DANCINGTHINGS: PROCEDURAL DATA GENERATION FOR OPTICAL FLOW WITH SPLINE-BASED DEFORMATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Existing optical flow methods do not perform well on non-rigid movement. We remedy this problem by introducing a procedural data generator focusing on non-rigid motion that can generate unlimited data for optical flow. Our generator takes a scene mesh as input and outputs an animated version of the scene where objects are deformed with a novel spline-based deformation method called SPD. SPD works by creating free-form deformation fields around the assets and animating them with 4D NURBS, where there is an extra time curve that controls how the control points move. We prove two theoretical results showing that our deformation method is expressive and general, making methods trained on our data robust to a wide range of real-world motion. We use our procedural generator to generate and render a sample-efficient optical flow dataset called DancingThings. To our knowledge, DancingThings is the first optical flow dataset that focuses on general non-rigid motion in contrast to previous works, which include only specific types of deformations such as human motion, cloth simulation, and soft-body dynamics. We show that optical flow methods trained on DancingThings achieve significantly higher accuracy on non-rigid motion compared to methods trained on standard datasets. In particular, DancingThings achieves higher accuracy on non-rigid benchmarks than FlyingChairs and TartanAir combined with **138× fewer image pairs**. Furthermore, DancingThings surprisingly improves performance on rigid-motion benchmarks as well, even outperforming rigid training datasets. All data and code will be open-sourced.

1 INTRODUCTION

Optical flow estimation is a fundamental 3D vision task where the input is a pair of successive frames and the output is a per-pixel 2D vector field that shows where each pixel at time t goes to at t+1. It is widely used in tasks like visual odometry/SLAM (Teed et al., 2023; Teed & Deng, 2021a; Kayan et al., 2025; Wimbauer et al., 2025), scene flow estimation (Teed & Deng, 2021b), stereo depth estimation (Lipson et al., 2021), frame interpolation (Huang et al., 2022), video inpainting (Zhang et al., 2022), and video depth estimation (Wang et al., 2023; Yasarla et al., 2023).

Despite the demand, it is difficult to collect real-world data for optical flow since it is cumbersome to annotate dense correspondences and since we lack the hardware that would provide high-quality ground-truth. Luckily, previous research suggests that optical flow generalizes well from synthetic data to real data (Lipson et al., 2021; Wang et al., 2024; Mayer et al., 2016; Teed & Deng, 2020).

However, current methods do not work well on non-rigid objects (Kiefhaber et al., 2025) even though this is a common case in everyday life. For instance, humans, soft materials such as cloth and plastic, hair-like objects (hair, foliage), and fluids all move in a non-rigid manner and are better modeled by deformations.

We hypothesize that the reason why these methods fail on such common cases is the lack of data that contains general non-rigid motion. For instance, the most commonly used optical flow datasets FlyingChairs (Dosovitskiy et al., 2015b) and FlyingThings3D (Mayer et al., 2016) do not contain any non-rigid motion. Sintel (Butler et al., 2012) and Spring (Mehl et al., 2023) contain some

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

Table 1: A comparison of DancingThings to existing optical flow datasets. DancingThings is the only dataset that contains a large amount and diversity of non-rigid motion. Despite being orders of magnitude smaller than popular optical flow datasets like FlyingChairs, FlyingThings, and TartanAir, it outperforms these datasets. DancingThings also includes rigid motion in contrast to existing work. Some figures are taken from the Spring (Mehl et al., 2023) paper.

Dataset	# Image Pairs	Photorealism	Source	Rigid Motion	Non-Rigid Motion	Occlusion Masks
MPI Sintel (Butler et al., 2012)	1593	high	synthetic	med	med	✓
Spring (Mehl et al., 2023)	5953	high	synthetic	med	med	✓
FlyingThings3D (Mayer et al., 2016)	24084	low	synthetic	med	none	✗
FlyingChairs (Dosovitskiy et al., 2015b)	22872	low	synthetic	med	none	✗
TartanAirWang et al. (2020)	306268	high	synthetic	low	low	✓
KITTI 2012 (Geiger et al., 2012)	389	N/A	real	none	none	✗
KITTI 2015 (Menze & Geiger, 2015)	400	N/A	real	low	low	✗
VIPER (Richter et al., 2017)	186285	high	synthetic	med	low	✗
Middlebury-OF (Baker et al., 2007)	16	high	syn/real	low	low	✗
VKITTI (Gaidon et al., 2016)	21210	low	synthetic	low	none	✗
VKITTI 2 (Cabon et al., 2020)	21210	med	synthetic	low	low	✗
HD1K (Kondermann et al., 2016)	1074	N/A	real	none	none	✗
Human OF (Ranjan et al., 2020)	238900	med	synthetic	low	low	✗
DeepDeform (Bozic et al., 2020)	5533	N/A	real	med	high	✓
DeformingThings4DLi et al. (2021)	122365	med	synthetic	low	high	✗
AutoFlow (Sun et al., 2021)	40000	low	synthetic	high	med	✗
DancingThings	**2376**	**high**	synthetic	**high**	**high**	✓

non-rigid motion, but they are limited in terms of the amount and the diversity of the motion, which is limited to animated characters, movement of hair, and articulations.

To overcome this problem, we introduce a procedural data generator for optical flow that can generate general non-rigid motion. Our generator starts with a pre-existing scene, such as one from Infinigen Indoors (Raistrick et al., 2023; 2024) (an indoor scene generator), animates the objects with an RRT*-based path planning algorithm (Karaman & Frazzoli, 2011; Raistrick et al., 2026), and deforms them randomly and procedurally with a novel algorithm that we call *Spline-based Procedural Deformation (SPD)*. The generator then renders a video from each scene and provides the successive frames with ground-truth optical flow as training data. It also provides occlusion masks that show which pixels have valid flow.

SPD first takes an object mesh as input and creates a free-form deformation (FFD) lattice (Sederberg & Parry, 1986) around it. These constitute the control points for the *space-time deformation field* instantiated by a 4D Non-Uniform Rational B-Splines (NURBS) that control the FFD lattice. At a given time slice, the deformation field corresponds to a volume specified by three spatial parameters. This volume is morphed across time by making each control point follow a NURBS trajectory over time. The input mesh is deformed by mapping each vertex to the NURBS parameters, and tracking them across time. In other words, we generate a procedural 4D NURBS model and track each vertex of the original mesh along its time dimension to generate procedural deformations.

Our deformations are expressive and can be thought of as a generalization of physically-plausible deformations. We provide some theoretical results showing why this is the case. Specifically, we prove in theorem 1 that any continuous physically-plausible deformation, like soft-body dynamics, can be approximated arbitrarily well by SPD deformations. We prove another result in theorem 2 showing that SPD can also approximate deformations that are continuous almost everywhere in the measure-theoretic sense, which includes cases like cutting food with a knife or tearing along a plane.

Our approach of creating synthetic training data is novel, especially the use of procedural non-rigid deformation. Previous work either did not involve procedural generation, or involved only randomized rigid motion. Because our approach is procedural, it can be applied to animate arbitrary 3D scenes, including existing ones, to create unlimited additional optical flow training data.

We use our procedural data generator to make DancingThings, a sample-efficient optical flow dataset that focuses on non-rigid motion. To our knowledge, DancingThings is the first optical flow dataset that includes general non-rigid motion and is not restricted to specific types of motion like soft-body dynamics, cloth simulation, etc. Fig. 1 shows a random sample of images from our dataset.

We show that DancingThings is much more sample-efficient than standard optical flow training sets like FlyingChairs (Dosovitskiy et al., 2015b) , FlyingThings (Mayer et al., 2016), and TartanAir (Wang et al., 2020); or deformation-based training sets like Autoflow (Sun et al., 2021) and De-



Figure 1: Randomly chosen images from DancingThings. The objects undergo both rigid motion and non-rigid motion induced by our SPD method. Objects are chosen with some probability to be deformed. Note that the background walls, floor, and ceiling are also deformed.

formingThings4D (Li et al., 2021). DancingThings outperforms existing training sets while being orders of magnitude smaller (see table 1). Furthermore, DancingThings does not hurt performance on rigid benchmarks either. On the contrary, it performs better than even rigid-motion training datasets when evaluated on KITTI (Menze & Geiger, 2015) and Spring Rigid (Mehl et al., 2023).

Our contributions are as follows:

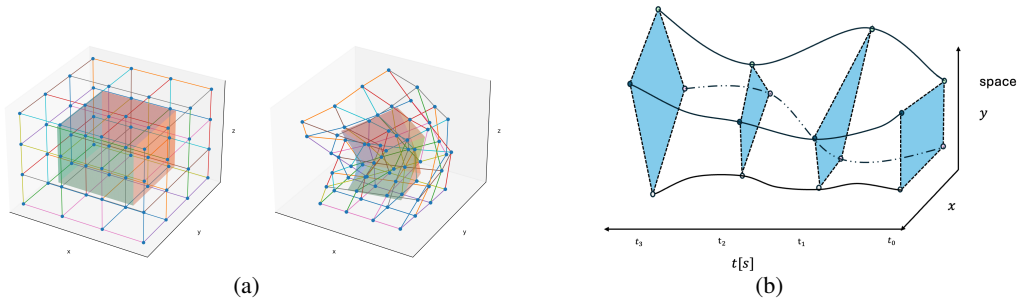
- A novel method called SPD that generates procedural deformations from arbitrary meshes.
- Theoretical results showing that SPD deformations are general and expressive.
- A procedural data generator for optical flow based on SPD deformations.
- A sample-efficient dataset called DancingThings, being the first dataset to focus on general non-rigid motion.

2 Related Work

Optical Flow Methods. Classical optical flow estimation methods (Zach et al., 2007; Pérez et al., 2013; Horn & Schunck, 1981; Lucas & Kanade, 1981) were based on optimization and the brightness constancy assumption. Nowadays, optical flow methods use neural networks (Wang et al., 2024; Teed & Deng, 2020; Wang & Deng, 2026; Deng et al., 2023; Dosovitskiy et al., 2015a; Jahedi et al., 2024b;a; Ilg et al., 2017; Hui et al., 2018; Leroy et al., 2024; Saxena et al., 2023; Luo et al., 2023; Shi et al., 2023; Sui et al., 2022; Sun et al., 2018; Xu et al., 2022) and train on synthetic datasets. As demonstrated in previous work (Kiefhaber et al., 2025; Li et al., 2025) and in experiments, current methods do not work well in non-rigid settings. The main reason for this drop in performance is the lack of training datasets that contain non-rigid motion and the lack of diversity in the datasets that do contain it. DancingThings remedies this issue by providing a sample-efficient dataset that contains objects deformed with SPD.

Optical Flow Datasets. Real-world datasets such as KITTI 2012 (Geiger et al., 2012), KITTI 2015 (Menze & Geiger, 2015), HD1K (Kondermann et al., 2016), and Middlebury (Baker et al., 2007) are small and generally used as benchmarks. Furthermore, they include negligible non-rigid motion. FlyingChairs (Dosovitskiy et al., 2015b) and FlyingThings3D (Mayer et al., 2016) are the most commonly used datasets that are used to train optical flow networks. However, the objects in these datasets only undergo only rigid motion where they fly around in the scene. Sintel (Butler et al., 2012) and Spring (Mehl et al., 2023) contain some non-rigid motion such as the movement of animated characters, simulations of plant growth, and hair simulations. However, the types of non-rigid motion are limited in these datasets. In contrast, DancingThings covers a wide range of non-rigid motion by using a spline-based deformation process. It includes scenes where every object is deformed including the background walls, floor, and the ceiling as shown in Fig. 1.

A related line of work are ones that propose non-rigid data for optical flow. AutoFlow (Sun et al., 2021) generates synthetic data by layering 2D images and warping them in 2D. Thus, it does not have an underlying 3D scene and cannot handle parallax as well as cases where the appearance of a point is view-dependent such as non-Lambertian objects. DeepDeform (Bozic et al., 2020)



(a) (b)

Figure 2: An illustration of SPD. As shown in (a), we initialize an FFD lattice around the bounding box of the input mesh. The FFD lattice consists of control points whose B-spline weighted combination gives the position of the mesh vertices across time. Thus, by moving the control points we gradually deform the mesh. Figure (b) shows how the control points move through time. Each control point has a trajectory determined by a time-indexed B-spline. We only show the x,y,t dimensions, although normally there would be a third spatial dimension.

relies on human annotations and cannot produce unlimited data. DeformingThings4D (Li et al., 2021) only includes animal and humanoid animations and does not have general deformations like DancingThings. It is also bottlenecked by data collection and thus not an unlimited data generator. In contrast, Infinigen Indoors (Raistrick et al., 2024) is an unlimited data generator for 3D vision. However, it only provides static scenes and does not animate objects rigidly or non-rigidly, whereas DancingThings animates scenes and produces a dataset.

Mesh Deformation Methods. There has been a lot of interest in the graphics community for mesh deformation methods. For instance, deformation transfer methods (Sumner & Popovic, 2004) take a source and target mesh as input and map the deformation of the source mesh onto the target mesh. Similarly, spacetime constraints (Witkin & Kass, 1988) takes a source mesh with a target deformation, and outputs the intermediate deformations that transform the source mesh into the target mesh. These methods are not ideal for procedural data generation since the range of deformations that can be generated is bottlenecked by the deformations that have already been generated. In contrast, our SPD can generate arbitrary deformations and its range is not limited by a library of deformations.

As-Rigid-As-Possible Surface Modeling (Sorkine & Alexa, 2007) has been proposed as a way of deforming a mesh while preserving its local rigidity. It moves a subset of initial vertices while making sure the rest of the mesh conforms to the constraints. In a similar vein, Free-Form Deformation (FFD) (Sederberg & Parry, 1986) was proposed as a way of deforming a mesh through a set of control points arranged in a lattice form. Our work also uses FFD lattices; However, our control points move in time through trajectories generated by a time-indexed B-spline, whereas the original FFD paper proposes moving the control points manually and also uses Bernstein polynomials instead.

D-NURBS (Terzopoulos & Qin, 1994) and Triangular NURBS (Qin & Terzopoulos, 1997) have been proposed as methods for creating trajectories for control points as well. However, the range of deformations that can be created by these methods is limited since they use dynamic physics simulations to move the control points. We show in section 4.4 that SPD is strictly more general than these formulations and not limited in the same way.

3 Background

Non-Uniform Rational B-Splines (NURBS) are a representation of curves and surfaces commonly used in computer graphics (Piegl & Tiller, 1997). A NURBS curve or surface is expressed as a weighted sum of control points, where the weights are B-spline basis functions defined over a knot vector, whose knots specify where the polynomial pieces meet.



Figure 3: Gradual deformation of an asset by SPD. Each row: rest pose at $t_0 \rightarrow$ deformation at $t_1 \rightarrow$ deformation at $t_2 \rightarrow$ deformation at t_3.

A NURBS curve is defined as:

$$\mathbf{C}(u) = \sum_{i=1}^{k} R_i(u)\,\mathbf{P}_i, \quad R_i(u) = \frac{N_{i,m}(u)\,w_i}{\sum_{j=1}^{k} N_{j,m}(u)\,w_j}, \quad U = \left\{ \underbrace{a, \ldots, a}_{m+1},\, u_{m+1}, \ldots, u_{p-m-1},\, \underbrace{b, \ldots, b}_{m+1} \right\} \tag{1}$$

where k is the number of control points $\mathbf{P}_i$, w_i are the weights, R_i are the rational basis functions, and $N_{i,m}$ is a B-spline of degree m defined on the nonperiodic knot vector U (Piegl & Tiller, 1997). By default, we let $a = 0, b = 1$ and $w_i > 0$ for every i.

A NURBS surface is a generalization of the 1D case and is defined by taking the tensor product of two 1D NURBS curves. Let u and v be independent parameters, then the surface is defined as $\mathbf{S}(u, v) = \sum_{i=1}^{k} \sum_{j=1}^{\ell} R_{i,j}(u, v)\,\mathbf{P}_{i,j}$ where $R_{i,j}(u, v) = \frac{N_{i,n}(u)\,N_{j,m}(v)\,w_{i,j}}{\sum_{p=1}^{k} \sum_{q=1}^{\ell} N_{p,n}(u)\,N_{q,m}(v)\,w_{p,q}}$.

4 METHOD

Our Spline-based Procedural Deformation (SPD) method takes a mesh as input and returns a gradual deformation of it, where the output is a deformed mesh for each timestep. We do this by generalizing the above discussion and using a three-parameter NURBS model for every time slice. This means that every time slice is modeled as a NURBS volume. We then add an additional B-Spline controlled by a time parameter, giving us a 4D NURBS model that we call *space-time deformation field*. Every point in the volume at t_0 can be tracked across time. This means that we can warp the vertices of the input mesh by tracking their transformation across time along the space-time deformation field.

An illustration of our method is given in fig. 2, which shows how the mesh is deformed via the NURBS control points. A gradual deformation on real assets is shown in fig. 3.

4.1 FFD LATTICE

Let $\mathbf{M} = \{\mathbf{x}_i\}_{i=1}^{n}$ be the vertices of an input mesh. We first compute the axis-aligned bounding box $\Omega \subset \mathbb{R}^3$ of the mesh. We then generate a lattice of control points $\mathbf{P}_{ijk} \in \mathbb{R}^3$ on this bounding box, where $i = 1, \ldots, n_u$, $j = 1, \ldots, n_v, k = 1, \ldots, n_w$. The ranges n_u, n_v, n_w are the numbers of control points per axis and determine the resolution of the lattice (Sederberg & Parry, 1986). Since we want the mesh vertices to be influenced by the control points, we need to work in lattice coordinates. We map each vertex $\mathbf{x} \in M$ from the bounding box to lattice coordinates by defining the coordinate transformation $\psi : \Omega \rightarrow [0, 1]^3$ with $\psi(\mathbf{x}) = (u, v, w)$. Here, ψ is the unique affine function that maps the corners of the box Ω to the corresponding corners of the unit cube $[0, 1]^3$. An illustration of the FFD lattice is given in fig. 2.

4.2 SPATIAL FFD

Sederberg and Parry (Sederberg & Parry, 1986) introduced Free-Form Deformation (FFD) lattices as a way of deforming solid geometric models. In the original paper, the lattice is used to define a tensor product trivariate Bernstein polynomial, where the original shape is deformed by manually moving the control points.

In our formulation, the FFD lattice induces a NURBS volume characterized by three parameters in lattice coordinates. Every vertex $\mathbf{x} \in \mathbf{M}$ corresponds to three coordinates $\psi(\mathbf{x}) = (u, v, w)$. Then, the position of the vertex is given by evaluating the NURBS volume

$$\mathbf{F}(\mathbf{x}) = \sum_{i=1}^{n_u} \sum_{j=1}^{n_v} \sum_{k=1}^{n_w} R_{i,j,k}(u,v,w)\, \mathbf{P}_{ijk}, \quad R_{i,j,k} := \frac{N_i(u)\, N_j(v)\, N_k(w)\, w_{i,j,k}}{\sum_{p=1}^{n_u} \sum_{q=1}^{n_v} \sum_{r=1}^{n_w} N_p(u)\, N_q(v)\, N_r(w)\, w_{p,q,r}}. \tag{2}$$

where $\mathbf{F}(\mathbf{x})$ is the spatial FFD, $R_{i,j,k}$ are the rational basis functions, and N_i, N_j, N_k are B-spline basis functions. Note that when control points are at their initial position, we have $\mathbf{F}(\mathbf{x}) = \mathbf{x}$, meaning that $\mathbf{F}$ is just identity. If the control points $\mathbf{P}_{ijk}$ are moved to a new position and $\mathbf{F}(\mathbf{x})$ is re-evaluated with the updated control points, then $\mathbf{F}(\mathbf{x}) = \mathbf{x}'$ gives the deformed position of the vertex $\mathbf{x}$.

4.3 Space-time deformation field

We define the trajectory of each control point as a B-spline with a time parameter. Mathematically,

$$\mathbf{P}_{ijk}(t) = \mathbf{P}_{ijk}(0) + \sum_{s=0}^{n_t-1} N_s(t)\, \mathbf{Q}_{ijks}, \tag{3}$$

where $\mathbf{P}_{ijk}(0)$ is the initial position of the lattice control points, $\mathbf{P}_{ijk}(t)$ is the position at time t, and $N_s(t)$ are B-spline basis functions with time parameter. $\mathbf{Q}_{ijks} \in \mathbb{R}^3$ are displacement control points which are sampled uniformly in a range $[-A, A]^3$, where A is an amplitude hyperparameter that controls the amount of deformation applied to the object.

Now that we have described the trajectory of lattice control points in time, we can re-evaluate the spatial FFD in eq. (2) at every timestep to get the trajectory of a vertex $\mathbf{x} \in \mathbf{M}$. Thus, the trajectory of the vertex is given by the *space-time deformation field*:

$$\mathbf{F}(\mathbf{x}, t) = \sum_{i,j,k} R_{i,j,k}(u,v,w)\, \mathbf{P}_{ijk}(t), \qquad (u,v,w) = \psi(\mathbf{x}). \tag{4}$$

In other words, the position of vertex $\mathbf{x}$ at time t is given by $\mathbf{x}(t) = \mathbf{F}(\mathbf{x}, t)$. eq. (4) just replaces the static lattice control points in eq. (2) with their positions at time t given by eq. (3).

Applying $\mathbf{F}(\mathbf{x}, t)$ to every vertex in the mesh gives us the deformed mesh at time t. Note that because the trajectories of the control points are smooth and geometrically continuous, the deformation of the mesh is also smooth and geometrically continuous.

One can alternatively formulate the space-time deformation field as a 4D-NURBS for properly defined control points $\widetilde{\mathbf{Q}}_{ijks}$:

$$\mathbf{F}(\mathbf{x}, t) = \sum_{i,j,k,s} R_{i,j,k,s}\, \widetilde{\mathbf{Q}}_{ijks}, \qquad R_{i,j,k,s} := \frac{N_i(u) N_j(v) N_k(w) N_s(t) w_{i,j,k,s}}{\sum_{p,q,r,b} N_p(u) N_q(v) N_r(w) N_b(t) w_{p,q,r,b}}. \tag{5}$$

4.4 Expressivity and Generality of SPD

We make minimal assumptions when designing SPD in order to include any type of motion that human vision can handle. This is because physically unrealistic motion that human vision can handle provides a useful signal to the models and makes them robust against outliers. For instance, humans are able to perceive a chair being squished like dough or falling up against gravity.

Even though our deformations are not guaranteed to be physically realistic, they can approximate any physically realistic deformation arbitrarily well, which means that they can be thought of as a generalization of physically realistic deformations. We prove the following theorem showing this is the case where we characterize physically plausible deformations as continuous functions $f : \mathbf{K} \to \mathbb{R}^3$, where $\mathbf{K} = I_1 \times I_2 \times I_3 \times I_4$ is a product of closed intervals in $\mathbb{R}^4$. We discuss this characterization in section B in more detail.

Theorem 1. *Let $\mathbf{K} = I_x \times I_y \times I_z \times I_t$ be a product of closed intervals in $\mathbb{R}^4$. Given any continuous function $f : \mathbf{K} \to \mathbb{R}^3$ and any $\varepsilon > 0$, there exists an SPD $\mathbf{S} : \mathbf{K} \to \mathbb{R}^3$ such that $\|f - S\|_\infty < \varepsilon$.*

Proof Sketch. We write $f = (f_1, f_2, f_3)$ in terms of its continuous components. Since $\mathbf{K}$ is compact, we can apply the Stone-Weierstrass approximation theorem and approximate each component with a polynomial P_i such that $\|f_i - P_i\|_\infty < \varepsilon/\sqrt{3}$. Defining $P = (P_1, P_2, P_3)$, it can be shown that $\|f - P\|_\infty < \varepsilon$, meaning that we can estimate our target deformation with a polynomial map. Intuitively, the rest of the proof follows from the property that polynomials are a special case of B-Splines, which are a special case of NURBS. The SPD $\mathbf{S}$ can be explicitly contructed by writing $P_i(x, y, z, t) = \sum_{jkls} w^i_{jkls} N_j(x) N_k(y) N_l(z) N_s(t)$ in a B-Spline basis and using eq. (5) with the control points constructed from the weights w^i_{jkls}. See section B for the full proof. □

Informally, theorem 1 means that given any physically plausible deformation f, we can find an SPD S arbitrarily close to it in the supremum-norm.

Now, as we discuss in section B, theorem 1 assumes that physically plausible deformations do not contain tearing or other non-topology preserving operations. The following theorem says that if f is an otherwise physically plausible deformation that contains tearing, SPD can still approximate it arbitrarily well, albeit in a weaker norm. The assumption we make is that the space along which tearing takes place has measure zero, which is the case, for instance, when cutting food with a knife.

Theorem 2. *Let $\mathbf{K} = I_x \times I_y \times I_z \times I_t$ be a product of closed intervals in $\mathbb{R}^4$ and $1 \leq p < \infty$. Suppose $f : \mathbf{K} \to \mathbb{R}^3$ is bounded and continuous almost everywhere. Then, for any $\varepsilon > 0$, there exists an SPD $S : \mathbf{K} \to \mathbb{R}^3$ such that $\|f - S\|_p < \varepsilon$ holds.*

Proof Sketch. It can be shown using measure theory that there exists a continuous function $g : \mathbf{K} \to \mathbb{R}^3$ that is arbitrarily close to f, i.e. $\|f - g\|_p$ can be made arbitrarily small. By theorem 1, it also follows that there exists an SPD S such that $\|g - S\|_\infty$ is arbitrarily small. We can then write $\|f - S\|_p \leq \|f - g\|_p + \|g - S\|_p$ using triangle inequality and make the right hand side arbitrarily small. This implies that $\|f - S\|_p$ can be made arbitrarily small. See section B for the full proof. □

5 DATASET

DancingThings starts from a pre-existing scene generated from Infinigen Indoors (Raistrick et al., 2024), then animates and procedurally deforms its objects. After generating a scene, we apply in the walls, floors, ceiling and objects textures from the Describable Texture Dataset (DTD) (Cimpoi et al., 2014), we animate the objects with the RRT*-based path planning algorithm (Raistrick et al., 2026) and deform them using our SPD method. In section C, we show that SPD leaves materials consistent through the deformation process.

For each generated and rendered scene, we export the ground truth optical flow and combine the 3D flow and depth to get the occlusion masks for the invalid pixels. Fig. 4 shows an example of successive frames with the ground-truth annotations.

For the SPD parameters in our dataset, we use spline degree 3, $n_u = n_v = n_w = 4$ along the three spatial axes, amplitude $A = 0.3$ and uniform basis weights $w_i, j, k = 1 \; \forall i, j, k$ in 2. For the space-time deformation field we use $n_t = 40$ temporal control points.

DancingThings consists of 24 scenes, each with 100 frames per scene, resulting in a total of 2376 pairs. Each video is generated in 24 fps in the Eevee rendering engine using the default settings, and has a resolution of 1280×720. See section C for extra dataset statistics.

Occlusion Masks In addition to the ground truth optical flow for all pixels, we also provide occlusion masks that indicate which pixels have valid flow. Specifically, in optical flow between two consecutive frames, occlusions are simply the pixels that don't have a valid matching point in the next frame because they disappear or reappear from occlusions. See section F for details.

6 EXPERIMENTS

Performance on Non-Rigid Benchmarks We conduct a series of experiments in order to show that baseline methods achieve higher accuracy on standard benchmarks with non-rigid motion

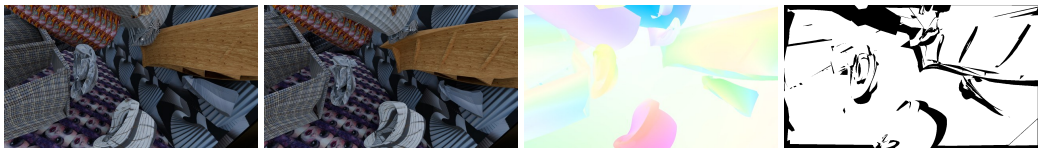


Figure 4: An illustration of our ground truth. Each row from left to right shows: image at frame t, image at frame $t+1$, optical flow for all the pixels, and occlusion masks. Note that the occlusion masks include pixels that appear from t to $t+1$ and disappear from t to $t+1$. Both pixels that move in and out of the field-of-view as well as pixels that go behind an object are masked.

Table 2: Results of SEA-RAFT (Wang et al., 2024) and WAFT (Wang & Deng, 2026) on the Spring Non-Rigid val set (Mehl et al., 2023) and Sintel (train) (Butler et al., 2012). C in the "Trained" column denotes FlyingChairs (Dosovitskiy et al., 2015b). Sintel has two rendering passes of the same scenes. Clean is the simpler render while final adds realistic blur, atmosphere, and lighting effects. Sintel results are reported as clean/final.

Model	Trained	Spring Non Rigid (val)				Sintel			
		EPE ↓	1 px ↓	3 px ↓	5 px ↓	EPE ↓	1 px ↓	3 px ↓	5 px ↓
SEA-RAFT	TartanAir	5.29	45.91	24.86	19.30	1.41/3.74	8.40/17.98	4.52/12.01	3.42 /9.77
SEA-RAFT	TartanAir + C	4.47	63.94	21.66	13.58	1.38/3.32	12.30/22.67	3.78/8.50	2.66/6.58
SEA-RAFT	**DancingThings**	**3.10**	**36.80**	**15.31**	**10.36**	1.58/**2.28**	**9.67/14.18**	**4.64 / 7.64**	3.55/**5.86**
WAFT	TartanAir	7.98	61.67	40.28	33.52	5.80/6.25	18.19/20.90	14.30/16.24	12.27/14.35
WAFT	TartanAir + C	4.96	73.11	27.06	18.68	2.35/3.78	13.29/21.35	7.23/11.73	5.73/9.43
WAFT	**DancingThings**	**3.90**	**44.74**	**19.62**	**13.93**	**1.94/2.30**	**14.09/16.37**	**6.19/8.05**	**4.41/5.78**

when trained on DancingThings. Specifically, we evaluate SEA-RAFT (Wang et al., 2024) and WAFT (Wang & Deng, 2026) on the Spring non rigid validation set (Mehl et al., 2023) and Sintel train set (Butler et al., 2012) and SEA-RAFT (Wang et al., 2024) on the Deformable Object Tracking Dataset (DOT) (Li et al., 2025) train set. As evaluation metrics we use the the average end-point-error (EPE), which measures the L2 distance between predicted and ground truth optical flow, as well as the 1px, 3px, 5px error. More details about the datasets can be found in section section E

On the Spring Non-rigid and Sintel, we use SEA-RAFT (Wang et al., 2024) trained on TartanAir (Wang et al., 2020) and on TartanAir (Wang et al., 2020) + FlyingChairs (Dosovitskiy et al., 2015b), and compare these models with SEA-RAFT trained only on DancingThings. As we can see from Table 2, even though our dataset is significantly smaller in size (2376 pairs) compared to Tartanair (306268 pairs) and FlyingChairs (22872 pairs), training on our dataset yields consistenly better performance across most of the metrics, with a significant reduction in EPE. Next, we also test WAFT's performance when trained on TartainAir and TartainAir + FlyingChairs versus WAFT trained on DancingThings and we see that the latter is consistenly better.

Table 3: Average EPE on DOT (train) (Li et al., 2025) by deformation category. T and C in the "Trained" column denote FlyingThings3D (Mayer et al., 2016) and FlyingChairs (Dosovitskiy et al., 2015b) respectively.

Model	Trained	Cloth	Paper	Rope	All
SEA-RAFT	TartanAir + C	0.55	0.63	1.20	0.95
SEA-RAFT	TartanAir + C + T	0.54	0.94	**0.69**	0.71
SEA-RAFT	**DancingThings**	**0.50**	**0.50**	0.71	**0.63**

For the DOT dataset, we evaluate SEA-RAFT (Wang et al., 2024) trained on different datasets across 3 categories: cloth, paper and ropes. As shown in table 3, DancingThings outperforms TartanAir, FlyingChairs, and FlyingThings combined. We also report the results across some individual sequences in section E.

Performance on Rigid-Motion Benchmarks Despite being deformation-based, DancingThings does not hurt performance on rigid-motion benchmarks. In fact, it performs better than rigid-motion-

Table 4: **Rigid Motion Benchmarks:** Despite being deformation-based, DancingThings surprisingly performs better than rigid-motion datasets on rigid motion benchmarks like KITTI (train) (Menze & Geiger, 2015).We provide also results in Spring (val) (all) (Mehl et al., 2023). C in the "Trained" column denotes FlyingChairs (Dosovitskiy et al., 2015b).

Model	Trained	Spring val (all)		KITTI	
		EPE	1px	EPE	F1-all
SEA-RAFT	TartanAir+C	0.93	15.84	9.44	34.51
SEA-RAFT	**DancingThings**	**0.83**	**14.63**	**5.62**	**16.13**
WAFT	TartanAir + C	0.96	16.02	29.91	68.27
WAFT	**DancingThings**	**0.90**	**14.27**	**9.12**	**21.22**

Table 5: **Deformation Datasets:** DancingThings, despite smaller in size, performs better than other deformation-based datasets with significantly lower EPE across Spring Non Rigid (val) (Mehl et al., 2023), Sintel(train) (Butler et al., 2012), and DOT(train) (Li et al., 2025) and lower Fl for KITTI(train) (Menze & Geiger, 2015).

Model	Trained	Spring Non Rigid	KITTI		Sintel		DOT(all avg)
			EPE	F1-all	Clean	Final	
SEA-RAFT	Autoflow (Sun et al., 2021)	3.11	12.10	21.23	2.12	2.27	0.66
SEA-RAFT	DeformingThings4D (Li et al., 2021)	5.82	22.58	74.59	5.61	4.68	0.95
SEA-RAFT	**DancingThings** with DTD	**3.10**	**5.62**	**16.13**	**1.59**	**2.27**	**0.63**

based datasets on KITTI (train) (Menze & Geiger, 2015) as shown in table 4. We provide also results in Spring (val) (all) (Mehl et al., 2023).

Comparison to Deformation-based Datasets We further compare DancingThings with other deformation-based training datasets DeformingThings4D (Li et al., 2021) and AutoFlow (Sun et al., 2021) on Spring (non rigid), KITTI(train) (Menze & Geiger, 2015), Sintel(train) (Butler et al., 2012) and DOT(val) (Li et al., 2025). Table 5 shows that DancingThings significantly outperforms both DeformingThings4D (122335 pairs) and AutoFlow (40K pairs), even though it is significantly smaller in size and has only 2376 pairs.

Ablation Study In order to analyze the impact of our design choices, we perform an ablation study. We show in table 6 that SPD deformations perform better than both static scenes of Infinigen Indoors (Raistrick et al., 2024) and rigid-motion. Also, rasterizing-based Eevee rendering engine performs better than ray-tracing based Cycles engine. An analysis of the ablation can be found on section section G.

We also compare our SPD directly against physical simulation-based deformations. SPD outperforms Blender soft-body simulation, reducing EPE from 4.62 to 3.73. See section D for details.

Table 6: Ablation study on 8217 frame pairs evaluating on Spring Non Rigid (val) (Mehl et al., 2023) different rendering techniques and motion applied to the scene. All rows use the SEA-RAFT (Wang et al., 2024) model.

Render	Category	A	n_t	n_{uvw}	EPE ↓
Eevee	Static	-	-	-	4.18
Eevee	Rigid Motion	-	-	-	3.86
Eevee	SPD	0.1	10	(4,4,4)	3.99
Eevee	SPD	0.5	10	(4,4,4)	3.83
Eevee	SPD	0.3	10	(4,4,4)	3.72
Cycles	SPD	0.3	10	(4,4,4)	3.78
Eevee	SPD	0.3	10	(6,6,6)	3.98
Eevee	SPD	0.3	4	(4,4,4)	3.81
Eevee	**SPD**	**0.3**	**40**	**(4,4,4)**	**3.70**

REFERENCES

Blender. https://blender.org.

Simon Baker, Daniel Scharstein, J. P. Lewis, Stefan Roth, Michael J. Black, and Richard Szeliski. A database and evaluation methodology for optical flow. In *IEEE 11th International Conference on Computer Vision, ICCV 2007, Rio de Janeiro, Brazil, October 14-20, 2007*, pp. 1–8. IEEE Computer Society, 2007. doi: 10.1109/ICCV.2007.4408903. URL https://doi.org/10.1109/ICCV.2007.4408903.

Aljaz Bozic, Michael Zollhöfer, Christian Theobalt, and Matthias Nießner. Deepdeform: Learning non-rigid RGB-D reconstruction with semi-supervised data. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2020, Seattle, WA, USA, June 13-19, 2020*, pp. 7000–7010. Computer Vision Foundation / IEEE, 2020. doi: 10.1109/CVPR42600.2020.00703. URL https://openaccess.thecvf.com/content_CVPR_2020/html/Bozic_DeepDeform_Learning_Non-Rigid_RGB-D_Reconstruction_With_Semi-Supervised_Data_CVPR_2020_paper.html.

Daniel J. Butler, Jonas Wulff, Garrett B. Stanley, and Michael J. Black. A naturalistic open source movie for optical flow evaluation. In Andrew W. Fitzgibbon, Svetlana Lazebnik, Pietro Perona, Yoichi Sato, and Cordelia Schmid (eds.), *Computer Vision - ECCV 2012 - 12th European Conference on Computer Vision, Florence, Italy, October 7-13, 2012, Proceedings, Part VI*, volume 7577 of *Lecture Notes in Computer Science*, pp. 611–625. Springer, 2012. doi: 10.1007/978-3-642-33783-3_44. URL https://doi.org/10.1007/978-3-642-33783-3_44.

Yohann Cabon, Naila Murray, and Martin Humenberger. Virtual kitti 2, 2020. URL https://arxiv.org/abs/2001.10773.

Mircea Cimpoi, Subhransu Maji, Iasonas Kokkinos, Sammy Mohamed, and Andrea Vedaldi. Describing textures in the wild. In *2014 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2014, Columbus, OH, USA, June 23-28, 2014*, pp. 3606–3613. IEEE Computer Society, 2014. doi: 10.1109/CVPR.2014.461. URL https://doi.org/10.1109/CVPR.2014.461.

Changxing Deng, Ao Luo, Haibin Huang, Shaodan Ma, Jiangyu Liu, and Shuaicheng Liu. Explicit motion disentangling for efficient optical flow estimation. In *IEEE/CVF International Conference on Computer Vision, ICCV 2023, Paris, France, October 1-6, 2023*, pp. 9487–9496. IEEE, 2023. doi: 10.1109/ICCV51070.2023.00873. URL https://doi.org/10.1109/ICCV51070.2023.00873.

Alexey Dosovitskiy, Philipp Fischer, Eddy Ilg, Philip Häusser, Caner Hazirbas, Vladimir Golkov, Patrick van der Smagt, Daniel Cremers, and Thomas Brox. Flownet: Learning optical flow with convolutional networks. In *2015 IEEE International Conference on Computer Vision, ICCV 2015, Santiago, Chile, December 7-13, 2015*, pp. 2758–2766. IEEE Computer Society, 2015a. doi: 10.1109/ICCV.2015.316. URL https://doi.org/10.1109/ICCV.2015.316.

Alexey Dosovitskiy, Philipp Fischer, Eddy Ilg, Philip Häusser, Caner Hazirbas, Vladimir Golkov, Patrick van der Smagt, Daniel Cremers, and Thomas Brox. Flownet: Learning optical flow with convolutional networks. In *2015 IEEE International Conference on Computer Vision, ICCV 2015, Santiago, Chile, December 7-13, 2015*, pp. 2758–2766. IEEE Computer Society, 2015b. doi: 10.1109/ICCV.2015.316. URL https://doi.org/10.1109/ICCV.2015.316.

Adrien Gaidon, Qiao Wang, Yohann Cabon, and Eleonora Vig. Virtualworlds as proxy for multi-object tracking analysis. In *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*, pp. 4340–4349. IEEE Computer Society, 2016. doi: 10.1109/CVPR.2016.470. URL https://doi.org/10.1109/CVPR.2016.470.

Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? the KITTI vision benchmark suite. In *2012 IEEE Conference on Computer Vision and Pattern Recognition, Providence, RI, USA, June 16-21, 2012*, pp. 3354–3361. IEEE Computer Society, 2012. doi: 10.1109/CVPR.2012.6248074. URL https://doi.org/10.1109/CVPR.2012.6248074.

Berthold K. P. Horn and Brian G. Schunck. Determining optical flow. *Artif. Intell.*, 17(1-3): 185–203, 1981. doi: 10.1016/0004-3702(81)90024-2. URL https://doi.org/10.1016/0004-3702(81)90024-2.

Zhewei Huang, Tianyuan Zhang, Wen Heng, Boxin Shi, and Shuchang Zhou. Real-time intermediate flow estimation for video frame interpolation. In Shai Avidan, Gabriel J. Brostow, Moustapha Cissé, Giovanni Maria Farinella, and Tal Hassner (eds.), *Computer Vision - ECCV 2022 - 17th European Conference, Tel Aviv, Israel, October 23-27, 2022, Proceedings, Part XIV*, volume 13674 of *Lecture Notes in Computer Science*, pp. 624–642. Springer, 2022. doi: 10.1007/978-3-031-19781-9_36. URL https://doi.org/10.1007/978-3-031-19781-9_36.

Tak-Wai Hui, Xiaoou Tang, and Chen Change Loy. Liteflownet: A lightweight convolutional neural network for optical flow estimation. In *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018*, pp. 8981–8989. Computer Vision Foundation / IEEE Computer Society, 2018. doi: 10.1109/CVPR.2018.00936. URL http://openaccess.thecvf.com/content_cvpr_2018/html/Hui_LiteFlowNet_A_Lightweight_CVPR_2018_paper.html.

Eddy Ilg, Nikolaus Mayer, Tonmoy Saikia, Margret Keuper, Alexey Dosovitskiy, and Thomas Brox. Flownet 2.0: Evolution of optical flow estimation with deep networks. In *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*, pp. 1647–1655. IEEE Computer Society, 2017. doi: 10.1109/CVPR.2017.179. URL https://doi.org/10.1109/CVPR.2017.179.

Azin Jahedi, Maximilian Luz, Marc Rivinius, and Andrés Bruhn. CCMR: high resolution optical flow estimation via coarse-to-fine context-guided motion reasoning. In *IEEE/CVF Winter Conference on Applications of Computer Vision, WACV 2024, Waikoloa, HI, USA, January 3-8, 2024*, pp. 6885–6894. IEEE, 2024a. doi: 10.1109/WACV57701.2024.00675. URL https://doi.org/10.1109/WACV57701.2024.00675.

Azin Jahedi, Maximilian Luz, Marc Rivinius, Lukas Mehl, and Andrés Bruhn. MS-RAFT+: high resolution multi-scale RAFT. *Int. J. Comput. Vis.*, 132(5):1835–1856, 2024b. doi: 10.1007/S11263-023-01930-7. URL https://doi.org/10.1007/s11263-023-01930-7.

Sertac Karaman and Emilio Frazzoli. Sampling-based algorithms for optimal motion planning. *Int. J. Robotics Res.*, 30(7):846–894, 2011. doi: 10.1177/0278364911406761. URL https://doi.org/10.1177/0278364911406761.

Karhan Kayan, Stamatis Alexandropoulos, Rishabh Jain, Yiming Zuo, Erich Liang, and Jia Deng. Princeton365: A diverse dataset with accurate camera pose. In *IEEE/CVF International Conference on Computer Vision, ICCV 2025, Honolulu, HI, USA, October 19-25, 2025*, pp. 1–10. IEEE, 2025. doi: 10.1109/ICCV51701.2025.00717. URL https://doi.org/10.1109/ICCV51701.2025.00717.

Simon Kiefhaber, Stefan Roth, and Simone Schaub-Meyer. Removing cost volumes from optical flow estimators. In *IEEE/CVF International Conference on Computer Vision, ICCV 2025, Honolulu, HI, USA, October 19-25, 2025*, pp. 79–89. IEEE, 2025. doi: 10.1109/ICCV51701.2025.00015. URL https://doi.org/10.1109/ICCV51701.2025.00015.

Daniel Kondermann, Rahul Nair, Katrin Honauer, Karsten Krispin, Jonas Andrulis, Alexander Brock, Burkhard Güssefeld, Mohsen Rahimimoghaddam, Sabine Hofmann, Claus Brenner, and Bernd Jähne. The HCI benchmark suite: Stereo and flow ground truth with uncertainties for urban autonomous driving. In *2016 IEEE Conference on Computer Vision and Pattern Recognition Workshops, CVPR Workshops 2016, Las Vegas, NV, USA, June 26 - July 1, 2016*, pp. 19–28. IEEE Computer Society, 2016. doi: 10.1109/CVPRW.2016.10. URL https://doi.org/10.1109/CVPRW.2016.10.

Vincent Leroy, Jérôme Revaud, Thomas Lucas, and Philippe Weinzaepfel. Win-win: Training high-resolution vision transformers from two windows. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net, 2024. URL https://openreview.net/forum?id=N23A4ybMJr.

Xinyuan Li, Yu Guo, Yubei Tu, Yu Ji, Yanchen Liu, Jinwei Ye, and Changxi Zheng. Textureless deformable object tracking with invisible markers. *IEEE Trans. Pattern Anal. Mach. Intell.*, 47 (9):7243–7254, 2025. doi: 10.1109/TPAMI.2024.3463422. URL https://doi.org/10.1109/TPAMI.2024.3463422.

Yang Li, Hikari Takehara, Takafumi Taketomi, Bo Zheng, and Matthias Nießner. 4dcomplete: Non-rigid motion estimation beyond the observable surface. In *2021 IEEE/CVF International Conference on Computer Vision, ICCV 2021, Montreal, QC, Canada, October 10-17, 2021*, pp. 12686–12696. IEEE, 2021. doi: 10.1109/ICCV48922.2021.01247. URL https://doi.org/10.1109/ICCV48922.2021.01247.

Lahav Lipson, Zachary Teed, and Jia Deng. Raft-stereo: Multilevel recurrent field transforms for stereo matching. In *International Conference on 3D Vision, 3DV 2021, London, United Kingdom, December 1-3, 2021*, pp. 218–227. IEEE, 2021. doi: 10.1109/3DV53792.2021.00032. URL https://doi.org/10.1109/3DV53792.2021.00032.

Bruce D. Lucas and Takeo Kanade. An iterative image registration technique with an application to stereo vision. In Patrick J. Hayes (ed.), *Proceedings of the 7th International Joint Conference on Artificial Intelligence, IJCAI '81, Vancouver, BC, Canada, August 24-28, 1981*, pp. 674–679. William Kaufmann, 1981.

Ao Luo, Fan Yang, Xin Li, Lang Nie, Chunyu Lin, Haoqiang Fan, and Shuaicheng Liu. Gaflow: Incorporating gaussian attention into optical flow. In *IEEE/CVF International Conference on Computer Vision, ICCV 2023, Paris, France, October 1-6, 2023*, pp. 9608–9617. IEEE, 2023. doi: 10.1109/ICCV51070.2023.00884. URL https://doi.org/10.1109/ICCV51070.2023.00884.

Zeyu Ma, Alexander Raistrick, and Jia Deng. Simpleproc: Fully procedural synthetic data from simple rules for multi-view stereo, 2026. URL https://arxiv.org/abs/2604.04925.

Nikolaus Mayer, Eddy Ilg, Philip Häusser, Philipp Fischer, Daniel Cremers, Alexey Dosovitskiy, and Thomas Brox. A large dataset to train convolutional networks for disparity, optical flow, and scene flow estimation. In *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*, pp. 4040–4048. IEEE Computer Society, 2016. doi: 10.1109/CVPR.2016.438. URL https://doi.org/10.1109/CVPR.2016.438.

Lukas Mehl, Jenny Schmalfuss, Azin Jahedi, Yaroslava Nalivayko, and Andrés Bruhn. Spring: A high-resolution high-detail dataset and benchmark for scene flow, optical flow and stereo. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2023, Vancouver, BC, Canada, June 17-24, 2023*, pp. 4981–4991. IEEE, 2023. doi: 10.1109/CVPR52729.2023.00482. URL https://doi.org/10.1109/CVPR52729.2023.00482.

Moritz Menze and Andreas Geiger. Object scene flow for autonomous vehicles. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2015, Boston, MA, USA, June 7-12, 2015*, pp. 3061–3070. IEEE Computer Society, 2015. doi: 10.1109/CVPR.2015.7298925. URL https://doi.org/10.1109/CVPR.2015.7298925.

Javier Sánchez Pérez, Enric Meinhardt-Llopis, and Gabriele Facciolo. TV-L1 optical flow estimation. *Image Process. Line*, 3:137–150, 2013. doi: 10.5201/IPOL.2013.26. URL https://doi.org/10.5201/ipol.2013.26.

Les A. Piegl and Wayne Tiller. *The NURBS Book, Second Edition*. Monographs in Visual Communication. Springer, 1997. ISBN 978-3-540-61545-3. doi: 10.1007/978-3-642-59223-2. URL https://doi.org/10.1007/978-3-642-59223-2.

Hong Qin and Demetri Terzopoulos. Triangular NURBS and their dynamic generalizations. *Comput. Aided Geom. Des.*, 14(4):325–347, 1997. doi: 10.1016/S0167-8396(96)00062-3. URL https://doi.org/10.1016/S0167-8396(96)00062-3.

Alexander Raistrick, Lahav Lipson, Zeyu Ma, Lingjie Mei, Mingzhe Wang, Yiming Zuo, Karhan Kayan, Hongyu Wen, Beining Han, Yihan Wang, Alejandro Newell, Hei Law, Ankit Goyal, Kaiyu Yang, and Jia Deng. Infinite photorealistic worlds using procedural generation. In *IEEE/CVF*

Conference on Computer Vision and Pattern Recognition, CVPR 2023, Vancouver, BC, Canada, June 17-24, 2023, pp. 12630–12641. IEEE, 2023. doi: 10.1109/CVPR52729.2023.01215. URL https://doi.org/10.1109/CVPR52729.2023.01215.

Alexander Raistrick, Lingjie Mei, Karhan Kayan, David Yan, Yiming Zuo, Beining Han, Hongyu Wen, Meenal Parakh, Stamatis Alexandropoulos, Lahav Lipson, Zeyu Ma, and Jia Deng. Infinigen indoors: Photorealistic indoor scenes using procedural generation. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2024, Seattle, WA, USA, June 16-22, 2024*, pp. 21783–21794. IEEE, 2024. doi: 10.1109/CVPR52733.2024.02058. URL https://doi.org/10.1109/CVPR52733.2024.02058.

Alexander Raistrick, Karhan Kayan, Jack Nugent, David Yan, Lingjie Mei, Meenal Parakh, Hongyu Wen, Dylan Li, Yiming Zuo, Erich Liang, and Jia Deng. Procfunc: Function-oriented abstractions for procedural 3d generation in python, 2026. URL https://arxiv.org/abs/2604.26943.

Anurag Ranjan, David T. Hoffmann, Dimitrios Tzionas, Siyu Tang, Javier Romero, and Michael J. Black. Learning multi-human optical flow. *Int. J. Comput. Vis.*, 128(4):873–890, 2020. doi: 10.1007/S11263-019-01279-W. URL https://doi.org/10.1007/s11263-019-01279-w.

Stephan R. Richter, Zeeshan Hayder, and Vladlen Koltun. Playing for benchmarks. In *IEEE International Conference on Computer Vision, ICCV 2017, Venice, Italy, October 22-29, 2017*, pp. 2232–2241. IEEE Computer Society, 2017. doi: 10.1109/ICCV.2017.243. URL https://doi.org/10.1109/ICCV.2017.243.

Walter Rudin. *Principles of Mathematical Analysis*. McGraw-Hill, 1953.

Saurabh Saxena, Charles Herrmann, Junhwa Hur, Abhishek Kar, Mohammad Norouzi, Deqing Sun, and David J. Fleet. The surprising effectiveness of diffusion models for optical flow and monocular depth estimation. In Alice Oh, Tristan Naumann, Amir Globerson, Kate Saenko, Moritz Hardt, and Sergey Levine (eds.), *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*, 2023. URL http://papers.nips.cc/paper_files/paper/2023/hash/7c119415672ae2186e17d492e1d5da2f-Abstract-Conference.html.

Thomas W. Sederberg and Scott R. Parry. Free-form deformation of solid geometric models. In David C. Evans and Russell J. Athay (eds.), *Proceedings of the 13th Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH 1986, Dallas, Texas, USA, August 18-22, 1986*, pp. 151–160. ACM, 1986. doi: 10.1145/15922.15903. URL https://doi.org/10.1145/15922.15903.

Xiaoyu Shi, Zhaoyang Huang, Dasong Li, Manyuan Zhang, Ka Chun Cheung, Simon See, Hongwei Qin, Jifeng Dai, and Hongsheng Li. Flowformer++: Masked cost volume autoencoding for pretraining optical flow estimation. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2023, Vancouver, BC, Canada, June 17-24, 2023*, pp. 1599–1610. IEEE, 2023. doi: 10.1109/CVPR52729.2023.00160. URL https://doi.org/10.1109/CVPR52729.2023.00160.

Olga Sorkine and Marc Alexa. As-rigid-as-possible surface modeling. In Alexander G. Belyaev and Michael Garland (eds.), *Proceedings of the Fifth Eurographics Symposium on Geometry Processing, Barcelona, Spain, July 4-6, 2007*, volume 257 of *ACM International Conference Proceeding Series*, pp. 109–116. Eurographics Association, 2007. doi: 10.2312/SGP/SGP07/109-116. URL https://doi.org/10.2312/SGP/SGP07/109-116.

Xiuchao Sui, Shaohua Li, Xue Geng, Yan Wu, Xinxing Xu, Yong Liu, Rick Siow Mong Goh, and Hongyuan Zhu. CRAFT: cross-attentional flow transformer for robust optical flow. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2022, New Orleans, LA, USA, June 18-24, 2022*, pp. 17581–17590. IEEE, 2022. doi: 10.1109/CVPR52688.2022.01708. URL https://doi.org/10.1109/CVPR52688.2022.01708.

Robert W. Sumner and Jovan Popovic. Deformation transfer for triangle meshes. *ACM Trans. Graph.*, 23(3):399–405, 2004. doi: 10.1145/1015706.1015736. URL https://doi.org/10.1145/1015706.1015736.

Deqing Sun, Xiaodong Yang, Ming-Yu Liu, and Jan Kautz. Pwc-net: Cnns for optical flow using pyramid, warping, and cost volume. In *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018*, pp. 8934–8943. Computer Vision Foundation / IEEE Computer Society, 2018. doi: 10.1109/CVPR.2018.00931. URL http://openaccess.thecvf.com/content_cvpr_2018/html/Sun_PWC-Net_CNNs_for_CVPR_2018_paper.html.

Deqing Sun, Daniel Vlasic, Charles Herrmann, Varun Jampani, Michael Krainin, Huiwen Chang, Ramin Zabih, William T. Freeman, and Ce Liu. Autoflow: Learning a better training set for optical flow. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2021, virtual, June 19-25, 2021*, pp. 10093–10102. Computer Vision Foundation / IEEE, 2021. doi: 10.1109/CVPR46437.2021.00996. URL https://openaccess.thecvf.com/content/CVPR2021/html/Sun_AutoFlow_Learning_a_Better_Training_Set_for_Optical_Flow_CVPR_2021_paper.html.

Zachary Teed and Jia Deng. RAFT: recurrent all-pairs field transforms for optical flow. In Andrea Vedaldi, Horst Bischof, Thomas Brox, and Jan-Michael Frahm (eds.), *Computer Vision - ECCV 2020 - 16th European Conference, Glasgow, UK, August 23-28, 2020, Proceedings, Part II*, volume 12347 of *Lecture Notes in Computer Science*, pp. 402–419. Springer, 2020. doi: 10.1007/978-3-030-58536-5_24. URL https://doi.org/10.1007/978-3-030-58536-5_24.

Zachary Teed and Jia Deng. DROID-SLAM: deep visual SLAM for monocular, stereo, and RGB-D cameras. In Marc'Aurelio Ranzato, Alina Beygelzimer, Yann N. Dauphin, Percy Liang, and Jennifer Wortman Vaughan (eds.), *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, pp. 16558–16569, 2021a. URL https://proceedings.neurips.cc/paper/2021/hash/89fcd07f20b6785b92134bd6c1d0fa42-Abstract.html.

Zachary Teed and Jia Deng. RAFT-3D: scene flow using rigid-motion embeddings. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2021, virtual, June 19-25, 2021*, pp. 8375–8384. Computer Vision Foundation / IEEE, 2021b. doi: 10.1109/CVPR46437.2021.00827. URL https://openaccess.thecvf.com/content/CVPR2021/html/Teed_RAFT-3D_Scene_Flow_Using_Rigid-Motion_Embeddings_CVPR_2021_paper.html.

Zachary Teed, Lahav Lipson, and Jia Deng. Deep patch visual odometry. In Alice Oh, Tristan Naumann, Amir Globerson, Kate Saenko, Moritz Hardt, and Sergey Levine (eds.), *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*, 2023. URL http://papers.nips.cc/paper_files/paper/2023/hash/7ac484b0f1a1719ad5be9aa8c8455fbb-Abstract-Conference.html.

Demetri Terzopoulos and Hong Qin. Dynamic NURBS with geometric constraints for interactive sculpting. *ACM Trans. Graph.*, 13(2):103–136, 1994. doi: 10.1145/176579.176580. URL https://doi.org/10.1145/176579.176580.

Wenshan Wang, Delong Zhu, Xiangwei Wang, Yaoyu Hu, Yuheng Qiu, Chen Wang, Yafei Hu, Ashish Kapoor, and Sebastian A. Scherer. Tartanair: A dataset to push the limits of visual SLAM. In *IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 2020, Las Vegas, NV, USA, October 24, 2020 - January 24, 2021*, pp. 4909–4916. IEEE, 2020. doi: 10.1109/IROS45743.2020.9341801. URL https://doi.org/10.1109/IROS45743.2020.9341801.

Yihan Wang and Jia Deng. Waft: Warping-alone field transforms for optical flow, 2026. URL https://arxiv.org/abs/2506.21526.

Yihan Wang, Lahav Lipson, and Jia Deng. SEA-RAFT: simple, efficient, accurate RAFT for optical flow. In Ales Leonardis, Elisa Ricci, Stefan Roth, Olga Russakovsky, Torsten Sattler, and Gül Varol (eds.), *Computer Vision - ECCV 2024 - 18th European Conference, Milan, Italy, September 29-October 4, 2024, Proceedings, Part VII*, volume 15065 of *Lecture Notes in Computer Science*, pp. 36–54. Springer, 2024. doi: 10.1007/978-3-031-72667-5_3. URL https://doi.org/10.1007/978-3-031-72667-5_3.

Yiran Wang, Min Shi, Jiaqi Li, Zihao Huang, Zhiguo Cao, Jianming Zhang, Ke Xian, and Guosheng Lin. Neural video depth stabilizer. In *IEEE/CVF International Conference on Computer Vision, ICCV 2023, Paris, France, October 1-6, 2023*, pp. 9432–9442. IEEE, 2023. doi: 10.1109/ICCV51070.2023.00868. URL https://doi.org/10.1109/ICCV51070.2023.00868.

Felix Wimbauer, Weirong Chen, Dominik Muhle, Christian Rupprecht, and Daniel Cremers. Anycam: Learning to recover camera poses and intrinsics from casual videos. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2025, Nashville, TN, USA, June 11-15, 2025*, pp. 16717–16727. Computer Vision Foundation / IEEE, 2025. doi: 10.1109/CVPR52734.2025.01558. URL https://openaccess.thecvf.com/content/CVPR2025/html/Wimbauer_AnyCam_Learning_to_Recover_Camera_Poses_and_Intrinsics_from_Casual_CVPR_2025_paper.html.

Andrew P. Witkin and Michael Kass. Spacetime constraints. In Richard J. Beach (ed.), *Proceedings of the 15th Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH 1988, Atlanta, Georgia, USA, August 1-5, 1988*, pp. 159–168. ACM, 1988. doi: 10.1145/54852.378507. URL https://doi.org/10.1145/54852.378507.

Haofei Xu, Jing Zhang, Jianfei Cai, Hamid Rezatofighi, and Dacheng Tao. Gmflow: Learning optical flow via global matching. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2022, New Orleans, LA, USA, June 18-24, 2022*, pp. 8111–8120. IEEE, 2022. doi: 10.1109/CVPR52688.2022.00795. URL https://doi.org/10.1109/CVPR52688.2022.00795.

Rajeev Yasarla, Hong Cai, Jisoo Jeong, Yunxiao Shi, Risheek Garrepalli, and Fatih Porikli. Mamo: Leveraging memory and attention for monocular video depth estimation. In *IEEE/CVF International Conference on Computer Vision, ICCV 2023, Paris, France, October 1-6, 2023*, pp. 8720–8730. IEEE, 2023. doi: 10.1109/ICCV51070.2023.00804. URL https://doi.org/10.1109/ICCV51070.2023.00804.

Christopher Zach, Thomas Pock, and Horst Bischof. A duality based approach for realtime tv-L^1 optical flow. In Fred A. Hamprecht, Christoph Schnörr, and Bernd Jähne (eds.), *Pattern Recognition, 29th DAGM Symposium, Heidelberg, Germany, September 12-14, 2007, Proceedings*, volume 4713 of *Lecture Notes in Computer Science*, pp. 214–223. Springer, 2007. doi: 10.1007/978-3-540-74936-3_22. URL https://doi.org/10.1007/978-3-540-74936-3_22.

Kaidong Zhang, Jingjing Fu, and Dong Liu. Flow-guided transformer for video inpainting. In Shai Avidan, Gabriel J. Brostow, Moustapha Cissé, Giovanni Maria Farinella, and Tal Hassner (eds.), *Computer Vision - ECCV 2022 - 17th European Conference, Tel Aviv, Israel, October 23-27, 2022, Proceedings, Part XVIII*, volume 13678 of *Lecture Notes in Computer Science*, pp. 74–90. Springer, 2022. doi: 10.1007/978-3-031-19797-0_5. URL https://doi.org/10.1007/978-3-031-19797-0_5.

A Appendix

B Proof of the SPD Approximation Theorems

In theorem 1, we stated that SPD can approximate any physically plausible deformation arbitrarily well, and thus can be thought of as a generalization of physically plausible deformations. In this section, we give a proof of this theorem and discuss its assumptions.

Our strategy will be to prove that any continuous function $f : \mathbf{K} \to \mathbb{R}^3$ can be approximated by SPD, where $\mathbf{K} = I_1 \times I_2 \times I_3 \times I_4$ is a product of closed intervals in $\mathbb{R}^4$. Note that this is a necessary, but not sufficient condition for most types of physically-realistic deformations. For instance, a continuous function can shrink the volume of an object to zero, which is not possible with physically realistic deformations. Moreover, physically-realistic deformations are subject to Newton's laws, but the set of continuous functions are not. However, this is not an issue for our result because if any physically-realistic deformation is necessarily a continuous function $f : \mathbf{K} \to \mathbb{R}^3$ and SPD can approximate any continuous function, then SPD can approximate any physically-realistic deformation.

To see the necessity, first note that the domain of physically-plausible functions can be formulated as a product of closed intervals $\mathbf{K} = I_x \times I_y \times I_z \times I_t$. The reason is that any physical object is spatially-finite and can be embedded in a finite box. Furthermore, we are interested in deformations that last a finite amount of time and do not contain "skips," which can be thought of as being defined on a closed interval. Then, what a deformation $f : \mathbf{K} \to \mathbb{R}^3$ means is that we place the object in a box $I_x \times I_y \times I_z$ at $t = 0$ and the point $\mathbf{X} \in \mathbb{R}^3$ on the object maps (i.e. deforms) to the point $f(\mathbf{X}, t) \in \mathbb{R}^3$ at time t.

Secondly, spatial continuity is necessary for the types of deformations we are interested in. Suppose $f_t(\mathbf{X}) = f(\mathbf{X}, t)$ is discontinuous at point $\mathbf{X}_0$. Then, there exists $\varepsilon > 0$ such that for all $\delta > 0$, there exists an $\mathbf{X}$ such that $||\mathbf{X} - \mathbf{X}_0||_2 < \delta$, but $||f_t(\mathbf{X}) - f_t(\mathbf{X}_0)||_2 > \varepsilon$. This corresponds to a tearing at $\mathbf{X}_0$, which we assume that the physically-realistic deformation does not have. Note that topology-preserving physically-realistic deformations do not have tearing, since by definition topology-preserving means that $f_t(\mathbf{X})$ is a homeomorphism, i.e. a continuous bijection with continuous inverse.

Thirdly, temporal continuity is necessary for physically-plausible deformations since $f_{\mathbf{X}}(t) = f(\mathbf{X}, t)$ being discontinuous would mean that point $\mathbf{X}$ teleports in space at the time of discontinuity, which is impossible because it would imply infinite velocity, and thus infinite energy.

Given these assumptions, we will now prove theorem 1. We will need the following well-known theorem from analysis:

Theorem (**Stone-Weierstrass Approximation Theorem**). *Let $\mathcal{A}$ be an algebra of real continuous functions on a compact set $\mathbf{K}$. If $\mathcal{A}$ separates points on $\mathbf{K}$ and if $\mathcal{A}$ vanishes at no point of $\mathbf{K}$, then the uniform closure of $\mathcal{A}$ consists of all real continuous functions on $\mathbf{K}$ (Rudin, 1953).*

The Stone-Weierstrass Approximation Theorem has the following corollary:

Corollary. *Let $\mathbf{K}$ be a compact set and let $g : \mathbf{K} \to \mathbb{R}$ be a continuous function on $\mathbf{K}$. Then for any $\varepsilon > 0$, there exists a polynomial P such that $\|g - P\|_\infty < \varepsilon$.*

Now we restate our theorem more formally.

Theorem 1. *Let $\mathbf{K} = I_x \times I_y \times I_z \times I_t$ be a product of closed intervals in $\mathbb{R}^4$. Given any continuous function $f : \mathbf{K} \to \mathbb{R}^3$ and any $\varepsilon > 0$, there exists an SPD $\mathbf{S} : \mathbf{K} \to \mathbb{R}^3$ such that $\|f - S\|_\infty < \varepsilon$ holds.*

Proof. Let $\varepsilon > 0$. Since $f(\mathbf{X}, t) = (f_1(\mathbf{X}, t), f_2(\mathbf{X}, t), f_3(\mathbf{X}, t))$ is continuous, f_i is continuous for each $i \in \{1, 2, 3\}$. Furthermore, Heine-Borel theorem implies that $\mathbf{K}$ is compact. Therefore, the Stone–Weierstrass approximation theorem implies that polynomials uniformly approximate each component f_i. That is, there is a polynomial P_i such that $\|f_i - P_i\|_\infty < \varepsilon/\sqrt{3}$. Define $P =$

(P_1, P_2, P_3). Now, we get

$$\begin{aligned}\|f - P\|_\infty &= \sup_x \|f(x) - P(x)\|_2 \\ &= \sup_x \sqrt{\sum_i |f_i(x) - P_i(x)|^2} \\ &\leq \sup_x \sqrt{\sum_i \|f_i - P_i\|_\infty^2} \\ &< \sqrt{\sum_i \varepsilon^2/3} \\ &= \varepsilon\end{aligned} \tag{6}$$

Therefore, we can approximate f arbitrarily well with a multivariate polynomial. Thus, the space of polynomials is dense with respect to the uniform norm in the space of physically plausible functions. At a high level, the result follows because the following hierarchy holds: Polynomials $\subset$ B-Splines $\subset$ NURBS. And if a set is dense in another space, a superset of that set is also dense in that space.

More precisely, we need to show that the polynomial P can be written as an SPD. Write each P_i in the same B-Spline basis as

$$P_i(x, y, z, t) = \sum_{jkls} w^i_{jkls} N_j(x) N_k(y) N_l(z) N_s(t) \tag{7}$$

Then, we can write

$$P(x) = \left(\sum_{jkls} w^1_{jkls} B_{jkls}(x, y, z, t), \sum_{jkls} w^2_{jkls} B_{jkls}(x, y, z, t), \sum_{jkls} w^3_{jkls} B_{jkls}(x, y, z, t) \right) \tag{8}$$

$$= \sum_{jkls} \mathbf{W}_{jkls} B_{jkls}(x, y, z, t) \tag{9}$$

where $\mathbf{W}_{jkls} = (w^1_{jkls}, w^2_{jkls}, w^3_{jkls})$, and

$$B_{jkls}(x, y, z, t) := N_j(x) N_k(y) N_l(z) N_s(t).$$

Now, note the similarity between our formulation of SPD in eq. (5) and this formulation. Thus, we define the following SPD with weights taken to be equal to 1:

$$\mathbf{S}(x, y, z, t) := \sum_{jkls} R_{jkls}(x, y, z, t)\ \mathbf{W}_{jkls}, \tag{10}$$

Where we took $\mathbf{W}_{jkls}$ to be our control points and

$$R_{jkls}(x, y, z, t) = \frac{N_j(x)\, N_k(y)\, N_l(z) N_s(t)}{\sum_{jkls} N_j(x) N_k(y) N_l(z) N_s(t)} \tag{11}$$

to be our rational basis functions. Now, observe that the denominator of the rational basis function equals 1 since the weights are 1:

$$\begin{aligned}\sum_{jkls} N_j(x) N_k(y) N_l(z) N_s(t) &= \underbrace{\sum_j N_j(x)}_{=1} \underbrace{\sum_k N_k(y)}_{=1} \underbrace{\sum_l N_l(z)}_{=1} \underbrace{\sum_s N_s(t)}_{=1} \\ &= 1,\end{aligned} \tag{12}$$

which follows from the fact that B-Spline bases constitute partitions of unity. Substituting eq. (12) into eq. (11) and eq. (10), we see that our SPD is equal to the polynomial $P(x, y, z, t)$ in the B-Spline basis form given in eq. (8):

$$\mathbf{S}(x, y, z, t) = \sum_{jkls} N_j(x)\, N_k(y)\, N_l(z) N_s(t) \mathbf{W}_{jkls} \tag{13}$$

$$= P(x, y, z, t) \tag{14}$$

Thus, substituting $\mathbf{S}(x, y, z, t)$ into the polynomial approximation we proved in eq. (6), we get

$$\|f - \mathbf{S}\|_\infty < \varepsilon \tag{15}$$

which means that there is an SPD deformation that is within ε of f. □

section B shows that SPD can approximate physically plausible functions that do not contain tearing arbitrarily well. We now present a second theorem which shows that SPD can approximate physically plausible functions that do contain tearing as long as 1) The function is continuous almost everywhere. 2) The function is bounded.

The second condition is always the case for physically plausible deformations. The first condition is the case when, for example, an object is torn along a finite number of planes like cutting food with a knife.

Theorem 2. *Let $\mathbf{K} = I_x \times I_y \times I_z \times I_t$ be a product of closed intervals in $\mathbb{R}^4$. Suppose $f : \mathbf{K} \to \mathbb{R}^3$ is bounded and continuous almost everywhere. Then, for any $\varepsilon > 0$, there exists an SPD $S : \mathbf{K} \to \mathbb{R}^3$ such that $\|f - S\|_p < \varepsilon$ holds for any $1 \leq p < \infty$.*

Proof. Let $\varepsilon > 0$. Since f is almost everywhere continuous, it is a measurable function. Since $\mathbf{K}$ is a compact subset of $\mathbb{R}^4$, it has finite measure. It is a standard result in measure theory that a measurable function defined on a finite measure space is $\mathcal{L}^p$ measurable. Therefore, we have $f \in \mathcal{L}^p(\mathbf{K}, \mathbb{R}^3)$. Another standard result in real analysis is that the set of continuous functions form a dense subset in $\mathcal{L}^p(\mathbf{K}, \mathbb{R}^3)$. Then, $f \in \mathcal{L}^p(\mathbf{K}, \mathbb{R}^3)$ implies that we can find a continuous function g that is arbitrarily close to f, in particular

$$\|f - g\|_p < \frac{\varepsilon}{2}. \tag{16}$$

Since g is a continuous function on a compact set, it satisfies the assumptions of theorem 1. Hence, we can find $S \in \mathrm{SPD}$ that is arbitrarily close to g in the sup-norm, in particular:

$$\|g - S\|_\infty < \frac{\varepsilon}{2\mu(\mathbf{K})^{1/p}}. \tag{17}$$

We can also bound the $\mathcal{L}^p$ distance between g and S with their sup-distance:

$$\begin{aligned} \|g - S\|_p &= \left(\int_{\mathbf{K}} |g - S|^p d\mu \right)^{1/p} \\ &\leq \left(\int_{\mathbf{K}} \|g - S\|_\infty^p \, d\mu \right)^{1/p} \\ &= \left(\|g - S\|_\infty^p \int_{\mathbf{K}} d\mu \right)^{1/p} \\ &= \|g - S\|_\infty \, \mu(\mathbf{K})^{1/p} \end{aligned} \tag{18}$$

Using the triangle inequality and the inequalities eq. (16), eq. (17), eq. (18) we get,

$$\begin{aligned} \|f - S\|_p &\leq \|f - g\|_p + \|g - S\|_p \\ &< \frac{\varepsilon}{2} + \|g - S\|_\infty \, \mu(\mathbf{K})^{1/p} \\ &\leq \frac{\varepsilon}{2} + \frac{\varepsilon}{2} \\ &= \varepsilon \end{aligned} \tag{19}$$

which means that there is an SPD arbitrarily close to f in the $\mathcal{L}^p$ norm. □

C More Details about the Dataset

DancingThings consists of 23 scenes, each with 100 frames per scene, having in a total of 2376 pairs. Some extra dataset statistics can be seen in Table 7. Note that although the median optical flow is around 35 for valid pixels, the average optical flow is around 46, which means that the distribution

of motion has high variation across the scenes, providing an appropriately challenging setting for optical flow methods.

Table 7: Dataset statistics for DancingThings.

Category	Number
Number of image pairs	2376
Avg number of deformed objects per scene	36
Frames per scene	100
Total number of scenes	23
Avg optical flow per scene (all)	57.09
Median optical flow per scene (all)	38.78
Avg optical flow per scene (valid)	45.87
Median optical flow per scene (valid)	35.46

DancingThings starts from a pre-existing scene generated from Infinigen Indoors (Raistrick et al., 2024), then animates and procedurally deforms its objects using our SPD method. Fig. 5 shows that SPD leaves Infinigen materials consistent through the deformation process.



Figure 5: The textures of the objects stay consistent across time as the mesh is deformed by SPD. We modify the materials of the assets from Infinigen Indoors (Raistrick et al., 2024) to ensure this is the case.

D SPD vs Physically Plausible Simulation

We also compare our SPD directly against a physically plausible simulation. We generate a set individual assets and we generate two datasets of $6K$ frames each, one with SPD and one with soft-body simulation in Blender (ble). We train SEA-RAFT on these datasets and evaluate on Spring Non-Rigid. Tab. 8 shows that using SPD is better compared to soft-body simulation.

Table 8: Comparison of Soft Body Simulation and SPD when applied to individually generated assets.

Model	Trained	Spring Non-Rigid (val)			
		EPE ↓	1 px	3 px	5 px
SEA-RAFT	Soft Body	4.62	47.80	24.56	17.03
SEA-RAFT	**SPD**	**3.73**	**36.56**	**16.55**	**11.49**

E Additional Experiments

In the main paper we showed that models when trained on DancingThings perform better on both rigid benchmarks like KITTI (Menze & Geiger, 2015) and non rigid motion benchmarks like Spring Non Rigid (Mehl et al., 2023), Sintel (Butler et al., 2012) and DOT (Li et al., 2025)

The DOT dataset is a real-world benchmark for tracking deformable objects. It contains deformations from four main categories: rope, cloth, paper and hands. For each category, the authors provide multiple sequences together with ground truth 2D and 3D correspondences. Specifically, they provide 20 scenes for paper, 30 scenes for cloth, 70 scenes for rope and 90 scenes for hands (Li et al., 2025).

MPI Sintel Flow Dataset and Spring dataset are two high resolution synthetic datasets commonly used for optical flow evaluation (Butler et al., 2012; Mehl et al., 2023). For Spring (Mehl et al., 2023), in addition to ground truth optical flow, the authors provide a rigidity map, which splits each frame to rigid and non rigid pixels. By converting this rigid map into a boolean mask and applying it to the optical flow ground truth, we obtain a non rigid validation set.

For our experiments we split randomly the official training scenes of KITTI (Menze & Geiger, 2015), Sintel (Butler et al., 2012), Spring (Mehl et al., 2023) and DOT (Li et al., 2025) on a train, validation, and test set. Specifically the official 200 training scenes KITTI are randomly split into 120 train, 40 val and 40 test and the 23 official training scenes of Sintel are split into 13 train, 5 val and 5 test scenes. For Spring the 37 official training scenes are split into 25 train, 6 val and 6 test. Lastly, the DOT official 120 scenes are split into 69 train scenes, 26 val and 25 test.

For all the experiments, the models were trained on 4 NVIDIA RTX 3090 GPUs with a batch size of 16, using the training protocol as in SEA-RAFT (Wang et al., 2024) and WAFT (Wang & Deng, 2026).

We provide some additional results on the DOT dataset (Li et al., 2025). As we mentioned in the main paper DOT dataset (Li et al., 2025) consists of 30 Cloth, 20 Paper and 70 Rope deformable real-world scenes with different texture. We trained SEA-RAFT (Wang et al., 2024) on three datasets seperately: TartanAir (Wang et al., 2020) + FlyingChairs (Dosovitskiy et al., 2015b) , TartanAir (Wang et al., 2020) + FlyingThings3D (Mayer et al., 2016) + FlyingChairs (Dosovitskiy et al., 2015b) and DancingThings. We compare the performance of SEA-RAFT (Wang et al., 2024) trained on each of these three datasets by evaluating the EPE on DOT (Li et al., 2025) train scenes.

The results on some randomly-chosen Paper scenes from DOT (Li et al., 2025) train, including those reported in the DOT paper (Li et al., 2025), are shown in Table 9. In the majority of the scenes, when SEA-RAFT (Wang et al., 2024) is trained on DancingThings it achieves superior performance. In addition, since the number of deformable Rope scenes is larger than both Cloth and Paper scenes, we show more random results in Tables 10, 11, 12, 13, including the Rope scenes reported in the DOT paper (Li et al., 2025). In these examples we can see that DancingThings gives also large reductions in the EPE. Specifically, compared to TartanAir (Wang et al., 2020) + FlyingThings3D (Mayer et al., 2016) + FlyingChairs (Dosovitskiy et al., 2015b), DancingThings, despite 138 $\times$ smaller in size, achieves up to a 32.9% reduction in EPE depending on the sequence, while compared to TartanAir (Wang et al., 2020) + FlyingChairs (Dosovitskiy et al., 2015b) it achieves up to 57.9% reduction in EPE, with the best performance on R17.

We also show results on some randomly-chosen Cloth scenes from DOT (Li et al., 2025), including those reported in the DOT paper (Li et al., 2025), in Table 14. We see that DancingThings outperforms both TartanAir (Wang et al., 2020) + FlyingThings3D (Mayer et al., 2016) and TartanAir (Wang et al., 2020) + FlyingThings3D (Mayer et al., 2016) + FlyingChairs (Dosovitskiy et al., 2015b) in the majority of scenes.

Overall, as we showed in the main paper the average performance on the DOT dataset is higher for DancingThings than both TartanAir (Wang et al., 2020) + FlyingChairs (Dosovitskiy et al., 2015b) and TartanAir (Wang et al., 2020) + FlyingThings3D (Mayer et al., 2016) + FlyingChairs (Dosovitskiy et al., 2015b) showing that optical flow models trained on DancingThings generalize better to complex real-world deformable scenes.

Table 9: Per-scene EPE (↓) on the following Paper scenes from DOT dataset (Li et al., 2025): P02, P04, P05, P06, P07, P08, P10, P11, P12, P13, P14, P16, P17. T and C in the "Trained" column denote FlyingThings3D (Mayer et al., 2016) and FlyingChairs (Dosovitskiy et al., 2015b) respectively. **Bold** denotes the best, underlined denotes the second best results.

Model	Trained	P02	P04	P05	P06	P07	P08	P10	P11	P12	P13	P14	P16	P17
SEA-RAFT	TartanAir + C	0.66	0.63	0.56	0.62	0.55	0.60	0.54	0.65	0.70	0.82	0.69	0.61	0.58
SEA-RAFT	TartanAir + T + C	0.92	0.55	0.51	0.51	0.47	0.51	1.20	1.28	0.75	1.87	1.17	2.30	0.55
SEA-RAFT	DancingThings	**0.48**	**0.48**	**0.47**	**0.47**	**0.44**	**0.49**	**0.48**	**0.51**	**0.57**	**0.59**	**0.51**	**0.52**	**0.52**

Table 10: Per-scene EPE (↓) on the following Rope scenes from DOT dataset (Li et al., 2025): R01, R03, R05, R06, R09, R11, R12, R14. T and C in the "Trained" column denote FlyingThings3D (Mayer et al., 2016) and FlyingChairs (Dosovitskiy et al., 2015b) respectively.

Model	Trained	R01	R03	R05	R06	R09	R11	R12	R14
SEA-RAFT	TartanAir + C	0.95	0.96	1.86	1.18	2.22	1.41	1.53	1.14
SEA-RAFT	TartanAir + T + C	0.61	0.68	0.88	0.65	0.83	0.77	0.63	0.70
SEA-RAFT	DancingThings	**0.61**	0.72	**0.86**	0.72	1.03	0.81	0.70	**0.68**

Table 11: Per-scene EPE (↓) on the following Rope scenes from DOT dataset (Li et al., 2025): R15, R16, R17, R18, R19, R23, R24, R25, R26. T and C in the "Trained" column denote FlyingThings3D (Mayer et al., 2016) and FlyingChairs (Dosovitskiy et al., 2015b) respectively. **Bold** denotes the best, underlined denotes the second best results.

Model	Trained	R15	R16	R17	R18	R19	R23	R24	R25	R26
SEA-RAFT	TartanAir + C	1.10	0.89	2.35	0.58	1.37	1.07	0.79	0.75	0.90
SEA-RAFT	TartanAir + T + C	0.69	0.68	0.79	0.79	0.78	0.62	0.61	0.68	0.71
SEA-RAFT	DancingThings	0.75	**0.65**	**0.99**	**0.53**	0.79	0.66	**0.61**	**0.65**	**0.66**

Table 12: Per-scene EPE (↓) on the following Rope scenes from DOT dataset (Li et al., 2025): R27, R29, R33, R34, R35, R36, R37, R38, R40. T and C in the "Trained" column denote FlyingThings3D (Mayer et al., 2016) and FlyingChairs (Dosovitskiy et al., 2015b) respectively.

Model	Trained	R27	R29	R33	R34	R35	R36	R37	R38	R40
SEA-RAFT	TartanAir + C	0.84	1.03	0.88	0.88	1.13	1.18	0.57	0.99	1.10
SEA-RAFT	TartanAir + T + C	0.65	0.72	0.64	0.70	0.78	0.88	0.45	0.60	0.66
SEA-RAFT	DancingThings	**0.61**	**0.70**	**0.61**	**0.62**	**0.74**	**0.81**	**0.44**	0.63	0.75

Table 13: Per-scene EPE (↓) on the following Rope scenes from DOT dataset (Li et al., 2025): R45, R49, R52, R54, R55, R62, R63, R64, R65. T and C in the "Trained" column denote FlyingThings3D (Mayer et al., 2016) and FlyingChairs (Dosovitskiy et al., 2015b) respectively.

Model	Trained	R45	R49	R52	R54	R55	R62	R63	R64	R65
SEA-RAFT	TartanAir + C	1.46	0.85	1.18	1.02	0.80	0.75	0.64	1.09	0.88
SEA-RAFT	TartanAir + T + C	0.69	0.61	0.78	0.57	0.61	0.56	0.52	0.66	0.64
SEA-RAFT	DancingThings	0.77	**0.56**	0.80	0.60	**0.58**	**0.56**	0.53	**0.65**	**0.61**

Table 14: Per-scene EPE (↓) on the following Cloth scenes from DOT dataset (Li et al., 2025): C02, C03, C04, C05, C06, C09, C10, C16, C20, C25, C30. T and C in the "Trained" column denote FlyingThings3D (Mayer et al., 2016) and FlyingChairs (Dosovitskiy et al., 2015b) respectively. **Bold** denotes the best, underlined denotes the second best results.

Model	Trained	C02	C03	C04	C05	C06	C09	C10	C16	C20	C25	C30
SEA-RAFT	TartanAir + C	0.52	0.55	0.62	0.63	0.76	0.48	0.68	0.51	0.50	0.53	0.51
SEA-RAFT	TartanAir + T + C	0.49	0.66	0.65	0.67	0.67	0.47	0.64	0.48	0.47	0.49	0.48
SEA-RAFT	DancingThings	**0.49**	**0.54**	**0.54**	**0.56**	**0.57**	0.48	**0.58**	**0.48**	**0.46**	**0.49**	0.48

F Occlusion Masks

We compute the occlusion masks using the ground truth depth and 3D flow for every pixel generated from the Infinigen-Indoors (Raistrick et al., 2024). For each pixel (x, y) in frame t, we have its depth value $d_t(x, y)$ and its 3D flow from t to $t + 1$

$$\text{flow}_{3D} = [u, v, \Delta z] \tag{20}$$



Figure 6: An illustration of our ground truth. Each row from left to right shows: image at frame t, image at frame $t+1$, optical flow for all the pixels, and occlusion masks. Note that the occlusion masks include pixels that appear from t to $t+1$ and disappear from t to $t+1$. Both pixels that move in and out of the field-of-view as well as pixels that go behind an object are masked.

We also have the depth map d_{t+1} for all pixels in frame $t+1$. We project each point from frame t to $t+1$ using 20: $x_2 = x + u, y_2 = y + v$

If the projected pixel (x_2, y_2) is outside the image, then this pixel is considered invalid, and as a result it is masked. For the rest, we get the depth value $z(x_2, y_2)$ by interpolating neighboring values using the d_{t+1} . We then compare $z(x_2, y_2)$ with the estimated depth of the source pixel in $t+1$

$$z_{est} = d_t(x, y) + \Delta z \tag{21}$$

If $z(x_2, y_2) \geq z_{est}$ then that pixel is visible in $t+1$ and it is considered valid. On the other hand, if $z(x_2, y_2) < z_{est}$, then another object is in front of that pixel and as a result the pixel is invalid and is marked as occluded. An illustration of the occlusion masks can be seen in Fig. 6 where we can see examples from two different scenes. As shown, both pixels that go inside or outside the field of view and the pixels that get occluded or disoccluded are masked as invalid.

G Additional Information about Ablation Study

In the main paper we performed an ablation study in order to analyze the paramaters of SPD. Specifically, we generate a set of 8217 frame pairs, we train SEA-RAFT (Wang et al., 2024) on 9 different versions of the same scenes evaluate its results on the Spring non-rigid validation set. In the first case *static*, all the assets are either static. In the second one, *rigid motion*, the assets are translated and rotated around the room. In the final two cases, render in both Eevee, and Cycles. The first two cases are rendered with Eevee. As we can see from the table 6, the lowest EPE and the lowest px errors are achieved when we deform the objects using SPD and render the frames with Eevee. Applying rigid motion to the objects also has a positive effect on the results.

Apart from the ablation for the SPD parameters, we conduct an additional one, in order to analyze the design choices of DancingThings. We generate 2376 pairs and we ablate the significance of motion blur, texture on the objects and camera motion. Each version of the same scenes is trained on SEA-RAFT and evaluated on Spring Non Rigid (val) set. Initially the generated scenes have no motion blur at all. As we can see from Table 15, when we render scenes with motion blur the EPE drops with the best perfromance happening when shutter is equal to 0.3 . Keeping shutter value at 0.3, we then double the speed of the camera. As we can see the result is slightly worse but still close to the initial one. However, the optical flow datasets in general have different camera motion speeds. In order to cover both slow and fast camera motion cases we generalize our formula and instead of a uniform distribution we use logUniform. Also not all the datasets include scenes with motion blur. In order to cover also here all the cases, half of our scenes are rendered with no motion blur at all and the rest with a shutter value from 0 to 0.3. We chose 0.3 since when we increased it to 0.5 the results became worse. Lastly, we check the importance of texture. We ablate two different types of textures. In the first case, we apply on the walls, floor, ceiling and objects texture from SimpleProc (Ma et al., 2026) and in the secong case from the Describable Textures Dataset (DTD) (Cimpoi et al., 2014). As we can see when we apply texture the EPE significantly drops. Since it is lower in the case of DTD, we generate DancingThings applying DTD texture on objects, walls, ceilings and floors and then we deform them.

Table 15: Ablation study on DancingThings evaluating on Spring Non Rigid (val) (Mehl et al., 2023) different techniques. All rows use the SEA-RAFT (Wang et al., 2024) model.

Blur	Texture	Camera Motion	EPE ↓
✗	Original Infinigen Materials	U(1,2)	3.77
Shutter 0.15	Original Infinigen Materials	U(1,2)	3.66
Shutter 0.3	Original Infinigen Materials	U(1,2)	3.54
Shutter 0.5	Original Infinigen Materials	U(1,2)	3.73
Shutter 0.3	Original Infinigen Materials	U(2,4)	3.57
Clamp(N(0, 0.2), 0, 0.30)	SimpleProc (Ma et al., 2026)	logU(1,4)	3.12
Clamp(N(0, 0.2), 0, 0.30)	DTD (Cimpoi et al., 2014)	logU(1,4)	3.10

H Compute Resources

For the generation of DancingThings each scene uses 1 NVIDIA GPU (RTX 3090, A6000 or A40). The memory per unit is 11 GB. Regarding the CPUs, the model was an Intel Xeon Gold 5220. The number of CPU cores used was 4-8 per job.

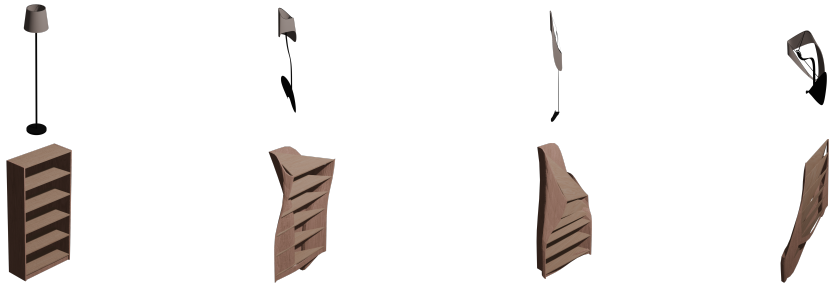


Figure 7: Gradual deformation of different objects by SPD. Each row: rest pose at $t_0 \rightarrow$ deformation at $t_1 \rightarrow$ deformation at $t_2 \rightarrow$ deformation at t_3.

I Additional Images

Another example of a gradual deformation on different objects is shown in Fig. 7.

Some other random images from DancingThings are depicted in Fig. 8.

J Limitations

Our dataset is currently limited to indoor scenes; however, our data generation method works with any collection of meshes, which means that one could generate outdoor scenes using our pipeline as well. Another limitation is that SPD deformations are not guaranteed to be physically plausible as discussed in section 4.4. This is not an issue for optical flow, but could be an issue for other tasks.

K AI use Statement

We used LLMs for coding, assisting in proofs, and for literature search. We did not use LLMs for writing the paper or polishing it. We have reviewed all LLM-assisted work manually. Meaning that we have read, reviewed, and evaluated LLM generated code, proof-steps, as well as related work. We take full responsibility for all the LLM generated parts of this project.

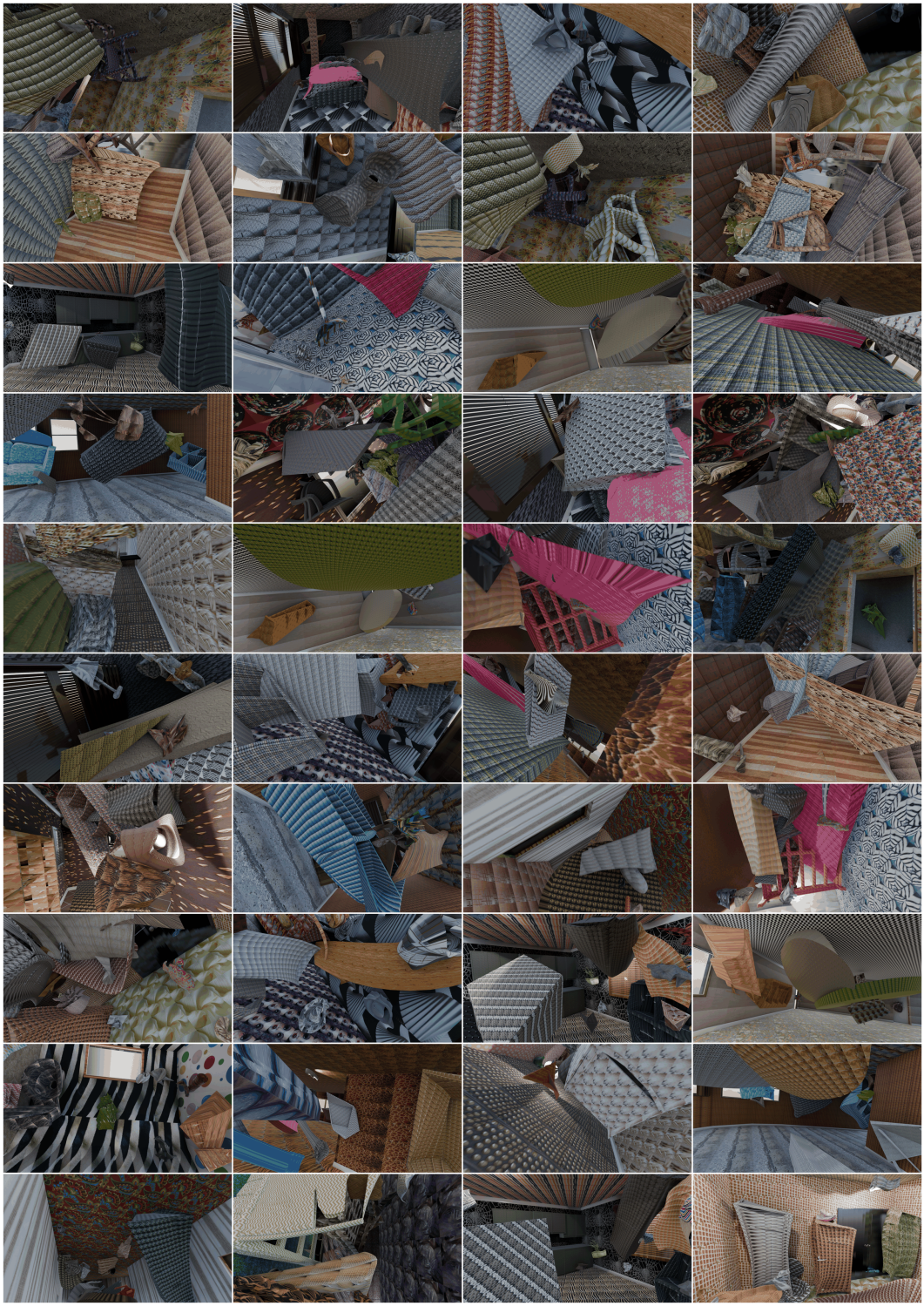


Figure 8: Random images from DancingThings