FLOWING 🌊: Implicit Neural Flows for Structure-Preserving Morphing

Arthur Bizzi[1] **Matias Grynberg**[2] **Vitor Matias**[3] **Daniel Perazzo**[4,5] **João Paulo Lima**[4,6]
Luiz Velho[4] **Nuno Gonçalves**[7,8] **João Pereira**[9] **Guilherme Schardong**[7] **Tiago Novello**[4]

[1]EPFL [2]University of Buenos Aires [3]University of São Paulo [4]IMPA [5]CSML-IIT
[6]Universidade Federal Rural de Pernambuco [7]ISR-UC [8]INCM [9]University of Georgia

Abstract

Morphing is a long-standing problem in vision and computer graphics, requiring a time-dependent warping for feature alignment and a blending for smooth interpolation. Recently, multilayer perceptrons (MLPs) have been explored as implicit neural representations (INRs) for modeling such deformations, due to their meshlessness and differentiability; however, extracting coherent and accurate morphings from standard MLPs typically relies on costly regularizations, which often lead to unstable training and prevent effective feature alignment. To overcome these limitations, we propose FLOWING (**FLOW** morph**ING**), a framework that recasts warping as the construction of a differential vector flow, naturally ensuring continuity, invertibility, and temporal coherence by encoding structural flow properties directly into the network architectures. This flow-centric approach yields principled and stable transformations, enabling accurate and structure-preserving morphing of both 2D images and 3D shapes. Extensive experiments across a range of applications—including face and image morphing, as well as Gaussian Splatting morphing—show that FLOWING achieves state-of-the-art morphing quality with faster convergence. Code and pretrained models are available in `https://schardong.github.io/flowing`.

1 Introduction

Morphing is a long-standing problem in computer vision and graphics [12; 56], with applications in image editing [48; 4], biometrics [46; 13], and 3D shape interpolation [27; 31; 42]. The task consists of continuously interpolating between two signals while ensuring that the intermediate representations remain structurally consistent. Traditionally, morphing is decomposed into two stages: a **warping** stage, which aligns source and target features over time (ideally through a one-parameter family of diffeomorphisms), and a **blending** stage, which interpolates between the aligned signals.

Recently, ifmorph [44] introduced a face morphing approach that employs *multilayer perceptrons* (MLPs) as *implicit neural representations* (INRs) to model the warping transformation. INRs provide a differentiable and memory-efficient solution for representing low-dimensional signals and have been successfully applied to surface reconstruction [47; 14; 58; 30; 45], surface evolution [27; 31; 41], radiance fields [28; 3], and image modeling [1; 34; 35]. However, applying generic MLPs to represent warping transformations presents important limitations. For instance, enforcing temporal coherence requires explicit regularization in the loss function, which greatly increases training time and may lead to convergence issues. In addition, unconstrained MLPs lack structural priors, making the learned deformation prone to undesirable behaviors such as singularities—regions where the mapping becomes non-invertible—resulting in artifacts that degrade the morphing quality (see the first row in Figure 1, bottom-left).

39th Conference on Neural Information Processing Systems (NeurIPS 2025).

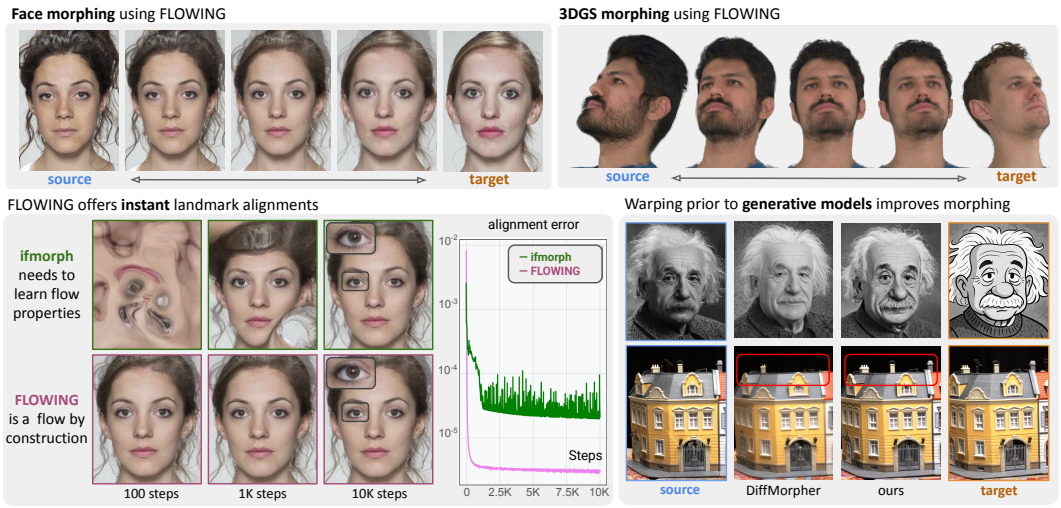

Figure 1: We present **FLOWING**, a robust and theoretically grounded framework for fast, accurate, and structure-preserving morphing. It enables smooth and temporally consistent morphs by learning a structure-preserving flow, applicable to both 2D (top-left) and 3D (top-right) data; the latter uses 3DGS for representing 3D faces. Bottom-left: FLOWING instantly aligns landmarks by construction, outperforming SoTA methods in both visual quality and convergence speed. Bottom-right: Integrating FLOWING with generative models improves morphing fidelity and semantic coherence.

To address these limitations, we propose **FLOWING**, a framework that employs specialized flow-based INRs that perform **structure-preserving warpings by construction**. We reframe morphing as the problem of constructing an interpolating flow, allowing us to inherit the structural properties of flow operators through flow-based neural architectures, such as *neural ODEs* (NODEs) [9] and *neural conjugate flows* (NCFs) [6]. FLOWING adapts these architectures to the 2D/3D warping contexts and leverages their architectural priors to produce morphings that are valid by construction. As a result, it enables near-instant training, significantly faster than generic MLPs that depend on soft regularization to enforce coherence. Figure 1 showcases FLOWING's ability to generate smooth, consistent, and structure-preserving morphs in both 2D images and *3D Gaussian splatting* (3DGS) [20], with clear improvements in visual quality, alignment, speed, and semantic coherence over prior methods. In summary, our contributions are:

- A principled, flow-based morphing approach that leverages specialized INRs to enforce continuity and temporal coherence by construction. This enables learning highly detailed, structure-preserving morphings near instantly, using only sparse landmark correspondences.
- An adaptation of flow-based architectures to the morphing setting, combined with SIRENs [47; 32], allowing the representation of complex, high-frequency deformations and accurate keypoint matching. To minimize spurious deformations, we incorporate thin-plate energy regularization [7; 49], enabled by a novel forward-mode differentiation scheme.
- Demonstration of our approach across diverse morphing tasks—including face morphing, general image morphing, and 3DGS morphing—achieving strong results in 3D splat morphing via a novel blending scheme.

2 Related works

Warping and morphing techniques have been widely studied in computer graphics and vision [12; 56; 55]. Classical approaches rely on geometric transformations such as thin-plate splines [26], radial basis functions [7], and triangle meshes to morph between images [4; 49]. While effective, these methods often struggle to handle smooth and complex deformations. Moreover, standard OpenCV-style morphing combines simple linear interpolation between features with a shared triangulated domain, which is not always available. In contrast, by introducing flow concepts into the model architecture, our method (FLOWING) achieves high-quality warping for both images and 3D data. It

requires only feature correspondences as input and supports nonlinear interpolation between features, enabling more flexible and robust morphing.

Implicit neural representations (INRs) have gained significant attention for encoding continuous low-dimensional signals directly in the parameters of neural networks. Early works such as SIREN [47] and Fourier feature networks [52] enhanced INR expressiveness by mapping input coordinates to sinusoidal functions, enabling detailed reconstructions of images, 3D shapes, and physical fields. More recently, INRs have been applied to 2D face morphing [44], where SIRENs are used to parameterize the warping and additional costly constraints are introduced to enforce flow properties. In contrast, our method leverages flow-based architectures that enable faster, more efficient, and robust morphing of both 2D and 3D data, with deformations learned as continuous, structure-preserving flows.

Flow-based networks have been explored for modeling continuous-time dynamics and transformations. NODEs [9] provide a powerful yet computationally intensive framework for learning continuous-time dynamical systems, where inference through discrete layers is replaced by numerical integration. Originally developed for continuous normalizing flows, NODEs have since been applied to medical image registration [2; 57; 50; 51], solving PDEs [5]. More recently, NCFs [6] were introduced as an alternative approach to model dynamics by topologically deforming affine flows, enabling greater efficiency through parallelism. In this work, we adapt both NODEs and NCFs to the morphing setting for images and 3DGS, further enhancing their representational power with sinusoidal activations.

Generative methods have also emerged as a powerful and flexible approach for morphing, with DiffMorpher [59] presenting general image morphing without reference landmarks. However, they face two key limitations: their outputs are constrained to a fixed resolution, and they require pre-aligned target images, adding significant preprocessing overhead. Our method addresses these challenges by introducing flow-based warping, which acts as a sophisticated, non-linear alignment mechanism. This enables seamless integration with generative blending techniques, improving both alignment quality and applicability.

3 FLOWING

3.1 Problem setup: morphing as flows

Given source and target media objects $f^0, f^1 : \Omega \subset \mathbb{R}^n \to \mathbb{R}^m$, where Ω denotes their supports, annotated with K feature correspondences $\{p_i^0, p_i^1\} \subset \Omega$, our goal is to construct a continuous, time-dependent family of warpings $\Phi : \Omega \times [0, 1] \to \mathbb{R}^n$ that allows us to smoothly interpolate between the source features at $t = 0$ and the corresponding target features at $t = 1$, such that $\Phi(p_i^0, 1) = p_i^1$. For morphing, Φ should have additional properties:

• **Uniqueness.** The path traced by each feature must be unique, otherwise distinct features may overlap, leading to unstructured, incoherent interpolations.

• **Invertibility.** The forward and backward mappings of source and target features should match at intermediate times and remain symmetric under time reversal to avoid artifacts during blending.

• **Energy-minimization.** Feature paths should be minimal and smooth for coherent interpolation; otherwise, overfitting may introduce artifacts or singularities that degrade morphing quality.

The key idea behind FLOWING is that the first two properties arise from the definition of **flows**: by constraining features to move as an integral over an underlying vector field, we may leverage the uniqueness and reversibility of its orbits. This allows us to recast the morphing problem as the construction of a flow operator [54], enforcing corresponding features to remain consistent over time:

$$\Phi(p_i^0, t) = \Phi(p_i^1, t - 1) \text{ for } i \in \{1, \dots, K\} \text{ and } t \in [0, 1]. \tag{1}$$

To enforce these structural properties on the warping map, we employ a θ-parametrized flow representation Φ_θ (see Sec. 3.3). The third property is imposed by minimizing the curvature of the orbits of Φ_θ across the domain Ω. This is achieved with the following thin-plate-like optimization problem:

$$\arg\min_\theta \|\mathbf{J}\Phi'_\theta(\mathbf{x}, 0)\|_2^2 \quad \text{subject to} \quad \Phi_\theta(p_i^0, t) = \Phi_\theta(p_i^1, t - 1), \tag{2}$$

where $\Phi'_\theta(\mathbf{x}, 0)$ denotes the time derivative of $\Phi_\theta(\mathbf{x}, t)$ at $t = 0$ and $\mathbf{J}\Phi'_\theta$ its Jacobian. Once the features $\{p_i^0, p_i^1\}$ are aligned, the resulting flow can be used to warp the source and target signals to

an intermediate time, after which the resulting warpings are blended together. Figure 2 provides an overview of the morphing process.

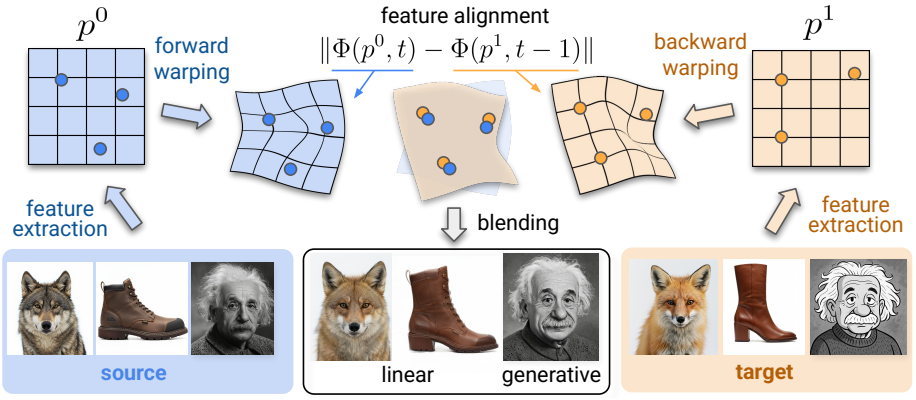

Figure 2: **Overview of FLOWING.** Given source and target images I^0, I^1 we extract landmark pairs (p_i^0, p_i^1) with a feature extractor (`dlib`, Xfeat, etc). We train a flow ϕ such that $\phi(p^0, t) = \phi(p^1, t-1)$, effectively mapping p^0 to p^1. At inference, we warp I^0 forward by t units and I^1 and backward by $t-1$ units, then blend them together with methods such as linear blending or generative models.

3.2 Training

Training a flow-based architecture Φ_θ requires defining a loss function to solve the optimization problem in (2). Note that the uniqueness and invertibility properties are guaranteed by construction, since Φ_θ is a flow. Therefore, the loss function $\mathcal{L}$ consists of a data constraint to enforce feature matching and a regularization term to penalize path distortion:

$$\mathcal{L}(\theta) = \underbrace{\sum_i \int_{[0,1]} \left\| \Phi_\theta(p_i^0, t) - \Phi_\theta(p_i^1, t-1) \right\|_2^2 \, dt}_{\text{Data constraint}} + \underbrace{\lambda \int_\Omega \|\mathbf{J}\Phi'_\theta(\mathbf{x}, 0)\|_2^2 \, d\mathbf{x}}_{\text{Thin-plate constraint}}, \tag{3}$$

where $\lambda > 0$ is a parameter. In practice, the data term is enforced on only a few time steps, since uniqueness ensures that feature correspondences hold across the interval. We find out that using $t = 0.5$ for NODE and $t = \{0, 0.25, 0.5, 0.75, 1\}$ for NCF is sufficient.

For thin-plate term, we minimize the Jacobian of $\Phi'_\theta(\mathbf{x}, 0)$ over the domain, which is equivalent to minimizing the second derivative of the integral path $\Phi(\mathbf{x}, t)$ for each $\mathbf{x}$. Since Φ is a flow, explicit time sampling is unnecessary—one reason flow-based architectures train faster than generic MLPs. The integral is approximated via Monte Carlo methods, with λ as the regularization weight. To further accelerate computation, we implement a *forward differentiation* (FD) scheme based on generalized dual-number arithmetic, enabling derivatives to be computed in parallel with inference, significantly reducing overhead. Our ablation study shows that FD makes the thin-plate loss calculation **23 times faster** than standard `autograd`. Full results and details are given in Appendix A.

3.3 Flow-based architectures

Flow-based networks provide a principled representation for the time-dependent warping Φ_θ, as they hold desirable properties such as continuity, invertibility, and temporal coherence. Compared to using MLPs to parametrize the flow operator Φ_θ, which require expensive additional terms to approximate uniqueness and invertibility in addition to (2), flow-based architectures enforce these properties by construction. As a result, they rely on simpler losses, may achieve near-instant training, and provide accurate alignment while avoiding the catastrophic artifacts that arise when MLPs fail to capture proper flow properties.

Figure 3 compares an MLP-based flow (ifmorph [44]) with our approach (FLOWING). For this experiment, we employed `dlib` to extract features centered on the faces, which explains why the ears are not aligned. In this work, we focus on two flow-based networks: Neural ODEs and Neural Conjugate Flows.

Neural ODEs. First, we propose using NODEs [9] to model the vector field Φ', the time derivative of the warping deformation Φ, with a neural network $\mathcal{F}_\theta$. Thus, inference for these models involves performing numerical integration over the vector field $\mathcal{F}_\theta$; analogously, it may be interpreted as deep ResNets[15] with "continuous depth". Specifically, NODEs take the following formulation:

$$\frac{d}{dt}\mathbf{x} = \mathcal{F}_\theta(\mathbf{x}) \implies \Phi(\mathbf{x}_0, t) = \mathbf{x}(0) + \int_0^t \mathcal{F}_\theta\big(\mathbf{x}(\tau)\big)\, d\tau. \tag{4}$$

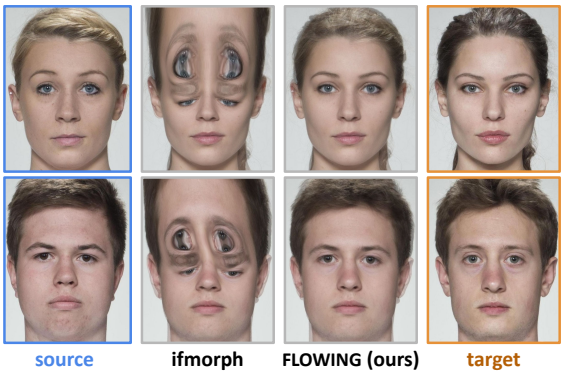

Figure 3: Comparison between ifmorph and FLOWING. While ifmorph fails to preserve structure, FLOWING produces clean and realistic interpolations by enforcing flow properties such as invertibility and trajectory uniqueness.

To achieve high-quality matching, the vector field $\mathcal{F}_\theta$ is modeled as a SIREN [47], allowing it to capture highly detailed spatial deformations necessary to handle the diverse range of distances and orbits traversed by each pair of correspondences. Moreover, to ensure that orbit deformations remain minimal, we penalize the norm of the Jacobian matrix of $\mathcal{F}_\theta$ over the domain Ω.

During training, we integrate $\mathcal{F}_\theta$ using a fourth-order Runge–Kutta method. We integrate the source and target points to $t = 0.5$ using the same number of integration steps for the forward and backward dynamics. Since NODE-based approaches are often prone to numerical errors during integration, we include in Appendix B an ablation study on the total number of integration steps involved in this process, showing that we can use very few integration steps while both preserving an accurate approximation of the vector field $\mathcal{F}_\theta$ and keeping the training efficient. Moreover, at inference time, the number of integration steps can be increased arbitrarily, without additional training cost, to more accurately capture the dynamics of $\mathcal{F}_\theta$.

Neural conjugate flows. While NODEs explicitly integrate neural vector fields, they can become computationally expensive when many steps are required during inference. Neural conjugate flows (NCFs) [6] offer an alternative formulation based on topological conjugation. Instead of sequential integration, NCFs employ invertible neural networks to deform the orbits of simple affine flows, enforcing a simplified topology. Precisely, the flow-based warping is given by:

$$\Phi_\theta(\mathbf{x}, t) = H_\theta^{-1} \circ \Psi\big(H_\theta(\mathbf{x}), t\big), \quad (\mathbf{x}, t) \in \mathbb{R}^n \times \mathbb{R}. \tag{5}$$

We parameterize H_θ and H_θ^{-1} using invertible architectures known as coupling layers [11]. These apply alternating, memory-equipped transformations in sequence, guaranteeing invertibility by construction. Importantly, the resulting conjugated flow Φ_θ remains a valid flow, inheriting key properties such as continuity, invertibility, and associativity from the original affine system.

3.4 Blending

Having established how FLOWING constructs flow-based warpings that guarantee structural properties and accurate feature alignment, we now turn to the second stage of morphing: **blending**.

Image morphing. Let $I^0, I^1 : \mathbb{R}^2 \to \mathcal{C}$ denote two input images and $\Phi_\theta : \mathbb{R}^2 \times \mathbb{R} \to \mathbb{R}^2$ be a flow aligning their features over time. We define the warped images as $\mathcal{I}^i(\mathbf{x}, t) = I^i\big(\Phi_\theta(\mathbf{x}, i - t)\big)$, ensuring spatial alignment at each intermediate timestep $t \in [0, 1]$. A straightforward morphing is then obtained by linearly blending the aligned images:

$$\mathcal{I}(\mathbf{x}, t) = (1 - t)\mathcal{I}^0(\mathbf{x}, t) + t\mathcal{I}^1(\mathbf{x}, t), \tag{6}$$

yielding a smooth function $\mathcal{I} : \mathbb{R}^2 \times \mathbb{R} \to \mathcal{C}$ that interpolates between I^0 and I^1.

While generative models can interpolate between images, they typically lack explicit feature alignment. To address this limitation, we first apply our flow-based warping Φ_θ to align features over time, and

then perform blending in the latent space of a pretrained generative model. Let $\mathscr{E}$ and $\mathscr{D}$ denote the encoder and decoder of a generative model. We define generative morphing as:

$$\mathscr{I}(\cdot, t) = \mathscr{D}\left((1-t)\boldsymbol{c}^0(t) + t\boldsymbol{c}^1(t)\right), \quad \text{where} \quad \boldsymbol{c}^i(t) = \mathscr{E}\big(\mathscr{I}^i(\cdot, t)\big). \tag{7}$$

This strategy combines explicit spatial alignment with perceptual quality, producing temporally coherent transitions that significantly outperform naive latent-space blending.

3D Gaussian Splatting morphing. Beyond images, we show that morphing with FLOWING can be extended to 3DGS [20; 36] which has recently emerged as a popular format for 3D representation. 3DGS enables photorealistic rendering from a set of Gaussians $\mathscr{G}$ with each Gaussian $g_k = \{p_k, \alpha_k, c_k, \Sigma_k\}$ consisting of a center $p_k \in \mathbb{R}^3$, opacity α_k, view-dependent color $c_k \in \mathbb{R}^3$, and covariance matrix $\Sigma_k \in \mathbb{R}^{3\times3}$ that encodes orientation and spread.

Our method morphs between two Gaussian sets, $\mathscr{G}^0$ and $\mathscr{G}^1$, producing an intermediate set $\mathscr{G}(t)$ with $\mathscr{G}(0) = \mathscr{G}^0$ and $\mathscr{G}(1) = \mathscr{G}^1$. We employ a FLOWING network Φ_θ to smoothly align Gaussian centers over time, while linearly blending the opacity parameters α_k^i to ensure seamless transitions. Specifically, for each Gaussian $g_k^i \in \mathscr{G}^i$, we compute:

$$g_k^i(t) = \big\{\Phi_\theta(p_k^i,\ t-i),\ |1-i-t|\,\alpha_k^i,\ c_k^i,\ \Sigma_k^i\big\}, \tag{8}$$

which jointly applies warping and blending. The final morphed set is then defined by $\mathscr{G}(t) = \mathscr{G}^0(t) \cup \mathscr{G}^1(t)$. This 3DGS morphing procedure preserves structural coherence through flow-based alignment and achieves photorealistic 3D transitions via our linear blending strategy.

4 Experiments

We evaluate FLOWING on face and image morphing, as well as on 3DGS morphing, using four diverse datasets. These experiments demonstrate the effectiveness of our approach in both 2D and 3D settings. Additionally, we provide ablation studies in Appendix B to validate our architectural choices and regularization strategies.

Evaluation datasets. For face images, we use the FRLL dataset [10], which contains 102 identities with 5 images captured at fixed angles and two expressions (neutral and smiling). We use the neutral, front-facing images. Following the protocol in [43], we construct 1220 pairs, with landmarks extracted using `dlib` [19; 40]. As a post-processing step, we apply FFHQ alignment and cropping, producing images at a resolution of 1350^2. The second dataset is a subset of MegaDepth [25], which provides multi-view scene landmarks. From this subset, we extract 275 image pairs based on the cosine similarity between their ResNet [15] embeddings, with feature correspondences obtained using Xfeat [38]. Third, we use eight subjects from NeRSemble [22], a multi-view collection of human heads. Following [39], we fit an FLAME model to each subject to extract 254 landmarks per head. Finally, we include in-the-wild face images from FFHQ [18] for qualitative evaluation.

FLOWING is implemented in PyTorch [33] and trained with the Adam optimizer [21]. At each training step, we sample $20{,}000$ points from the spatial domain $[-1, 1]^2$ for the selected values of t.

4.1 Quantitative comparisons

To evaluate the performance of FLOWING on the morphing task, we consider two key components: **warping**, which aligns landmarks across signals over time, and **blending**, which generates smooth intermediate transitions. Accordingly, we organize our comparisons into two aspects: *landmark alignment* (4.1.1) and *blending quality* (4.1.2).

4.1.1 Landmark alignment

We compare FLOWING with ifmorph [44], testing both NCF and NODE backbones with sigmoid and SIREN activations. For a fair comparison, all methods are supervised under the same set of landmark points.

Metrics. We report the *mean squared error* (MSE) between warped target landmark positions across intermediate timesteps. For each landmark pair, we sample 10 evenly spaced values of $t \in [0, 1]$, warp the landmarks accordingly, and compute their MSE, averaging the results across all timesteps. A

perfect alignment would yield an average MSE of 0, since source and target landmarks would coincide exactly at every value of t. Experiments are conducted on three benchmarks: face / monument / 3D Gaussian avatar alignment.

Sinusoidal vs. non-sinusoidal activations. We compare NCF- and NODE-based variants of FLOWING using SIREN activations and their sigmoid counterparts to justify our architectural choices for high-frequency representations in morphing quality and convergence. Table 1 reports the average MSE across the FRLL, MegaDepth, and NeRSemble datasets for face warping, monument alignment, and 3D Gaussian avatar morphing. Results are shown for ifmorph, NCF, and NODE. For NCF and NODE, we evaluate both SIREN and sigmoid activations. Note that NCF (SIREN) and NODE (SIREN) outperform ifmorph by one to three orders of magnitude, while SIREN activations consistently yield lower errors than the sigmoid case. In particular, NODEs with sigmoid ("vanilla" configurations) perform significantly worse, confirming the importance of sinusoidal activations for accurate warping alignment.

Table 1: Results for the landmark alignment across the FRLL [10], MegaDepth [25] and NeRSemble [22] datasets. The best, second and third best results for each dataset/model combination are shown in green, yellow, and orange, respectively.

Model (Activation)	FRLL (↓)	MegaDepth (↓)	NeRSemble (↓)
ifmorph [44]	1.5E-3	2.9E-1	1.0E-3
NCF (sigmoid)	8.7E-3	2.0E+1	7.2E-5
NCF (SIREN)	3.9E-5	4.2E-4	7.3E-5
NODE (sigmoid)	6.2E-2	1.9E-1	5.0E-4
NODE (SIREN)	1.4E-4	4.9E-4	5.9E-5

NCF vs. NODE. Figure 4 shows that NCFs and NODEs achieve better results with significantly fewer training steps compared to ifmorph. While NODEs converge quickly, they typically show poorer feature alignment. In contrast, NCFs require more steps to converge, resulting in better feature alignment overall. However, due to minor pixel-level variations between source and target features, this does not necessarily translate to noticeable visual differences.

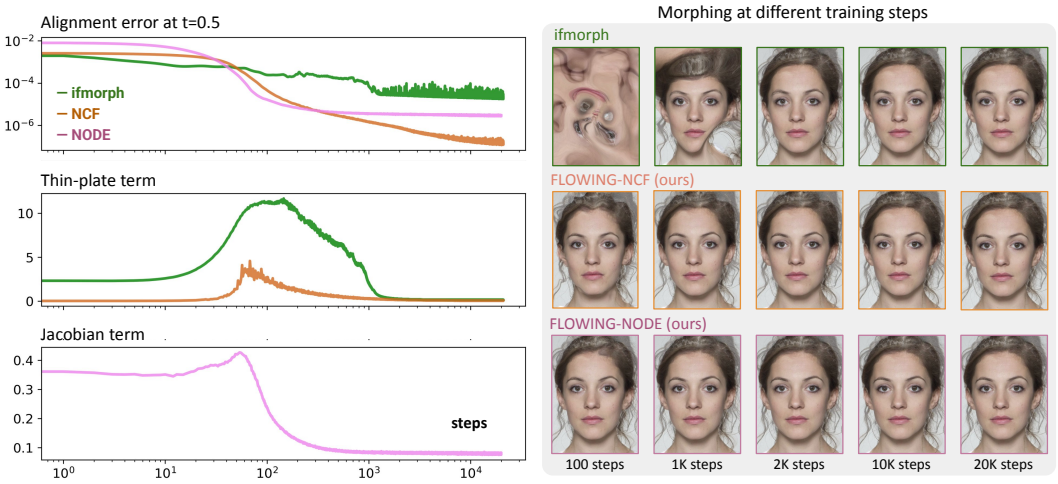

Figure 4: Convergence analysis of ifmorph (green), NCF (orange), and NODE (violet). Left: alignment and deformation metrics over training steps. For NODE, alignment is measured using the Jacobian term, while ifmorph and NCF use the Hessian-based thin-plate term. Right: qualitative morphing results at $t = 0.5$ after 100, 1000, 2000, 10000, and 20000 training steps. NCF and NODE achieve accurate alignment by 1000 steps, whereas ifmorph requires at least 2000 steps.

4.1.2 Blending quality

We evaluate blending quality using FRLL [10] for linear face morphing, following the same experimental setup as in the landmark alignment experiment (Sec. 4.1.1). We compare FLOWING with SIREN activations against ifmorph and a standard OpenCV baseline using Delaunay triangulation warping with linear blending[1]. Additional comparisons with the classical thin-plate spline method are given in Appendix F, and further experiments on video interpolation are discussed in Appendix E.

Metrics. Perceptual quality is measured using the *learned perceptual image patch similarity* (LPIPS) metric [60], computed between intermediate frames at $t = 0.5$ and both source and target images. We also measure the *Fréchet inception distance* (FID) [16] between generated morphs and original images, using 2048-dimensional feature vectors.

Table 2: Linear image blending results on the FRLL dataset [10]. The best, second-best, and third-best results are highlighted in green, yellow, and orange, respectively.

Morphing type	LPIPS(I^0, I) (↓)	LPIPS(I, I^1) (↓)	FID (↓)
OpenCV	0.233	0.236	32.426
ifmorph [44]	0.250	0.252	38.427
NCF (Ours)	0.221	0.224	33.300
NODE (Ours)	0.210	0.213	31.952

Table 2 shows that both NCF and NODE outperform ifmorph across all metrics. NCF and NODE also obtained better results than OpenCV in LPIPS. For FID, NODE achieves the best overall performance, while NCF remains competitive, slightly below OpenCV. Overall, NODE produces higher-quality and more consistent morphing results than NCF.

4.1.3 Warping and morphing times

Table 3 summarizes training and morphing times obtained on an RTX 4090 GPU for both warp training and morphing (warping inference + blending). We report NODE with both 2000 and 20000 training steps to highlight its fast convergence. As shown, training times for NODE and ifmorph remain stable across different landmark counts, whereas NCF exhibits increased training time with additional landmarks. Morphing time depends on image resolution: NODE provides faster training but slower morphing, while NCF has slower training and intermediate morphing speed.

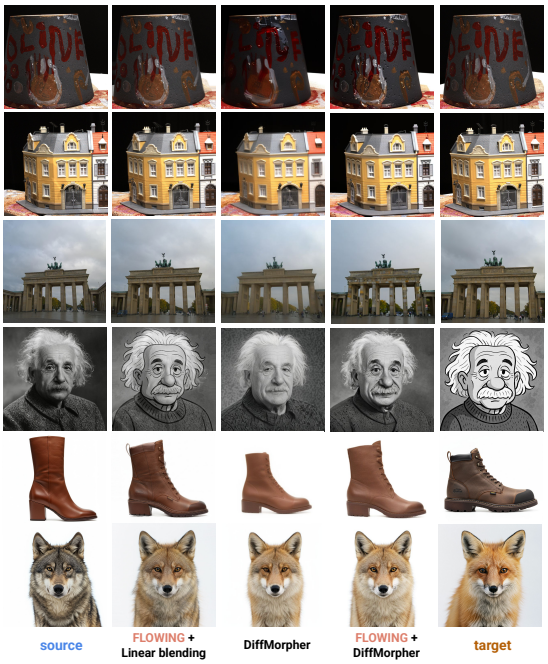

Figure 5: Applications of FLOWING to view interpolation (Rows 1-3), and stylization and object morphing (Rows 4-6). FLOWING produces smooth transitions even under environmental variations, maintaining geometric and photometric consistency. When combined with DiffMorpher, it further enhances structural coherence and visual fidelity, recovering details missed by DiffMorpher alone, such as the missing chimneys and the loss of column detail (Rows 2-3).

4.2 Image and face morphing

As shown in the previous sections, FLOWING can be used to warp between two images given a set of landmark correspondences. The warped images can then be blended using various techniques, such as linear blending, Poisson image editing [37], or generative blending [59]. In this work, we focus primarily on linear blending and generative blending via DiffMorpher [59]. Rows 1–3 of Figure 5 illustrate the view interpolation task, where FLOWING produces smooth transitions while better preserving both

[1] https://github.com/Azmarie/Face-Morphing/

geometric and photometric consistency across viewpoints. These examples show the benefit of applying FLOWING prior to generative blending in contrast to the usage of generative blending on its own. It improves both the consistency of the scene structure and reduces the number of missing elements or details. For instance, the building chimneys (Row 2) and architectural details (Row 3) remain intact, unlike in competing approaches. Rows 4–6 demonstrate stylization and object morphing, where combining FLOWING with generative models enhances structural coherence and visual fidelity across diverse visual domains, for example, in the photo-to-cartoon Einstein interpolation.

Table 3: Warp training times (68 and 130 landmarks) and morphing times at resolutions 256^2 and 1350^2. NODE is reported for both 2k and 20k training steps to highlight its fast convergence.

		Warping training		Morphing time	
Method	Steps	68 landmarks	130 landmarks	Res. 256^2	Res. 1350^2
OpenCV	–	–	–	0.01s	0.06s
ifmorph [44]	20k	05m26s	05m22s	0.05s	0.10s
NCF (Ours)	20k	08m48s	15m05s	0.02s	0.31s
NODE (Ours)	2k	00m16s	00m15s	0.03s	1.48s
NODE (Ours)	20k	02m39s	02m38s	0.03s	1.48s

Figure 6 shows the application of FLOWING to non-aligned, in-the-wild face images. Using the NODE backbone for warping and DiffMorpher for blending, our method produces morphings that are both visually coherent and perceptually realistic. Additional examples of generative morphing results are provided in Appendix D.

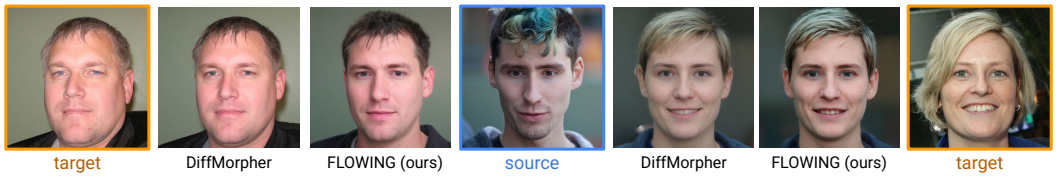

Figure 6: Qualitative results on in-the-wild faces from the FFHQ dataset [18]. By applying our flow-based warping prior to generative blending, FLOWING produces morphings that are significantly more coherent and realistic than those generated by DiffMorpher.

4.3 3D face morphing using Gaussian splatting

In this section, we evaluate our 3D face morphing framework using GaussianAvatars [39], an extension of 3DGS for photorealistic human head representation. Each face in GaussianAvatars is associated with a FLAME model [24], which enables the extraction of 3D facial landmarks spatially aligned with the Gaussian distribution. To morph between two GaussianAvatars, we apply the warping and blending formulation defined in (8). Figure 7 provides an overview of FLOWING applied to 3DGS morphing. On the left, we visualize the 3D flow field between two subjects derived from their landmarks and illustrate how the warped Gaussians can be combined to form an intermediate representation at $t = 0.5$, blending structural and appearance features from both faces. The middle panel shows additional morphing results for different subjects across multiple time steps. Our method achieves smooth and geometrically consistent transitions between subjects with distinct facial structures and appearance attributes. In particular, it naturally handles complex variations such as hair (second and third rows), thanks to the volumetric nature of 3DGS.

Finally, the right panel compares different blending strategies. The top row corresponds to a purely 2D setting, where warping is applied to the rendered images followed by linear blending, which results in severe misalignments and ghosting artifacts. In contrast, the middle and bottom rows show our 3D warping results, first with linear blending of the rendered images, and then with direct blending of the 3D Gaussians. The fully 3D pipeline produces smoother and more coherent transitions, effectively eliminating ghosting and preserving fine structural details across viewpoints. Additional qualitative examples of 3D morphing results are presented in Appendix C.

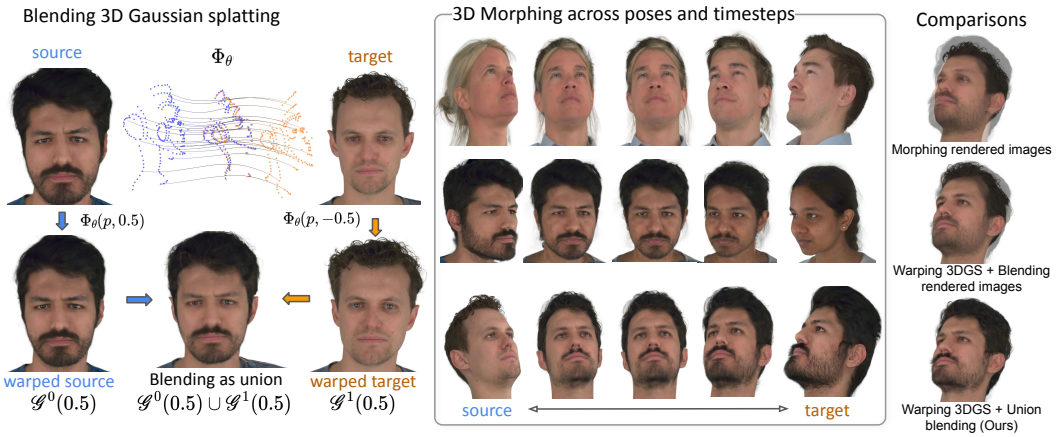

Figure 7: Overview of FLOWING for morphing between 3D faces using Gaussian splatting. Our approach warps the space to enforce 3D landmark alignment and applies union-based 3DGS blending, directly combining Gaussians in 3D space (left). This yields geometrically consistent morphs and preserves photorealistic appearance across poses and viewpoints (middle). The right panel compares blending strategies, showing that our union-based 3DGS blending produces smoother and more coherent results than morphing rendered images or blending after warping.

5 Conclusion and limitations

This work introduced FLOWING, a novel framework for morphing between graphical objects using flow-structured INRs. By leveraging the mathematical properties of flows, our approach enables faster convergence, more stable training, and robust warping behavior that avoids catastrophic singularities. We demonstrated the versatility of FLOWING across a wide range of tasks—including face morphing, view interpolation, and Gaussian-based 3D morphing—achieving high-quality and temporally coherent results. Moreover, FLOWING can be seamlessly integrated into generative pipelines as a replacement for traditional alignment procedures, producing morphings comparable to state-of-the-art generative approaches.

Despite these advantages, FLOWING inherits certain limitations intrinsic to flow-based formulations. Since it enforces invertibility by construction, it cannot model transformations involving occlusions or topological changes, such as morphing between faces with open and closed mouths. In such cases, generative models can complement our approach during the blending stage to recover missing structures. Additionally, as in other warping-based techniques, FLOWING depends on the quality and accuracy of landmark correspondences, whether extracted manually or through automated detectors.

As future work, we plan to extend FLOWING to point cloud alignment and registration, and further improve our Gaussian morphing approach.

Acknowledgments

Guilherme and Nuno would like to thank Fundação de Ciência e Tecnologia (FCT) projects UIDB/00048/2020[2] and UIDP/00048/2020 for partially funding this work. Guilherme would also like to thank FCT project 2024.07681.IACDC[3] for partially funding this work. João Paulo would like to thank Fundação Carlos Chagas Filho de Amparo à Pesquisa do Estado do Rio de Janeiro (FAPERJ) grant SEI-260003/012808/2024 for funding this work. Vitor and Daniel gratefully acknowledge support from CAPES, grants 88887.842584/2023-00 and 88887.832821/2023-00, respectively for supporting this research. João Pereira is thankful for a start-up grant from the University of Georgia. We also thank Google for funding this research.

[2]DOI: `https://doi.org/10.54499/UIDB/00048/2020`
[3]DOI: `https://doi.org/10.54499/2024.07681.IACDC`

References

[1] Ivan Anokhin, Kirill Demochkin, Taras Khakhulin, Gleb Sterkin, Victor Lempitsky, and Denis Korzhenkov. Image generators with conditionally-independent pixel synthesis. In *CVPR*, pages 14278–14287, 2021.

[2] Guha Balakrishnan, Amy Zhao, Mert R Sabuncu, John Guttag, and Adrian V Dalca. Voxelmorph: a learning framework for deformable medical image registration. *IEEE transactions on medical imaging*, 2019.

[3] Jonathan T. Barron, Ben Mildenhall, Dor Verbin, Pratul P. Srinivasan, and Peter Hedman. Mip-nerf 360: Unbounded anti-aliased neural radiance fields. *CVPR*, 2022.

[4] Thaddeus Beier and Shawn Neely. Feature-based image metamorphosis. *ACM SIGGRAPH computer graphics*, 26(2):35–42, 1992.

[5] Arthur Bizzi, Leonardo M Moreira, Márcio Marques, Leonardo Mendonça, Christian Júnior de Oliveira, Vitor Balestro, Lucas dos Santos Fernandez, Daniel Yukimura, Pavel Petrov, João Pereira, Tiago Novello, and Nissenbaum Lucas. Neuro-spectral architectures for causal physics-informed networks. *arXiv preprint arXiv:2509.04966*, 2025.

[6] Arthur Bizzi, Lucas Nissenbaum, and João M. Pereira. Neural conjugate flows: A physics-informed architecture with flow structure. *Proceedings of the AAAI Conference on Artificial Intelligence*, 39(15):15576–15586, Apr. 2025. doi: 10.1609/aaai.v39i15.33710. URL `https://ojs.aaai.org/index.php/AAAI/article/view/33710`.

[7] Fred L. Bookstein. Principal warps: Thin-plate splines and the decomposition of deformations. *IEEE Transactions on pattern analysis and machine intelligence*, 11(6):567–585, 1989.

[8] Daniel J Butler, Jonas Wulff, Garrett B Stanley, and Michael J Black. A naturalistic open source movie for optical flow evaluation. In *Computer Vision–ECCV 2012: 12th European Conference on Computer Vision, Florence, Italy, October 7-13, 2012, Proceedings, Part VI 12*, pages 611–625. Springer, 2012.

[9] Ricky TQ Chen, Yulia Rubanova, Jesse Bettencourt, and David K Duvenaud. Neural ordinary differential equations. *Neurips*, 31, 2018.

[10] Lisa DeBruine and Benedict Jones. Face research lab london set, May 2017.

[11] Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. Density estimation using real nvp. In *International Conference on Learning Representations*, 2017.

[12] Jonas Gomes, Lucia Darsa, Bruno Costa, and Luiz Velho. *Warping & morphing of graphical objects*. Morgan Kaufmann, 1999.

[13] Marcel Grimmer and Christoph Busch. Ladimo: Face morph generation through biometric template inversion with latent diffusion, 2024. URL `https://arxiv.org/abs/2410.07988`.

[14] Amos Gropp, Lior Yariv, Niv Haim, Matan Atzmon, and Yaron Lipman. Implicit geometric regularization for learning shapes. In *37th International Conference on Machine Learning, ICML 2020*, pages 3747–3757, 2020.

[15] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016. doi: 10.1109/CVPR.2016.90.

[16] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. GANs trained by a two time-scale update rule converge to a local nash equilibrium. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Neurips*, volume 30. Curran Associates, Inc., 2017. URL `https://proceedings.neurips.cc/paper_files/paper/2017/file/8a1d694707eb0fefe65871369074926d-Paper.pdf`.

[17] Zhewei Huang, Tianyuan Zhang, Wen Heng, Boxin Shi, and Shuchang Zhou. Real-time intermediate flow estimation for video frame interpolation. In *European Conference on Computer Vision*, pages 624–642. Springer, 2022.

[18] Tero Karras, Samuli Laine, Miika Aittala, Janne Hellsten, Jaakko Lehtinen, and Timo Aila. Analyzing and improving the image quality of stylegan. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8107–8116. Computer Vision Foundation / IEEE, 2020. doi: 10.1109/CVPR42600.2020.00813.

[19] Vahid Kazemi and Josephine Sullivan. One millisecond face alignment with an ensemble of regression trees. In *CVPR*, June 2014.

[20] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3d gaussian splatting for real-time radiance field rendering. *ACM Transactions on Graphics*, 42(4), July 2023. URL `https://repo-sam.inria.fr/fungraph/3d-gaussian-splatting/`.

[21] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2017. URL `https://arxiv.org/abs/1412.6980`.

[22] Tobias Kirschstein, Shenhan Qian, Simon Giebenhain, Tim Walter, and Matthias Nießner. Nersemble: Multi-view radiance field reconstruction of human heads. *ACM Transactions on Graphics (TOG)*, 42(4):1–14, 2023.

[23] Lingtong Kong, Boyuan Jiang, Donghao Luo, Wenqing Chu, Xiaoming Huang, Ying Tai, Chengjie Wang, and Jie Yang. Ifrnet: Intermediate feature refine network for efficient frame interpolation. In *CVPR*, pages 1969–1978, 2022.

[24] Tianye Li, Timo Bolkart, Michael J Black, Hao Li, and Javier Romero. Learning a model of facial shape and expression from 4d scans. *ACM Trans. Graph.*, 36(6):194–1, 2017.

[25] Zhengqi Li and Noah Snavely. Megadepth: Learning single-view depth prediction from internet photos. In *CVPR*, pages 2041–2050, 2018.

[26] Jing Liao, Rodolfo S Lima, Diego Nehab, Hugues Hoppe, Pedro V Sander, and Jinhui Yu. Automating image morphing using structural similarity on a halfway domain. *ACM Transactions on Graphics (TOG)*, 33(5):1–12, 2014.

[27] Ishit Mehta, Manmohan Chandraker, and Ravi Ramamoorthi. A level set theory for neural implicit evolution under explicit flows. In *Computer Vision – ECCV 2022*, pages 711–729, Cham, 2022. Springer Nature Switzerland. ISBN 978-3-031-20086-1.

[28] Ben Mildenhall, Pratul P Srinivasan, Matthew Tancik, Jonathan T Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. In *European Conference on Computer Vision*, pages 405–421. Springer, 2020.

[29] Richard D Neidinger. Introduction to automatic differentiation and matlab object-oriented programming. *SIAM review*, 52(3):545–563, 2010.

[30] Tiago Novello, Guilherme Schardong, Luiz Schirmer, Vinícius Da Silva, Hélio Lopes, and Luiz Velho. Exploring differential geometry in neural implicits. *Computers & Graphics*, 108:49–60, 2022.

[31] Tiago Novello, Vinicius Da Silva, Guilherme Schardong, Luiz Schirmer, Helio Lopes, and Luiz Velho. Neural implicit surface evolution. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 14279–14289, 2023.

[32] Tiago Novello, Diana Aldana, Andre Araujo, and Luiz Velho. Tuning the frequencies: Robust training for sinusoidal neural networks. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 3071–3080, 2025.

[33] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zach DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library, 2019. URL `https://arxiv.org/abs/1912.01703`.

[34] Hallison Paz, Daniel Perazzo, Tiago Novello, Guilherme Schardong, Luiz Schirmer, Vinicius da Silva, Daniel Yukimura, Fabio Chagas, Helio Lopes, and Luiz Velho. Mr-net: Multiresolution sinusoidal neural networks. *Computers & Graphics*, 2023.

[35] Hallison Paz, Tiago Novello, and Luiz Velho. Spectral periodic networks for neural rendering. In *ACM SIGGRAPH 2024 Posters*, pages 1–2. 2024.

[36] Vitor Pereira Matias, Daniel Perazzo, Vinicius Silva, Alberto Raposo, Luiz Velho, Afonso Paiva, and Tiago Novello. From volume rendering to 3d gaussian splatting: Theory and applications. In *SIBGRAPI Conference on Graphics, Patterns and Images*, 2025.

[37] Patrick Pérez, Michel Gangnet, and Andrew Blake. Poisson image editing. In *Seminal Graphics Papers: Pushing the Boundaries, Volume 2*, pages 577–582. 2023.

[38] Guilherme Potje, Felipe Cadar, André Araujo, Renato Martins, and Erickson R Nascimento. Xfeat: Accelerated features for lightweight image matching. In *CVPR*, pages 2682–2691, 2024.

[39] Shenhan Qian, Tobias Kirschstein, Liam Schoneveld, Davide Davoli, Simon Giebenhain, and Matthias Nießner. GaussianAvatars: Photorealistic head avatars with rigged 3d gaussians, 2024. URL `https://arxiv.org/abs/2312.02069`.

[40] Christos Sagonas, Epameinondas Antonakos, Georgios Tzimiropoulos, Stefanos Zafeiriou, and Maja Pantic. 300 faces in-the-wild challenge: database and results. *Image and Vision Computing*, 47:3–18, 2016. ISSN 0262-8856. doi: https://doi.org/10.1016/j.imavis.2016.01.002. URL `https://www.sciencedirect.com/science/article/pii/S0262885616000147`. 300-W, the First Automatic Facial Landmark Detection in-the-Wild Challenge.

[41] Lu Sang, Zehranaz Canfes, Dongliang Cao, Florian Bernard, and Daniel Cremers. Implicit neural surface deformation with explicit velocity fields, 2025. URL `https://arxiv.org/abs/2501.14038`.

[42] Lu Sang, Zehranaz Canfes, Dongliang Cao, Riccardo Marin, Florian Bernard, and Daniel Cremers. 4deform: Neural surface deformation for robust shape interpolation. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, 2025.

[43] Eklavya Sarkar, Pavel Korshunov, Laurent Colbois, and Sébastien Marcel. Vulnerability analysis of face morphing attacks from landmarks and generative adversarial networks, October 2020. URL `https://arxiv.org/abs/2012.05344`.

[44] Guilherme Schardong, Tiago Novello, Hallison Paz, Iurii Medvedev, Vinícius Da Silva, Luiz Velho, and Nuno Gonçalves. Neural implicit morphing of face images. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7321–7330, 2024.

[45] Luiz Schirmer, Tiago Novello, Vinícius da Silva, Guilherme Schardong, Daniel Perazzo, Hélio Lopes, Nuno Gonçalves, and Luiz Velho. Geometric implicit neural representations for signed distance functions. *Computers & Graphics*, 125:104085, 2024.

[46] Jag Mohan Singh and Raghavendra Ramachandra. 3-d face morphing attacks: Generation, vulnerability and detection. *IEEE Transactions on Biometrics, Behavior, and Identity Science*, 6 (1):103–117, 2024. doi: 10.1109/TBIOM.2023.3324684.

[47] Vincent Sitzmann, Julien Martel, Alexander Bergman, David Lindell, and Gordon Wetzstein. Implicit neural representations with periodic activation functions. *Neurips*, 33, 2020.

[48] Douglas Smythe. A two-pass mesh warping algorithm for object transformation and image interpolation. Technical Report 1030, ILM Technical Memo, Computer Graphics Department, Lucasfilm Ltd., 1990.

[49] Olga Sorkine and Marc Alexa. As-rigid-as-possible surface modeling. In *Proceedings of the Fifth Eurographics Symposium on Geometry Processing*, SGP '07, page 109–116, Goslar, DEU, 2007. Eurographics Association. ISBN 9783905673463.

[50] Shanlin Sun, Kun Han, Deying Kong, Hao Tang, Xiangyi Yan, and Xiaohui Xie. Topology-preserving shape reconstruction and registration via neural diffeomorphic flow. In *CVPR*, 2022.

[51] Shanlin Sun, Kun Han, Chenyu You, Hao Tang, Deying Kong, Junayed Naushad, Xiangyi Yan, Haoyu Ma, Pooya Khosravi, James S Duncan, et al. Medical image registration via neural fields. *Medical Image Analysis*, 97:103249, 2024.

[52] Matthew Tancik, Pratul Srinivasan, Ben Mildenhall, Sara Fridovich-Keil, Nithin Raghavan, Utkarsh Singhal, Ravi Ramamoorthi, Jonathan Barron, and Ren Ng. Fourier features let networks learn high frequency functions in low dimensional domains. *Neurips*, 33:7537–7547, 2020.

[53] Zachary Teed and Jia Deng. Raft: Recurrent all-pairs field transforms for optical flow. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part II 16*, pages 402–419. Springer, 2020.

[54] Marcelo Viana and José M Espinar. *Differential equations: a dynamical systems approach to theory and practice*, volume 212. American Mathematical Society, 2021.

[55] George Wolberg. *Digital image warping*, volume 10662. IEEE computer society press Los Alamitos, CA, 1990.

[56] George Wolberg. Image morphing: a survey. *The visual computer*, 14(8-9):360–372, 1998.

[57] Yifan Wu, Tom Z Jiahao, Jiancong Wang, Paul A Yushkevich, M Ani Hsieh, and James C Gee. Nodeo: A neural ordinary differential equation based optimization framework for deformable image registration. In *CVPR*, 2022.

[58] Guandao Yang, Serge Belongie, Bharath Hariharan, and Vladlen Koltun. Geometry processing with neural fields. *Neurips*, 34:22483–22497, 2021.

[59] Kaiwen Zhang, Yifan Zhou, Xudong Xu, Bo Dai, and Xingang Pan. Diffmorpher: Unleashing the capability of diffusion models for image morphing. In *CVPR*, pages 7912–7921, 2024.

[60] Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *CVPR*, 2018.

A Forward differentiation

We implement forward differentiation (FD) to speed up the calculation of Jacobian or Hessian terms, for thin-plate regularization. Forward mode differentiation is an alternative way to implement automatic differentiation, opposite to backward (or reverse mode) differentiation. FD is typically implemented using tangent (or dual) numbers [29]. Let us illustrate FD with an example. Suppose we want to evaluate $\frac{\partial x_n}{\partial x_0}$, where

$$x_i = f_i(x_{i-1}), \quad i = 1, \dots, n. \tag{9}$$

Using the chain rule, we obtain that

$$\frac{\partial x_n}{\partial x_0} = \prod_{i=1}^{n} \frac{\partial x_i}{\partial x_{i-1}} = \prod_{i=1}^{n} f_i'(x_{i-1}).$$

To implement FD, we introduce the tangent (or dual) variables $\dot{x}_k$, $k = 0, \dots, n$, defined by the recursion $\dot{x}_0 = \frac{\partial x_0}{\partial x_0} = 1$, and

$$\dot{x}_i = f_i'(x_{i-1})\dot{x}_{i-1}, \quad i = 1, \dots, n \tag{10}$$

Using induction, it follows that

$$\dot{x}_k = f_k'(x_{k-1})\dot{x}_{k-1} = \prod_{i=1}^{k} f_i'(x_{i-1}) = \frac{\partial x_k}{\partial x_0}, \quad k = 1, \dots, n.$$

Finally, the last element yields the desired derivative $\dot{x}_n = \frac{\partial x_n}{\partial x_0}$. To implement this approach, we can replace every function f_i by an FD version, that calculates both x_i and $\dot{x}_i$ at the same time.

Effectively, the FD version of f_i maps ordered pairs $(x_{i-1}, \dot{x}_{i-1})$ to ordered pairs $(x_i, \dot{x}_i)$, so that both (9) and (10) are satisfied:

$$(x_i, \dot{x}_i) = f_i^{\text{FD}}(x_{i-1}, \dot{x}_{i-1}) := (f_i(x_{i-1}), f_i'(x_{i-1})\dot{x}_{i-1}), \quad i = 1, \dots, n$$

Since we need to calculate Hessians, we develop a multivariate FD implementation involving ordered triplets $(z, \dot{z}, \ddot{z})$. Here, $\dot{z}$ contains intermediary gradients, and $\ddot{z}$ contains intermediary Hessians, with respect to a variable x. More specifically, if $z \in \mathbb{R}^m$ and $x \in \mathbb{R}^n$, our implementation ensures that $\dot{z} = \nabla_x z \in \mathbb{R}^{m \times n}$ and $\ddot{z} = \text{Hess}_x(z) \in \mathbb{R}^{m \times n \times n}$.

To calculate the Hessian of the model $f_\theta(x)$ with respect to x, we first write it as a composition of simpler functions (linear operators, non-linear activations, flow operator of affine flows), for which we have implemented forward differentiation: $f_\theta = f_1 \circ f_2 \circ \cdots \circ f_m$. Then, noting that $\nabla_x x = \mathbf{I}_n$ (the $n \times n$ identity matrix) and $\text{Hess}_x(x) = \mathbf{0}_{n \times n \times n}$, we calculate

$$f_\theta^{\text{FD}}(x, \mathbf{I}_n, \mathbf{0}_{n \times n \times n}) = f_1^{\text{FD}} \circ \cdots \circ f_m^{\text{FD}}(x, \mathbf{I}_n, \mathbf{0}_{n \times n \times n}) = (u, \dot{u}, \ddot{u}),$$

and set $\text{Hess}_x(f_\theta(x)) = \ddot{u}$.

Followingly, we provide the details for the FD implementation of linear transformations, non-linear activations, and the flow operator of an affine flow. By composing these, we can construct all architectures we consider in the paper.

Linear operators: Consider the linear operator $\mathcal{A}\mathbf{z} \mapsto \mathbf{A}\mathbf{z} + \mathbf{b}$. Since derivatives are also linear operators, the FD implementation is straightforward.

$$\mathcal{A}^{\text{FD}}(\mathbf{z}, \dot{\mathbf{z}}, \ddot{\mathbf{z}}) = (\mathbf{A}\mathbf{z} + \mathbf{b},\ \mathbf{A}\dot{\mathbf{z}},\ \mathbf{A} \times_1 \ddot{\mathbf{z}}),$$

where $\mathbf{A}\times_1$ denotes multiplication by $\mathbf{A}$ in the first dimension.

Non-linear activations: Suppose that $z \in \mathbb{R}$, $g : \mathbb{R} \to \mathbb{R}$, $\dot{z} \in \mathbb{R}^n$ and $\ddot{z} \in \mathbb{R}^{n \times n}$. Then using the chain rule, one obtains:

$$\begin{aligned} g^{\text{FD}}(z, \dot{z}, \ddot{z}) &= (g(z),\ \nabla g(z),\ \text{Hess}(g(z))), \\ &= (g(z),\ g'(z)\dot{z},\ \nabla(g'(z)\dot{z})), \\ &= \left(g(z),\ g'(z)\dot{z},\ g'(z)\ddot{z} + g''(z)\dot{z}\dot{z}^T\right). \end{aligned}$$

Here, the terms $g(z), g'(z)$ and $g''(z)$ are the derivatives of the activation function. For instance, for SIRENs, we have $g(z) = \frac{1}{w_0}\sin(w_0 z)$, $g'(z) = \cos(w_0 z)$ and $g''(z) = -w_0 \sin(w_0 z) = -w_0^2 g(z)$.

Flow operator of an Affine Flow: The formula for an affine flow is provided in [6].

$$\Phi(\mathbf{x}, t) = e^{\mathbf{A}t}\mathbf{x} + \int_0^t e^{\mathbf{A}\tau}\mathbf{b}\, d\tau,$$

where the affine integral term is calculated exactly using an augmentation trick [6]. This flow has some nice properties: the map is linear on $\mathbf{x}$ and the derivatives of the exponential in terms of t are easier to calculate. Suppose we want to evaluate $\Phi_{\text{FD}}(\langle \mathbf{x}, \dot{\mathbf{x}}, \ddot{\mathbf{x}} \rangle, \langle t, \dot{t}, \ddot{t} \rangle)$, where $\mathbf{x} \in \mathbb{R}^d$, $\dot{\mathbf{x}} \in \mathbb{R}^{d \times n}$, $\ddot{\mathbf{x}} \in \mathbb{R}^{d \times n \times n}$, $t \in \mathbb{R}$, $\dot{t} \in \mathbb{R}^n$ and $\ddot{t} \in \mathbb{R}^{n \times n}$. Letting $\mathbf{f} = \Phi(\mathbf{x}, t)$, $\dot{\mathbf{f}} = \mathbf{A}\mathbf{f} + \mathbf{b}$ and $\mathbf{E} = e^{\mathbf{A}t}$, we obtain, using the chain rule:

$$\begin{aligned} \Phi^{\text{FD}}\left((\mathbf{x}, \dot{\mathbf{x}}, \ddot{\mathbf{x}}), (t, \dot{t}, \ddot{t})\right) &= \left(\Phi(\mathbf{x}, t),\ \nabla_x \Phi\, \dot{\mathbf{x}} + \Phi_t \dot{t}^T,\ \nabla_x^2 \Phi\, \ddot{\mathbf{x}} + \nabla_x \Phi_t \otimes \dot{t} + \Phi_{tt} \otimes \ddot{t}\right), \\ &= \left(\mathbf{f},\ \mathbf{E}\dot{\mathbf{x}} + \dot{\mathbf{f}}\dot{t}^T, \mathbf{E} \times_1 \ddot{\mathbf{x}} + (\mathbf{A}\mathbf{E}\dot{\mathbf{x}}) \otimes \dot{t} + (\mathbf{A}\dot{\mathbf{f}}) \otimes \ddot{t}\right). \end{aligned}$$

B Ablation studies

NODE integration steps. Neural ODEs require solving differential equations through numerical integration, both during training and inference, to associate each point with its corresponding source and target coordinates by integrating forward and backward in time. Thus, it might appear that the trajectories need to be approximated using a large number of steps, making the process computationally expensive. However, we find that this is not the case: high-quality solutions can

be obtained with only a few steps. We sampled several initial conditions from the $[-1, 1]^2$ grid and displaced the points using the trained NODE model with the same number of integration steps used during training. We then approximated the reference (ground-truth) trajectories by computing a high-resolution baseline with 1000 integration steps. As a quality metric, we measured the maximum squared error between the predicted trajectory points and the linearly interpolated baseline points; a lower error indicates that the numerical integration used during training accurately captures the underlying dynamics. Table 4 shows the results of this experiment. We observe that only a small number of integration steps is sufficient to approximate the flow with high fidelity. Moreover, applying Jacobian regularization during training further stabilizes the dynamics, enabling reliable results even with very few integration steps.

Table 4: Comparison of trajectory quality (maximum squared error) of NODEs with different integration steps against a baseline of 1000 steps. NODE-based morphing needs very few steps in order to obtain high quality approximations of the associated flow.

Int. steps	MegaDepth Sample		FRLL Sample	
	No regularization	With regularization	No regularization	With regularization
3	5.44E-05	6.43E-07	1.36E-07	7.16E-10
5	3.39E-08	3.27E-09	2.08E-10	6.19E-11
7	1.88E-09	5.24E-10	5.46E-11	6.00E-11
15	8.88E-12	5.18E-10	5.46E-11	6.10E-11

Forward differentiation (FD). We conducted an experiment to evaluate the computational speed-up achieved by our FD scheme for Hessian computation. An NCF model was initialized using the same configuration as in our main experiments. For this model, we computed the Hessian at 1,000 domain points using both PyTorch's `autograd` and our FD implementation, then evaluated the thin-plate loss and performed backpropagation. Each method was run 100 times, and we report the average computation time per iteration. On an NVIDIA GeForce RTX 4090 GPU, our FD implementation achieved an average iteration time of 6.93×10^{-3} seconds, compared to 1.62×10^{-1} seconds using `autograd`, corresponding to a $23.4\times$ speed-up. This demonstrates the substantial efficiency gains of our FD approach over standard automatic differentiation.

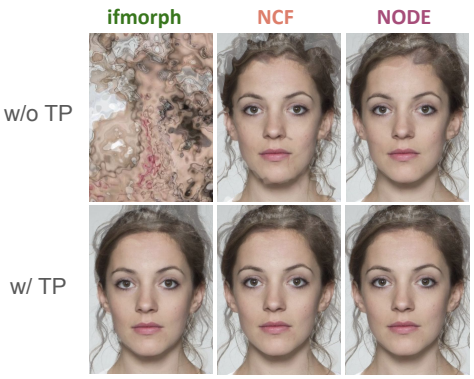

Figure 8: Effect of TP regularization on ifmorph (left) and our methods, NCF (center) and NODE (right). The top row shows results without TP regularization, while the bottom row shows the same models with TP applied. TP regularization promotes smoother deformations and enhances structural coherence across the morphing process.

Thin-plate regularization. We perform an ablation study to evaluate the effect of thin-plate (TP) regularization on deformation smoothness and structural consistency. As illustrated in Figure 8, TP regularization significantly improves the coherence of the deformation field, reducing distortions and enforcing smoother transitions between source and target landmarks. The effect is most pronounced for the baseline ifmorph, which exhibits severe artifacts in the absence of regularization. In contrast, our models (NCF and NODE) already demonstrate stable deformation behavior due to their flow-based formulation, yet still benefit from TP regularization, yielding even more consistent and visually coherent morphings.

C Additional 3DGS morphing experiments

Flow evaluation Figure 9 illustrates the 3D flow streamlines corresponding to landmark trajectories from the source image for three methods used in 3DGS morphing: ifmorph, and FLOWING with NCF and NODE backbones. As shown, ifmorph produces flow fields that are less stable and exhibit pronounced curvature, while NCF and NODE yield smoother, more coherent, and regularized flows.

This difference is particularly evident in regions not directly constrained by landmarks—such as the hair—where ifmorph introduces noticeable distortions. In contrast, the FLOWING variants preserve both structural continuity and spatial consistency across the field.

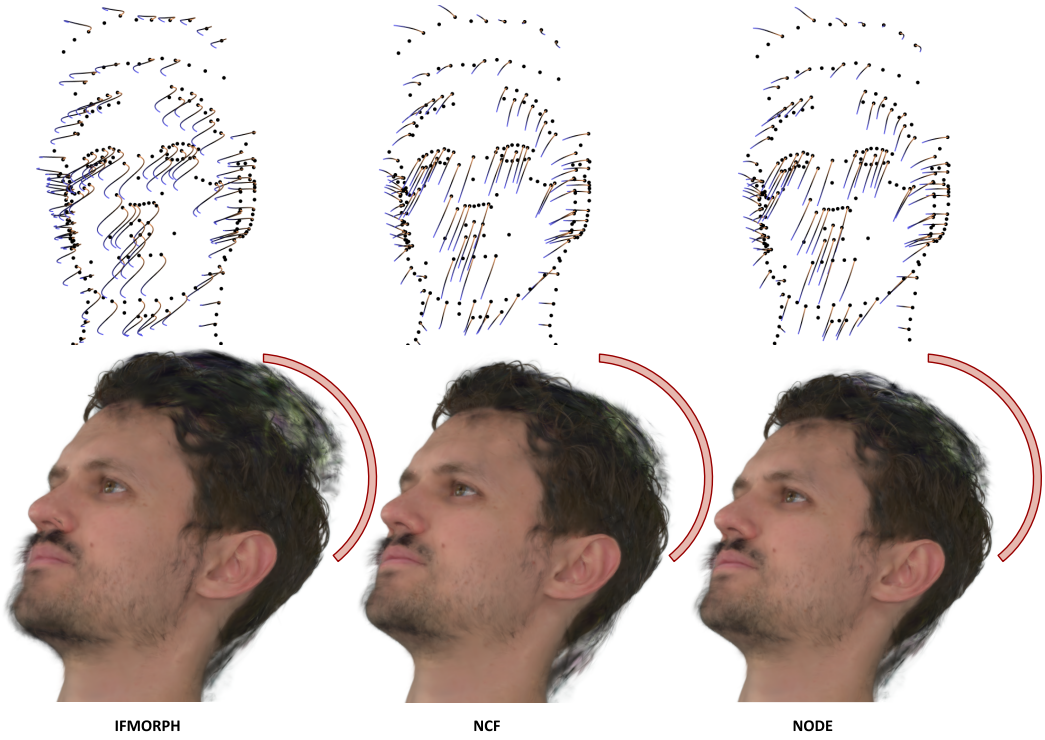

Figure 9: Visualization of 3D flow fields generated by ifmorph, NCF, and NODE in the 3DGS morphing task. The red arch highlights instability in the hair region for ifmorph, whereas NCF and NODE preserve structure more reliably.

Training details Our 3DGS morphing configuration extends the 2D morphing pipeline while maintaining the same loss weights. To accommodate the higher dimensionality, we use slightly larger learning rates (LRs) and an LR scheduler for both NODE and NCF to ensure convergence within 20,000 steps. The initial LRs are set to 0.001 for NODE, 0.002 for NCF, and 0.0001 for ifmorph. Early stopping is employed with a patience of 500 epochs for NCF and NODE, and 1,000 for ifmorph. If the loss plateaus for 100 epochs, the LR is reduced, and training terminates once the patience threshold is reached. The best-performing model is selected based on the lowest loss.

Additional results Figure 10 presents additional qualitative results for our Gaussian morphing framework. FLOWING consistently achieves smooth and coherent transitions across diverse subjects, effectively handling challenging regions such as facial and head hair (third row). Furthermore, it demonstrates strong generalization when morphing between subjects of different genders, maintaining both geometric structure and photometric consistency.

D Additional comparisons using generative blending

Figure 11 shows additional qualitative results combining FLOWING with DiffMorpher as a generative blending strategy. These examples further confirm that introducing a warping stage prior to generative blending significantly improves the quality of the final morphs. In particular, FLOWING enhances spatial alignment and structural coherence, producing smoother intermediate representations, even in simple scenarios such as morphing between spherical objects.

Figure 10: Additional results of Gaussian morphing across multiple timesteps (first and last columns show the targets). FLOWING achieves smooth and coherent transitions across diverse subjects while preserving structure and appearance.

Figure 11: Additional examples comparing generative blending with and without FLOWING as a warping prior. Incorporating FLOWING before applying DiffMorpher (NODE+DiffMorpher, ours) yields superior structural preservation and smoother transitions.

E Applications in video interpolation

Video interpolation aims to synthesize intermediate frames between two given frames, a key component in applications such as video compression and frame rate upscaling. Most deep learning-based approaches rely on estimating optical flow between consecutive frames, which describes pixel motion as a vector field [17; 23]. Assuming temporal proximity between frames, optical flow can be leveraged to approximate intermediate frames by integrating a time-independent ODE defined by the flow field.

Inspired by this principle, we explore the use of FLOWING to learn a compact representation of optical flow from sparse samples and to reconstruct intermediate frames. For this experiment, we employ a cropped sequence from the Sintel dataset [8] and use RAFT [53] to generate dense optical flow between frames. We then sample a sparse set of image points and construct landmark pairs (p^0, p^1) by displacing them according to the optical flow. FLOWING is trained to model a continuous flow field aligning these points over time. To evaluate interpolation quality, we warp and linearly blend the input frames at $t = 0.5$ using the learned flow and compare the resulting image against the ground-truth middle frame. We further compare against three baselines: (1) linear frame blending, (2) dense optical-flow warping followed by linear blending, and (3) ifmorph [44]. As shown in Table 5, FLOWING achieves lower MSE than all baseline interpolation methods, indicating improved accuracy in predicting intermediate frames. These preliminary results suggest that learning a continuous optical flow representation from sparse correspondences is a promising direction for video interpolation.

Table 5: Comparison of interpolation accuracy against the ground-truth intermediate frame on Sintel [8]. Lower MSE indicates better reconstruction.

Method	MSE (↓)
Linear Blend	2.6E-4
Optical Flow Warp	4.9E-5
ifmorph	5.8E-5
NCF (Ours)	**4.0E-5**
NODE (Ours)	**3.9E-5**

F Comparison with TPS warping

Another standard morphing algorithm is thin-plate-spline warping. While TPS warping does not involve an iterative process, the method does require solving a linear system of equations in order to find the appropriate linear combination of the basis functions. The matrix of such a system can, in some cases, be badly conditioned. When using the scikit-image implementation of the algorithm, we find that this problem leads to crashes or low-quality warpings in approximately 10% of FRLL samples. This can negatively impact metrics such as the LPIPS. FLOWING, on the other hand, does not present these problems as it does not rely on solving a similar badly-conditioned system. For comparison, using TPS warping combined with linear blending for the FRLL dataset, we obtain $\text{LPIPS}(I^0, I) = 0.371$ and $\text{LPIPS}(I^1, I) = 0.374$.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: We have described the method and performed experiments to demonstrate the paper's contributions.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: We have testes our method presenting ablation studies on its parameters to show some limitations.

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory Assumptions and Proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [Yes]

Justification: We present a clear statement of our definitions with experiments to motivate them.

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental Result Reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: We presented all the details and, in case of the paper being accepted, we'll make available all experiments and code.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: We presented all the details and, in case of the paper being accepted, we'll make available all experiments and code.

Guidelines:

- The answer NA means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental Setting/Details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

   Answer: [Yes]

   Justification: For all experiments we have presented the detailed information about relevant hyperparameters and evaluation.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment Statistical Significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: We presented statistical information in our main results.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).

- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments Compute Resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: The information related is given at beginning of Section 4.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code Of Ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: We read the Code of Ethics.

   Guidelines:

   - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader Impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [NA]

    Justification: We mostly deal with theoretical and technical issues of implicit neural networks.

    Guidelines:

    - The answer NA means that there is no societal impact of the work performed.
    - If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.

- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

    Answer: [NA]

    Justification: Our work poses no such risks.

    Guidelines:

    - The answer NA means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: We reference the authors of the code used in Section 4.

    Guidelines:

    - The answer NA means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.

- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New Assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [NA]

    Justification: Our work doesn't release new assets.

    Guidelines:

    - The answer NA means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and Research with Human Subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [NA]

    Justification: Our work doesn't involve crowdsourcing nor researach with human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional Review Board (IRB) Approvals or Equivalent for Research with Human Subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [NA]

    Justification: Our work doesn't involve crowdsourcing nor researach with human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.