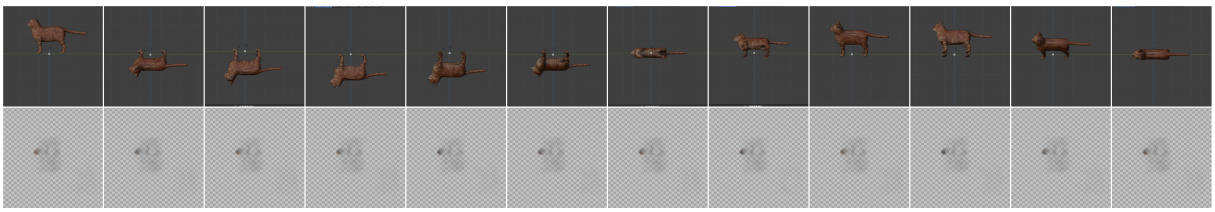


Figure 4.11: With 360 degrees misalignment on the y axis the whole shared content is removed

increases to 360 degrees or more, the entire shared content is removed, exemplifying for the first time a complete failure to align the dataset.

Z Axis

The results reveals significant challenges in handling misalignment along the Z axis. fig-

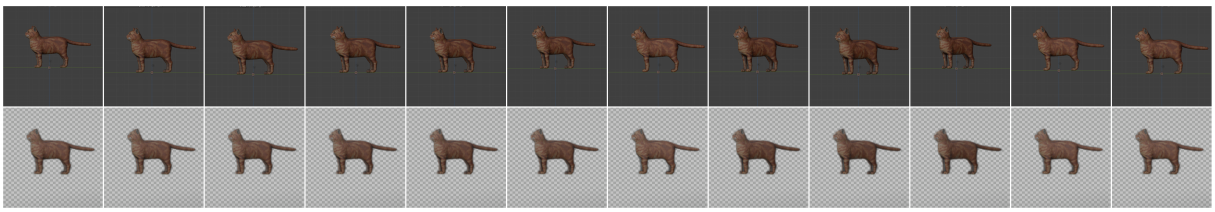


Figure 4.12: With 36 degrees misalignment on the z axis no alignment is really carried just cropping

ure 4.12shows With a 36-degree misalignment, the framework performs no meaningful alignment and instead resorts to cropping.

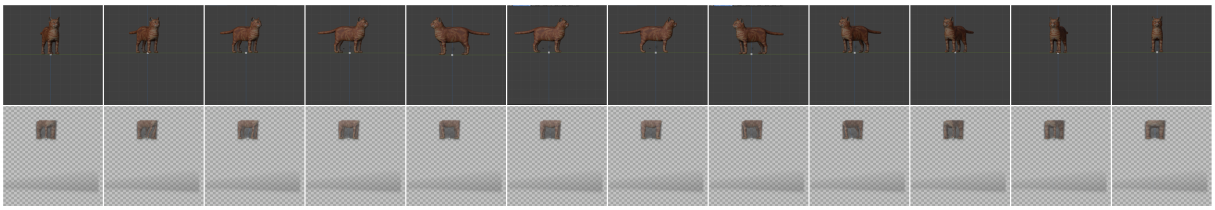


Figure 4.13: With 360 degrees misalignment on the z misalignment the whole upper body is removed and even then the lower body isn't aligned

figure 4.13 shows At 360 degrees of misalignment, the upper body is completely removed, and even the remaining lower body fails to align properly. These results highlight again the framework's limitations in managing rotational misalignment along 3D axis.

4.1.3 Multiple Poses

The Neural Congealing framework demonstrates limitations in jointly aligning datasets containing images of kangaroos and pigeons with varied poses.

figure 4.14 shows that For kangaroos, the framework fails to align images where some are standing and others are jumping, already indicating difficulty in handling objects of the class kind but with pose variation.

Similarly, figure 4.15 highlights the framework's difficulty in aligning images of pigeons that share identical poses but face opposite directions, such as some facing right while others face left. These challenges emphasize the limitations of the framework in achieving joint alignment when objects exhibit opposing orientations.

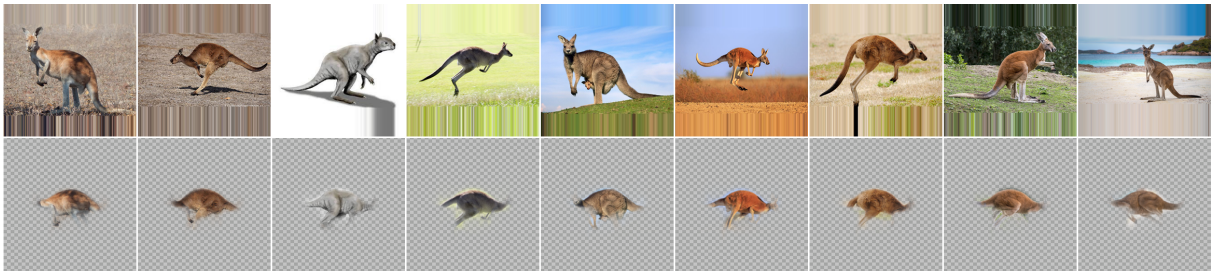


Figure 4.14: Failed attempt to jointly align different images of kangaroos with different poses, where some are standing and some are jumping

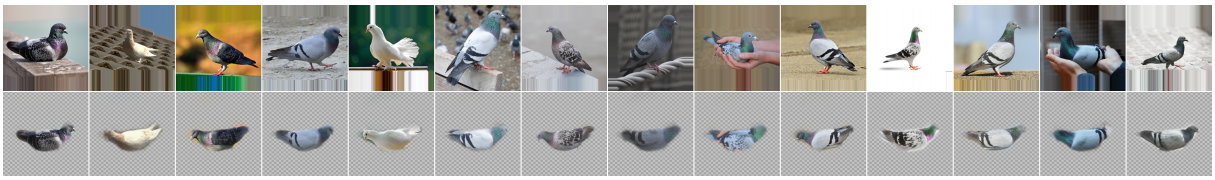


Figure 4.15: Failed attempt to jointly align different images of pigeons with slightly different poses, where some are pointing to the right and some to the left

4.1.4 Multiple Objects

The framework demonstrates mixed performance when aligning images with multiple objects.

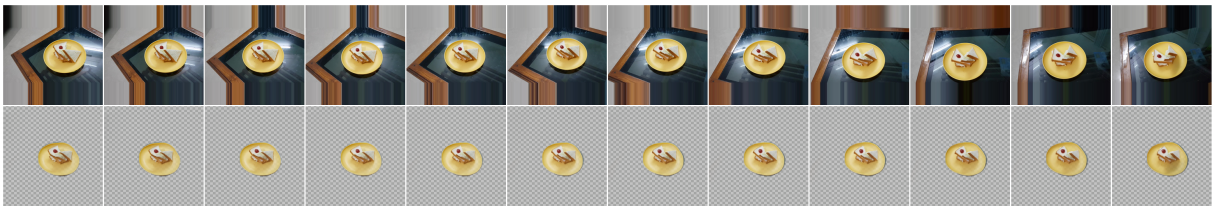


Figure 4.16: Correctly jointly aligning a couple images, where each image includes two sandwiches and each image is rotated slightly

figure 4.16 shows that it successfully aligns a set of images, each containing two sandwiches with slight rotations, But as the rotation degree increases as in figure 4.17, alignment fails.

Furthermore, figure 4.18 demonstrates the framework's inability to detect and align shared content in images featuring multiple geese. This limitation highlights challenges not only in aligning shared features of images with multiple objects but also in reliably detecting at least one object per image.

4.2 Atlas Initialization

In Neural Congealing, the initialization of the joint atlases is achieved by averaging DINO keys and initial saliency masks. This process allows the framework to construct a shared semantic

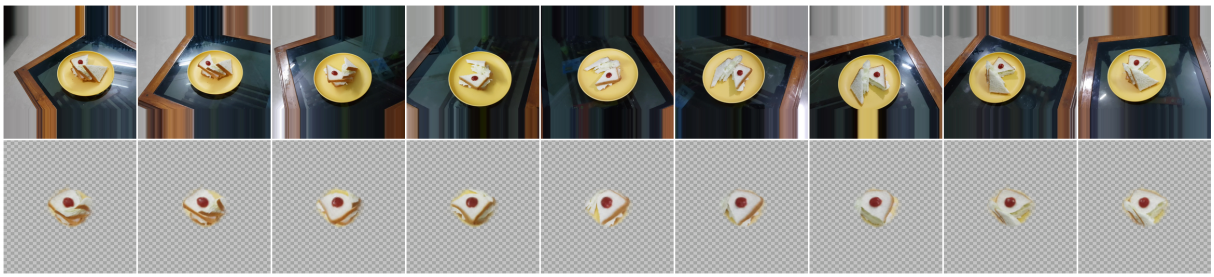


Figure 4.17: Increasing the rotation degree leads to a failure in jointly aligning the images of the two sandwiches

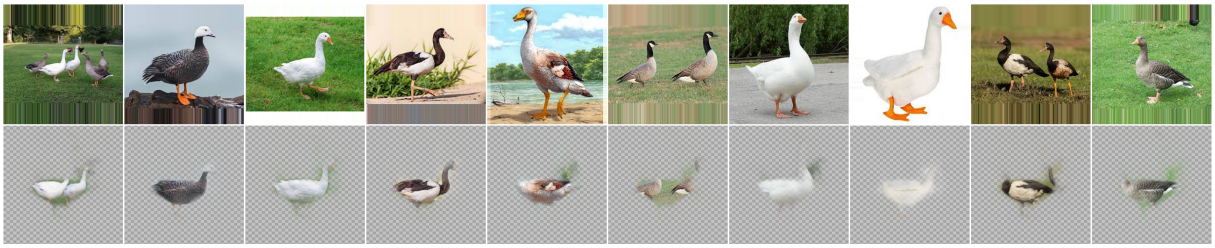


Figure 4.18: Failed attempt to even detect the main shared content correctly among all images that contain multiple geese

atlas that serves as a reference for aligning images into a common, canonical space. The atlas is refined iteratively, progressively aligning key features such as edges, textures, and other invariant visual elements across images with varying poses, transformations, and backgrounds. This iterative refinement helps the framework to maintain robust alignment performance even when dealing with diverse datasets Ofri-Amar et al., 2023.

However, the method of averaging features to initialize the atlas has notable drawbacks. By treating all features—relevant or not—equally, the framework risks focusing on less meaningful, noisy content. This approach fails to prioritize semantically significant information, potentially causing inefficiencies in the alignment process. The inability to distinguish between critical and irrelevant features can negatively impact the framework's convergence on semantically consistent representations. Consequently, this limitation may reduce alignment accuracy and hinder overall performance Ofri-Amar et al., 2023.

4.2.1 K-means Clustering

K-means clustering can mitigate the limitations of averaging by grouping similar features based on their distance in the feature space. Within Neural Congealing, K-means clusters features derived from DINO keys and binary co-segmentation masks into distinct groups that represent semantically more similar content. Each cluster reflects subsets of features with shared characteristics, creating smaller, coherent atlases that align more effectively.

The application of K-means clustering depends on the dataset type. For datasets generated through image augmentation, clustering co-segmented images using their RGB pixel values captures meaningful variations introduced during augmentation. On the other hand, for datasets containing images of objects from the same class but in different poses, clustering based on binary co-segmentation masks derived from DINO is preferable. These masks isolate shared semantic content while ignoring irrelevant RGB information, ensuring that alignment focuses exclusively on invariant, meaningful features. One detail to add is that in this work, the application of K-means clustering relied on a pre-trained ResNet-50 model (He et al., 2015), which is a 50-layer CNN with bottleneck layers (1x1→3x3→1x1 convolutions to reduce and restore dimensions efficiently) and skip connections (identity shortcuts for direct gradient flow), to extract features from the images, independent of the dataset type.

4.2.2 Results

This section focuses on how the K-means clustering algorithm was applied to improve the results presented in section 4.1.

Geese

For the results, the first dataset consists of geese, shown in figure 4.19. The framework failed to jointly align this dataset because some images contain multiple geese, making it challenging for the framework to perform accurate alignment. Due to the dataset in figure 4.19 containing

Figure 4.19: A dataset that the framework failed to jointly align, as shown in figure 4.18, due to the presence of multiple geese in some of the images.

geese of varying colors, clustering based on binary co-segmentation masks provides a more structured organization. In figure 4.21, the clustering results are visualized: the number of rows corresponds to the number of clusters. For example, the images in the first row belong to the first cluster, while those in the second row belong to the second cluster.
By clustering the geese into two groups, figure 4.20 demonstrates that the clustering successfully grouped geese pointing to the right together, while excluding images that do not contain at least one goose oriented in that direction

Using the images from the first cluster in figure 4.20 and applying the framework for joint alignment, figure 4.21 demonstrates that the framework successfully aligns the subset of geese images.

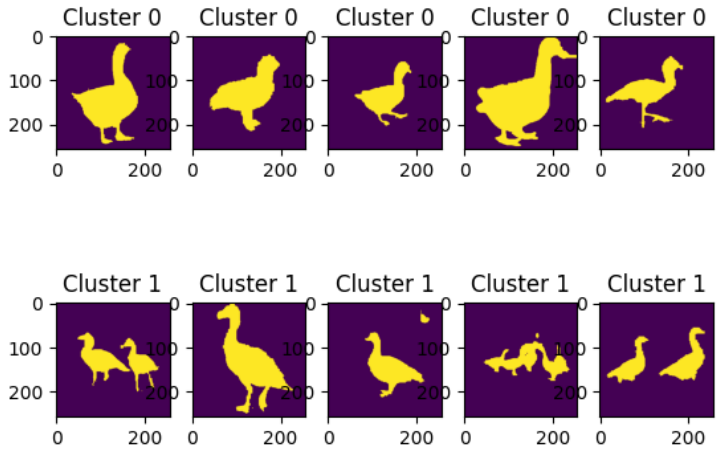


Figure 4.20: Using $k = 2$ clustering on the geese dataset in figure 4.19, based on their binary co-segmentation masks, it is evident that geese facing to the right are grouped into the same cluster.

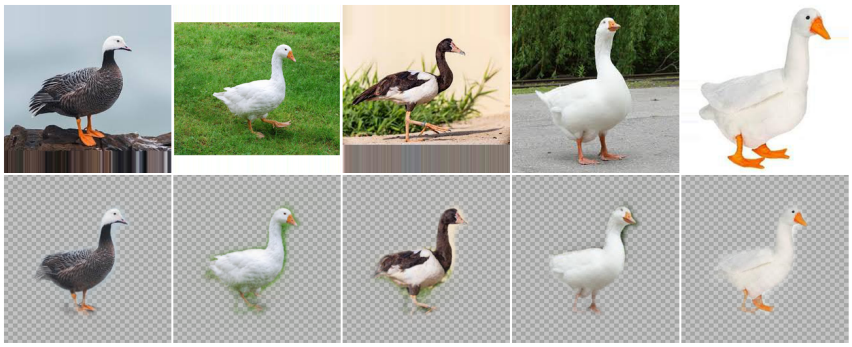


Figure 4.21: The framework succeeds in jointly aligning the gooses sub-dataset that was clustered through k-means clustering

Figure 4.22: The initialized average atlas before the training starts of figure 4.18

Figure 4.23: The initialized average atlas before the training starts of figure 4.21

By comparing figure 4.22 and figure 4.23, the reason the framework successfully aligns the goose dataset in figure 4.21 becomes clear. In figure 4.23, the initialized average atlas is more coherent, with the geese images grouped effectively through a clustering algorithm, making it even easier to visually distinguish the individual geese compared to the unclustered version in figure 4.22.

Pigeons

Another dataset where the framework struggled with joint alignment is the pigeons dataset, shown in figure 4.24. The framework fails to flip the pigeons correctly, as seen in figure 4.15.

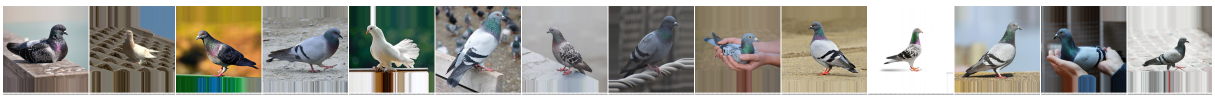


Figure 4.24: The pigeons dataset includes two types of pigeons, either pointing to the right or to the left.

By applying the K-means clustering algorithm to the pigeons dataset, the algorithm successfully divides the dataset into two clusters. The pigeons facing left are assigned to the first cluster, as shown in figure 4.25, while those facing right are grouped into the second cluster.

By taking the images of pigeons from the second cluster in figure 4.25 and allowing the framework to jointly align them, the framework is able to successfully align the pigeons sub-dataset. This shows how clustering the dataset based on orientations aids the framework in achieving better alignment by addressing the challenge of misaligned pigeon orientations. The result, seen in figure 4.21, illustrates improved joint alignment within the clustered subset of the dataset.

Quantitative Results

The similarities in the table seen in table 4.1 were calculated between the images grouped by the clustering algorithm and their corresponding congealed counterparts in the atlases, for both the normal and cluster joint alignment methods. The results show that it was more effective to first

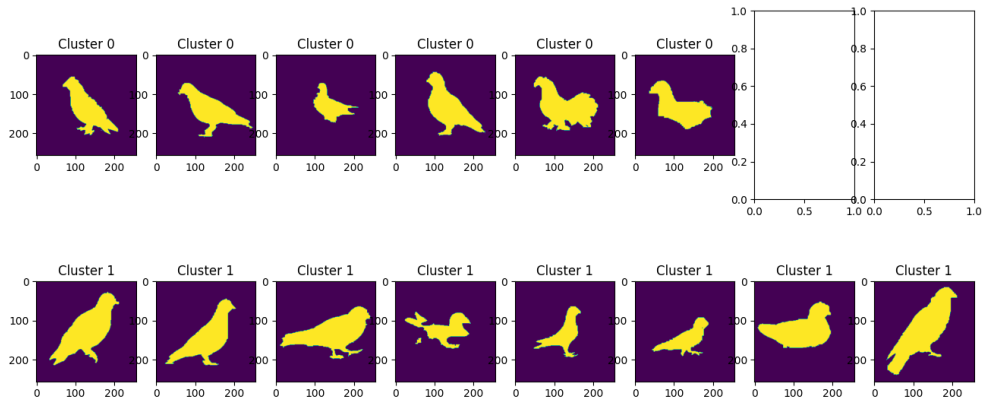


Figure 4.25: the K-means algorithm manages to cluster the pigeons of the dataset in figure 4.24 to two clusters according to their orientation.

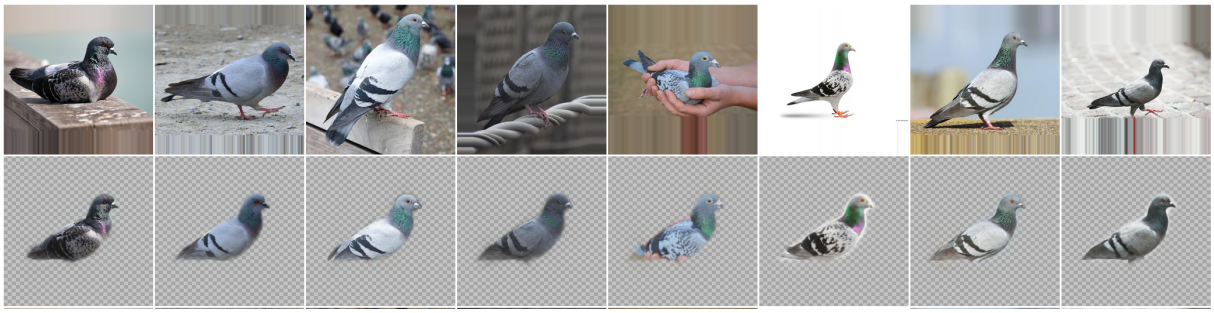


Figure 4.26: The framework succeeds in jointly aligning the pigeons sub-dataset that was clustered through K-means clustering. Specifically, the framework managed to jointly align the sub-dataset that contains pigeons pointing to the right.

cluster the images and then allow the framework to jointly align them separately. This approach improved alignment accuracy, as demonstrated by the higher similarity values for the cluster joint alignment method compared to the normal alignment method for both the Geese and Pigeons datasets.

4.2.3 Multiple Passes Joint Alignment

The K-means algorithm helps generate average atlas spaces that are easier to align, reducing misalignment and the framework's workload. However, it may inadvertently exclude defining features by classifying some as irrelevant or outliers. This raises the question: can the framework rely on clustering to align the feature space without missing any features, by clustering the dataset and then jointly aligning all the clusters in successive passes until a final canonical pose

Dataset	Before Alignment	Normal Joint Alignment	Clustering-based Joint Alignment
Geese	0.563	0.604	0.680
Pigeons	0.564	0.639	0.708

Table 4.1: Comparison of Similarity values Before and After Different Methods of Joint Alignments

is reached?

To explore this, a dataset is created from 20 wood chair images with rotation increments of 8 degrees, resulting in a total misalignment of 160 degrees.

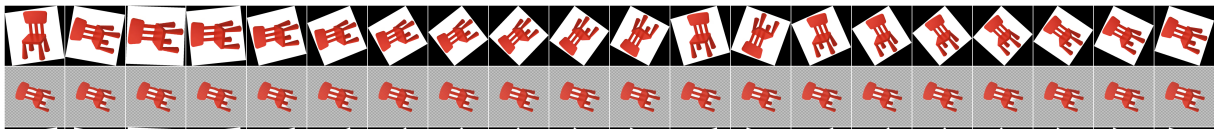


Figure 4.27: The framework successfully managed to jointly align the Wood Chairs dataset, which contains 20 images with a rotational misalignment of 160 degrees.

The framework successfully aligned the Wood Chair dataset, as seen in figure 4.27, rotating all images to the same angle. This raises the question of whether clustering the dataset beforehand could achieve the same result.

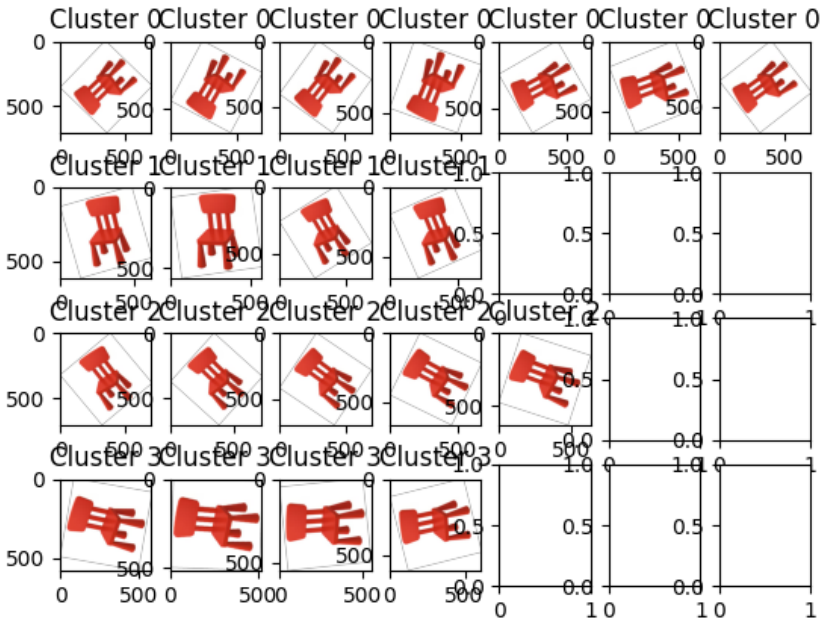
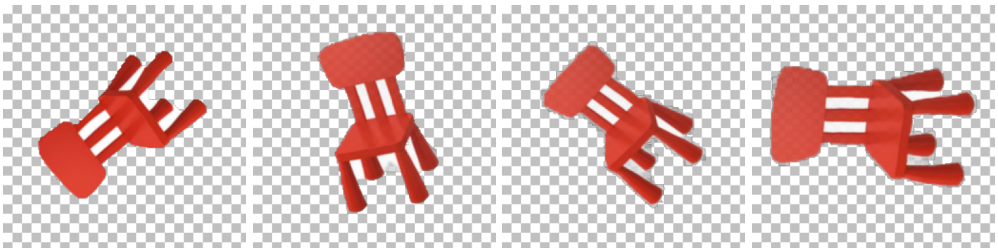


Figure 4.28: clustering the wood chairs dataset by the RGB pixels values to four clusters

The experiment begins by inputting the co-segmented wood chair images into the K-means algorithm with four clusters. The aim is to determine whether the framework can correctly align each cluster or sub-dataset.
As shown in figure 4.28, the K-means algorithm grouped the images into four sub-datasets. The maximum size of a sub-dataset is 7 images, and the misalignment on the x-axis is visually limited to 45 degrees at most.



(a) the framework managed to correctly align the first cluster which is shown in the first row in figure 4.28

(b) the framework managed to correctly align the second cluster which is shown in the second row in figure 4.28

(c) the framework managed to correctly align the third cluster which is shown in the third row in figure 4.28

(d) the framework managed to correctly align the forth cluster which is shown in the forth row in figure 4.28

Figure 4.29: the framework managed to jointly align all the four sub-datasets generated by the K-means algorithm separately.

As seen in figure 4.29, the framework successfully aligned the four sub-datasets generated by the K-means algorithm, reducing the dataset from 20 images to just 4.

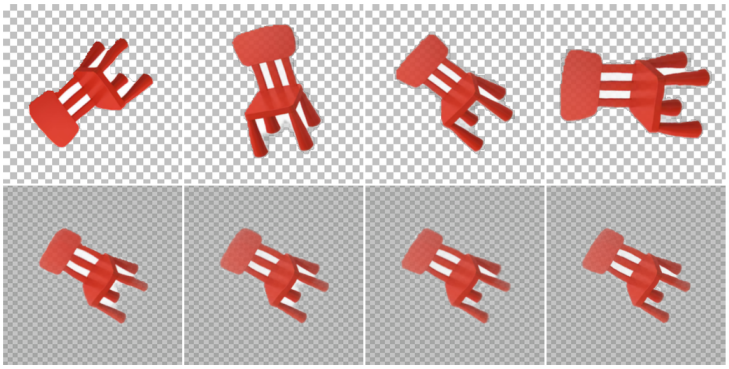


Figure 4.30: the framework managed to jointly align the 4 images resulting in jointly aligning the whole dataset and the final result is very similar to the result to the original result in figure 4.27

When the framework attempts to jointly align the 4 images in figure 4.29, the result closely mirrors the original alignment in figure 4.27, achieved without clustering. This demonstrates that the clustering-based approach can yield similar results to the naive method, where all features are treated equally. It highlights that instead of aligning the dataset in one pass, performing joint alignment in multiple passes may be a valuable alternative.

5 Conclusion

This thesis outlines the methodology used to test and optimize the Neural Congealing framework, specifically designed to align datasets that suffer from various forms of misalignment, noise, and clutter. The framework's ability to handle datasets with significant transformations, such as rotational misalignments, scale variations, and the presence of background noise, is carefully evaluated across multiple experiments. A series of qualitative and quantitative tests were conducted to explore the performance of the framework under these conditions.

The analysis begins by testing the framework's robustness under different types of misalignments, with particular attention to the effect of 2D and 3D rotations, axis misalignments, and the challenges presented by aligning datasets containing multiple objects. The framework's performance is assessed by tracking its alignment accuracy under various image transformations, including rotations ranging from minor angular shifts to extreme angles that challenge the algorithm's ability to realign the images properly.

The framework's limitations are also discussed, particularly its struggles with handling datasets that involve high rotational misalignments (particularly greater than 180 degrees), 2D and 3D axis misalignments, and datasets with multiple objects or cluttered backgrounds. The framework tends to falter when dealing with these complex scenarios, often failing to correctly align images that require flipping or dealing with multiple objects simultaneously.

To address these limitations, K-means clustering is introduced as a technique to improve the initial alignment of images. By grouping similar images before joint alignment, K-means clustering helps the framework handle datasets more effectively, especially in cases where images contain multiple objects or exhibit extreme misalignments. The experimental results with the Geese and Pigeons datasets demonstrate that clustering significantly improves the alignment accuracy, compared to the primary atlas initialization method.

Quantitative results clearly show that the clustering-based alignment method outperforms the primary method of atlas initializations. The improved initialization through clustering enhances the framework's ability to handle more complex dataset. These findings suggest that clustering can be a valuable approach in improving the performance of the Neural Congealing framework,

particularly for challenging alignment tasks.

Bibliography

Caron, M., H. Touvron, I. Misra, H. Jégou, J. Mairal, P. Bojanowski, and A. Joulin (2021). *Emerging Properties in Self-Supervised Vision Transformers*. arXiv: 2104.14294 [cs.CV].

Chen, M., N. J. Tustison, R. Jena, and J. C. Gee (2023). "Image Registration: Fundamentals and Recent Advances Based on Deep Learning". In: *Machine Learning for Brain Disorders*. Ed. by O. Colliot. New York, NY: Springer US, pp. 435–458. URL: https://doi.org/10.1007/978-1-0716-3195-9_14.

He, K., X. Zhang, S. Ren, and J. Sun (2015). *Deep Residual Learning for Image Recognition*. arXiv: 1512.03385 [cs.CV]. URL: https://arxiv.org/abs/1512.03385.

Ikotun, A. M., A. E. Ezugwu, L. Abualigah, B. Abuhaija, and J. Heming (2023). "K-means clustering algorithms: A comprehensive review, variants analysis, and advances in the era of big data". In: *Information Sciences* 622, pp. 178–210. URL: https://www.sciencedirect.com/science/article/pii/S0020025522014633.

Learned-Miller, E. (2006). "Data driven image models through continuous joint alignment". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 28.2, pp. 236–250.

Ofri-Amar, D., M. Geyer, Y. Kasten, and T. Dekel (2023). "Neural Congealing: Aligning Images to a Joint Semantic Atlas." In: arXiv: 2302.03956.

Zhang, L., J. Song, A. Gao, J. Chen, C. Bao, and K. Ma (2019). *Be Your Own Teacher: Improve the Performance of Convolutional Neural Networks via Self Distillation*. arXiv: 1905.08094 [cs.LG].