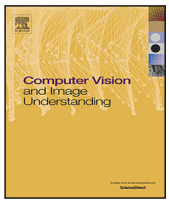
Computer Vision and Image Understanding 249 (2024) 104155

Contents lists available at ScienceDirect

Computer Vision and Image Understanding

journal homepage: www.elsevier.com/locate/cviu



Deformable surface reconstruction via Riemannian metric preservation

Oriol Barbany *, Adrià Colomé, Carme Torras

Institut de Robòtica i Informàtica Industrial (CSIC-UPC), C/ Llorens i Artigas 4-6, Barcelona, 08028, Spain

ARTICLE INFO

Communicated by Akihiro Sugimoto

Keywords:
Shape-from-template
3D reconstruction
Deformable surfaces
Differential geometry
Surface parametrization

ABSTRACT

Estimating the pose of an object from a monocular image is a fundamental inverse problem in computer vision. Due to its ill-posed nature, solving this problem requires incorporating deformation priors. In practice, many materials do not perceptibly shrink or extend when manipulated, constituting a reliable and well-known prior. Mathematically, this translates to the preservation of the Riemannian metric. Neural networks offer the perfect playground to solve the surface reconstruction problem as they can approximate surfaces with arbitrary precision and allow the computation of differential geometry quantities. This paper presents an approach for inferring continuous deformable surfaces from a sequence of images, which is benchmarked against several techniques and achieves state-of-the-art performance without the need for offline training. Being a method that performs per-frame optimization, our method can refine its estimates, contrary to those based on performing a single inference step. Despite enforcing differential geometry constraints at each update, our approach is the fastest of all the tested optimization-based methods.

1. Introduction

The problem of inferring the shape of a generic non-rigid object from a sequence of monocular images given a 3D template of it is known as Shape from Template (SfT) (Bartoli et al., 2015). SfT appears in areas like robotic manipulation (Bodenhagen et al., 2014), in which perceiving the state of deformable objects is one of the bottlenecks often pinpointed as hindering their manipulation by robots (Sanchez et al., 2018). In this area, estimating the cloth state is crucial to simulate its evolution under manipulation (Coltraro et al., 2022) and apply model-predictive control (Luque et al., 2024) to guide the robots.

The SfT problem is inherently ill-posed as it permits infinitely many solutions leading to accurate projection to the input 2D images (Tretschk et al., 2023). Thus, solving it requires the incorporation of deformation priors, isometry being one of the most common (Brunet et al., 2010; Bartoli et al., 2015, 2012; Parashar et al., 2015; Chhatkuli et al., 2017; Casillas-Pérez et al., 2021). Isometry models surfaces that cannot extend or shrink, a condition often not restrictive in practice as many deformable objects made of materials such as cloth and paper are nearly inextensible (Salzmann and Fua, 2011). Although this constitutes a powerful and widely applicable prior, it defines a differential equation that, in most cases, has to be approximated to make the problem computationally tractable.

In this work, we propose to encode the parametric equations of the object of interest in the weights of a neural network. Doing so yields a continuous surface contrasting with the discrete representations used in previous works. To estimate the parametric equations, we perform an iterative procedure enforcing three soft constraints accounting for a correct surface projection, the preservation of the surface metric (equivalent to the isometry constraint), and temporal consistency with the previous surface. We depict the presented method in Fig. 1.

The isometry assumption is equivalent to the hypothesis that the geodesics on the surface may not change their length over time. Following this assumption, we compute the surface metric analytically using the learned parametric equations and impose its preservation as a soft constraint. This allows for a fast estimation of the surface, which is in stark contrast with previous methods struggling to optimize non-convex objectives and considering differential equations. While previous works used neural parametric representations for single-view reconstruction (Groueix et al., 2018; Bednarík et al., 2020), this is the first method to employ them without offline training and in conjunction with the popular isometry constraint.

We evaluate the proposed method against five approaches and a baseline introduced in this paper on two publicly available datasets. The quantitative results showcase the effectiveness and robustness of the proposed solution, which is also the fastest of all the tested optimization-based approaches despite enforcing isometry.

Our main contributions are:

- We combine the powerful and widely applicable assumption of metric preservation with the representation power of neural networks. In particular, we learn a surface parametrization (Groueix

* Corresponding author.
E-mail address: obarbany@iri.upc.edu (O. Barbany).

https://doi.org/10.1016/j.cviu.2024.104155
Received 17 March 2023; Received in revised form 24 July 2024; Accepted 4 September 2024
Available online 19 September 2024

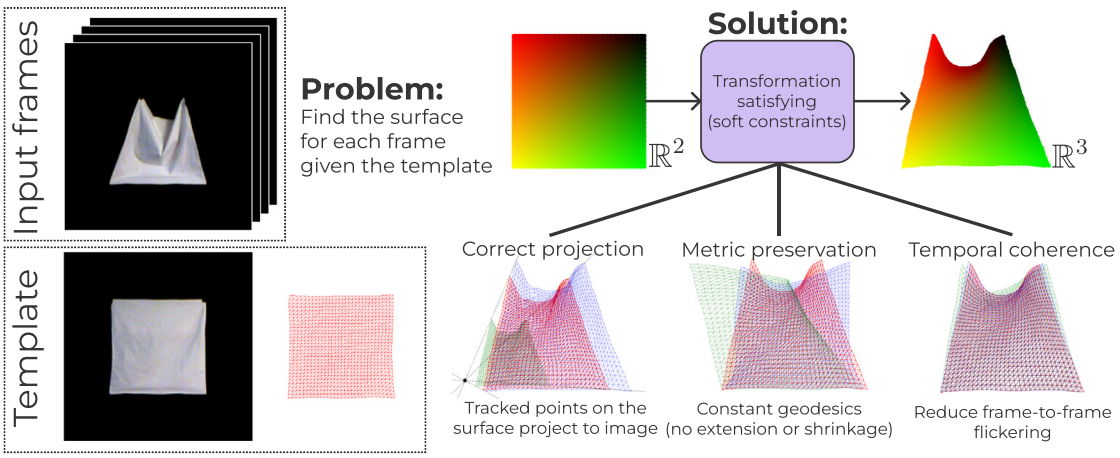


Fig. 1. Summary of the presented method. Given a template consisting of a mesh and an image showing such a surface, we want to recover deformations of the same surface as it appears in the input frames. We propose describing the surface by a learned parametric equation, which maps a point in $\mathbb{R}^2$, uniquely defining a point on the surface, to its spatial coordinates in $\mathbb{R}^3$ (in the example shown above, this mapping preserves the colors, which we add for easier visualization). To find such a parametric equation, we impose three soft constraints that alone are insufficient but, when combined, allow us to recover the surface. We illustrate the ambiguities of each condition by showing feasible solutions that are either correct (in red) or incorrect (in blue and green). The constraints and ambiguities are: (1) The surface correctly projects to the input image, a condition satisfied by any surface obtained after arbitrarily moving the tracked points along the line of sight. (2) The geodesic distances are preserved, a condition satisfied for any isometric transformation of the template surface. (3) The surface does not vary too much in consecutive frames, which is compatible with all surfaces whose distance between the surface from the previous frames is small.

et al., 2018), which allows **representing continuous surfaces** rather than discretized representations (*e.g.*, point clouds, voxels, or meshes).
- We advocate for imposing **metric preservation of the surface as a soft constraint**, which accounts for the fact that physical quantities are not preserved exactly (Alet et al., 2021a,b; Salzmann et al., 2008a).
- We learn a parametric surface during inference and **without an offline training process**. Hence, the method requires neither a dataset nor fine-tuning to new sequences, materials, or templates.

The rest of this paper is structured as follows. In Section 2, we review previous approaches to the SfT problem. Then, we introduce the proposed method in Section 3 and show its qualitative and quantitative performance in Section 4. Finally, we conclude the paper in Section 5.

2. Related work

In this section, we review several approaches to the SfT problem and group them under three umbrellas: the methods that use hand-crafted constraints to define the manifold of surfaces, those that infer the deformation models from data, and the approaches that combine analytical and data-driven models.

Table 1 presents a summary of the discussed papers.

2.1. Hand-crafted constraints

One approach to SfT is to use explicit properties of the tracked surface to determine feasible shapes. While these methods result in easily modifiable constraints, some equations may be too simplistic for accurately describing complex and nonlinear physics of real-world surfaces (Salzmann et al., 2008a).

Assuming constant Riemannian metric allows for unambiguous surface reconstruction (Bartoli et al., 2012, Theorem 1). Metric preservation can be enforced analytically by leveraging a differentiable warp (Bartoli et al., 2012, 2015; Parashar et al., 2015; Chhatkuli et al., 2017; Casillas-Pérez et al., 2021), but this struggles with sharp folds and is undefined in occluded areas. Relaxations using equality constraints on Euclidean norms (Salzmann et al., 2008a, 2007c) are incompatible with sharp folds, and inequality constraints (Salzmann and Fua, 2009, 2011; Salzmann et al., 2008b) are prone to vertex collapsing.

Laplacian meshes constrain the problem by reducing the number of free parameters (Ngo et al., 2016; Wang et al., 2019), but lose resolution on the edges of the surface and yields 3D shape estimates that re-project to the image but are not necessarily accurate.

For high enough frame rates, the tracked object barely changes in neighboring frames. Some works incorporate this constraint in the objective by minimizing the difference with the previous shape (Yu et al., 2015), a window of past surfaces (Salzmann and Fua, 2009), second derivatives of surface parameters (Salzmann et al., 2007c), or the change of edge orientation (Salzmann et al., 2007a), and others use it to provide good initializations for future shapes (Ngo et al., 2015, 2016; Kairanda et al., 2022). Enforcing temporal smoothness helps recover surfaces with severe deformations and reduces jitter.

2.2. Data-based constraints

Data-based approaches use statistical learning techniques to learn the manifold of plausible surfaces. However, they require a dataset of surface configurations that should ideally represent all deformations and be representative of the surface dynamics. These approaches may also require different datasets for each material, surface, lighting conditions, and mesh resolution.

Previous works used PCA (Salzmann and Fua, 2011; Salzmann et al., 2008a, 2007c), sparse GP-LVMs (Salzmann et al., 2008b), and constrained latent variable models (Varol et al., 2012). In these approaches, the dimensionality of the latent space may depend on the material, and the models can yield extensible surfaces even if the dataset only consists of inextensible surfaces (Salzmann et al., 2007c).

Neural networks can be used to predict surface vertices (Bednarík et al., 2018; Pumarola et al., 2018) or depth maps (Fuentes-Jimenez et al., 2022, 2021). The former only allow prohibitively small mesh sizes, *e.g.* 10×10 in Pumarola et al. (2018), and the latter needs post-processing like ARAP (Sorkine and Alexa, 2007) to obtain occluded areas.

An interesting approach is to learn the parametric equations of the surface depicted in the input image (Groueix et al., 2018; Bednarík et al., 2020), which can generate continuous surfaces. These approaches train on pairs of images and their point clouds. Hence they are also object-specific.

Table 1
Summary of related work. Our method is the first to satisfy all the listed desirable properties for the reconstruction of deformable surfaces.

Method	Works with non-planar template	Models sharp folds	Does not need dataset	Temporal coherence	Variable mesh resolution
Hand-crafted constraints (Section 2.1)					
Iterative isometric surfaces (Brunet et al., 2010)	✗	✓	✓	✗	✗
Closed-form isometric surfaces (I) (Bartoli et al., 2015, 2012)	✓	✗	✓	✗	✗
Closed-form isometric surfaces (II) (Parashar et al., 2015; Chhatkuli et al., 2017; Casillas-Pérez et al., 2021)	✓	✓	✓	✗	✗
Iterative constant Euclid. Salzmann et al. (2007c)	✗	✗	✗	✓	✗
Closed-form constant Euclid. Salzmann et al. (2008a)	✗	✗	✗	✗	✗
Inextensible surfaces (Salzmann and Fua, 2009)	✗	✓	✓	✗	✗
Dense registration (Ngo et al., 2015)	✗	✓	✓	✓	✗
Laplacian meshes (Ngo et al., 2016; Wang et al., 2019)	✓	✓	✓	✓	✗
Edge orientation changes (Salzmann et al., 2007a)	✓	✓	✓	✓	✗
Vertex coordinate changes (Yu et al., 2015)	✓	✓	✓	✓	✗
Data-based constraints (Section 2.2)					
Latent space (Salzmann and Fua, 2011; Salzmann et al., 2008b; Varol et al., 2012)	✓	✓	✗	✗	✗
Neural network: Image to vertices/depth (Bednarík et al., 2018; Fuentes-Jimenez et al., 2022, 2021; Pumarola et al., 2018)	✓	✓	✗	✗	✗
Surface parametrization (Groueix et al., 2018; Bednarík et al., 2020)	✓	✓	✗	✗	✓
Hybrid approaches (Section 2.3)					
GP constant Euclid. Salzmann and Urtasun (2010)	✗	✗	✗	✗	✗
Differentiable physics simulator and renderer (Kairanda et al., 2022)	✓	✓	✓	✓	✗
Linear (ours)	✓	✓	✓	✓	✓
Neural (ours)	✓	✓	✓	✓	✓

2.3. Hybrid approaches

A promising research direction highlighted in the context of perception for robotic cloth manipulation (Yin et al., 2021) is to combine analytical and data-driven models. Salzmann and Urtasun (2010) used GPs implicitly satisfying a set of quadratic equality constraints. However, this method requires all training examples to satisfy all the constraints and is overly simplistic for sharply folding materials like clothes (Varol et al., 2012).

Kairanda et al. (2022) proposed integrating a differentiable physics simulator and renderer. The drawbacks of this method are that it imposes hard constraints through the physics simulator, requires 16–24 h on a GPU to process an image sequence, and uses a fixed resolution of around 300 vertices, which prevents capturing fine wrinkles (Kairanda et al., 2022).

This work falls into this category, as we combine a data-driven method, namely neural networks, with equations to constrain the surface dynamics. Concretely, we propose using the dynamic equations originally derived to simulate clothes in Coltraro et al. (2022) and then solve the inverse problem. Concretely, we adapt the model for textile manipulation by Coltraro et al. (2022) based on metric preservation.

3. Methodology

In this section, we first introduce the problem notation and some preliminaries. Then, we integrate the relaxation of the isometry constraint developed for a cloth simulator (Coltraro et al., 2022) into an optimization scheme with linear objective and linear constraint. This method is used as a baseline and dubbed Linear (ours). Finally, we introduce the method that constitutes the main contribution of this paper, which we refer to as Neural (ours).

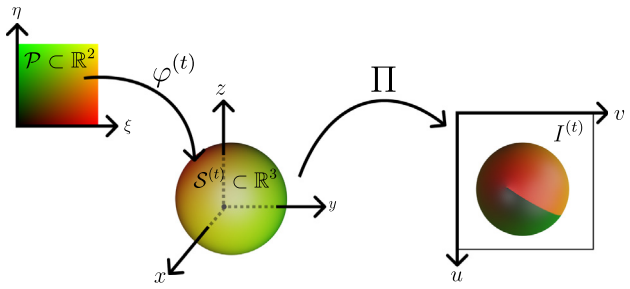


Fig. 2. Problem notation. The parametric equations $\varphi^{(t)}$ map the input domain $\mathcal{P} \subset \mathbb{R}^2$, in this case, the interior of a unit square, to the 3D coordinates of $S^{(t)}$, the surface at time t. The image $I^{(t)}$ is obtained by projecting $S^{(t)}$ with Π.

3.1. Preliminaries

The first fundamental form of a surface S allows measuring curve lengths, angles of tangent vectors, and areas of regions on it (Do Carmo, 2016). Using a parametrization $\varphi : \mathcal{P} \subset \mathbb{R}^2 \to \mathbb{R}^3$, the Riemannian metric of S is then uniquely defined by the metric tensor $\mathbf{J}_\varphi^T \mathbf{J}_\varphi$, where $\mathbf{J}_\varphi$ is the Jacobian matrix of the map φ. We focus on modeling surfaces in such a way that we preserve the Riemannian metric. That is, at all times, the length of any curve inside the surface remains constant.

Assumption 1. The sequence of monocular images used as input in the SfT problem is obtained with a calibrated camera with known intrinsic parameters.

Following the pinhole camera model, given a point $\mathbf{s} = (x, y, z) \in S$, which w.l.o.g. we assume to be expressed in the camera referential, we can compute the position (u, v) in the image captured by the camera as follows:

$$d \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \mathbf{K} \begin{bmatrix} x \\ y \\ z \end{bmatrix}, \tag{1}$$

where $\mathbf{K} \in \mathbb{R}^{3\times3}$ is the intrinsic matrix, known according to Assumption 1, a common assumption in SfT, and d is the depth along the line of sight. In the following, let $\Pi(\mathbf{s}) := (u, v)$. The camera referential assumption is satisfied without losing generality in case the camera is fixed or the relative motion is known and properly compensated. We include a summary of notation in Fig. 2.

Problem. Given a sequence of images $\{I^{(t)}\}_{t\in[T]}$, where $[T] := \{1, \dots, T\}$, and a template surface $\mathcal{T}$, estimate the surface $S^{(t)}$ at any time t.

3.2. Linear approach

Let $\mathcal{M}_S := (\mathcal{V}, \mathcal{E}, \mathcal{F})$ be a triangle mesh that approximates the smooth surface S with n vertices $\mathcal{V} := \{\mathbf{v}_i\}_{i\in[n_v]}$, edges $\mathcal{E} \subseteq \mathcal{V}^2$ and triangular faces $\mathcal{F} \subseteq \mathcal{V}^3$.

A typical strategy to formulate the SfT problem is to proceed in two steps. First, in the registration step, we find 3D-to −2D correspondences

$$M^{(t)} := \{(\mathbf{s}, \mathbf{i}) : \mathbf{s} \in S^{(t)}, \mathbf{i} = \Pi(\mathbf{s}) \in I^{(t)}\}, \tag{2}$$

where $\Pi(\mathbf{s}) := \mathbf{i} \in \mathbb{R}^2$ is the projection of $\mathbf{s}$ into the image plane.

Then, in the reconstruction step, we obtain the deformed shape, which requires inferring the depth of the points $\mathbf{i}$ in $M^{(t)}$.

Let $\mathbf{x}^{(t)} := \text{vec}\left(\begin{bmatrix} \mathbf{v}_1^{(t)} & \cdots & \mathbf{v}_N^{(t)} \end{bmatrix}\right)$. The projection constraints given by $M^{(t)}$ can be expressed as

$$\mathbf{M}^{(t)} \mathbf{x}^{(t)} = \mathbf{0}, \tag{3}$$

where $\mathbf{M}^{(t)} \in \mathbb{R}^{2|M^{(t)}|\times3n_v}$ expresses the correspondences employing barycentric coordinates (Salzmann et al., 2007b).

To compute $\mathbf{M}^{(t)}$, the tracked points on the surface are expressed using barycentric coordinates. Recall that, given a point $\mathbf{p}$ belonging to facet f of $\mathcal{M}_S$, we have that $\mathbf{p} = \sum_{i\in[3]} b_i \mathbf{v}_{f,i}$, where $\sum_{i\in[3]} b_i = 1$ and $\{b_i\}_{i\in[3]}$ and $b_i \geq 0$ are the barycentric coordinates of $\mathbf{p}$. If the $j-$th match in $M^{(t)}$ links the previous point $\mathbf{p}$ to an image coordinate $\mathbf{i}$, this correspondence is encoded in $\mathbf{M}^{(t)}$ as

$$\mathbf{M}^{(t)}[2j-1, 2j \;;\; \text{Index}(\mathbf{v}_{f,i})] := b_i(\mathbf{K}[1,2] - \mathbf{i} \otimes \mathbf{K}[3]), \tag{4}$$

where we define submatrices as [row indices ; column indices]. In case column indices are not specified, all columns are selected. For convenience, we define the Index operator, which returns the indices of the x, y, z coordinates of a given vertex according to the vectorization used to define $\mathbf{x}^{(t)}$. We refer the interested reader to Salzmann et al. (2007b) for the full derivation of this result.

Every solution to Eq. (3) re-projects correctly to the image, but the vertex positions are not guaranteed to correspond with the true ones because of the depth ambiguity. However, Eq. (3) is severely under-constrained in practice, with around a third of the singular values of $\mathbf{M}^{(t)}$ being very close to zero (Salzmann et al., 2007b). That means there are several possible solutions, in this case, obtained by moving each point along the line of sight. To factor out these incorrect solutions, we require additional constraints.

Given that $\mathcal{M}_S$ uses triangular faces, we define a surface parametrization consisting of an ensemble of triangles Ω_f for each face $f \in \mathcal{F}$ such that $S \simeq \cup_f \Omega_f$ (Coltraro et al., 2022). For an element $f = (\mathbf{v}_{f,1}, \mathbf{v}_{f,2}, \mathbf{v}_{f,3})$, we define the local parametrization

$$\varphi_f : [-1, 1]^2 \to \Omega_f \quad ; \quad \varphi_f(\xi, \eta) := \sum_{i\in[3]} \mathbf{v}_{f,i} N_i(\xi, \eta), \tag{5}$$

where we define the shape functions N_i as

$$N_1 := \xi \quad ; \quad N_2 := \eta \quad ; \quad N_3 := 1 - \xi - \eta, \tag{6}$$

whose local coordinates of the vertices in the triangular face are $(\xi, \eta) = (1, 0),\ (0, 1),$ and $(0, 0)$, respectively (Zienkiewicz et al., 2013).

Coltraro et al. (2022) used the Riemannian metric preservation assumption to constrain the possible vertices of a mesh for the simulation of deformable surfaces. In particular, they introduced an easy-to-evaluate function C that, given a parametrization $\varphi^{(t)}$ of $\mathcal{M}_S^{(t)}$, satisfies

$$\mathbf{J}_{\varphi^{(t)}}^T \mathbf{J}_{\varphi^{(t)}} = \mathbf{J}_{\varphi^{\text{temp.}}}^T \mathbf{J}_{\varphi^{\text{temp.}}} \iff C(\mathbf{x}^{(t)}) = \mathbf{0}, \tag{7}$$

where $\varphi^{\text{temp.}}$ is the parametrization of the template shape. Given the realistic simulations achieved by Coltraro et al. (2022), we consider the assumption to be reasonable. We refer the interested reader to Coltraro et al. (2022, Algorithm 1) for a detailed computation of the function C, from which we replace the local parametrization of quadrilateral faces with (5).

We can relax Eq. (3) as in Ngo et al. (2016), Salzmann et al. (2007a) and minimize the error in the square sense subject to the metric preservation given by C. That is, for $\Delta\mathbf{x}^{(t)} := \mathbf{x}^{(t)} - \mathbf{x}^{(t-1)}$, the vertices of the mesh can be found by solving

$$\begin{aligned} &\min_{\Delta\mathbf{x}^{(t)}} \left\| \mathbf{M}^{(t)}(\mathbf{x}^{(t-1)} + \Delta\mathbf{x}^{(t)}) \right\|_2^2, \\ &\text{s.t. } \mathbf{C}(\mathbf{x}^{(t-1)} + \Delta\mathbf{x}^{(t)}) = \mathbf{0} \end{aligned} \tag{8}$$

which is a quadratic program with quadratic constraints. To make it computationally tractable, we approximate Eq. (8) with a sequence of quadratic programs with linear constraints using the first-order Taylor expansion $\mathbf{C}(\mathbf{x}^{(t)}) \approx \mathbf{C}(\mathbf{x}^{(t-1)}) + \nabla\mathbf{C}(\mathbf{x}^{(t-1)})\Delta\mathbf{x}^{(t)}$ (Coltraro et al., 2022; Varol et al., 2012).

Overall, for each time step t, the vertex positions of Linear (ours) are found by solving the linear system

$$\begin{cases} \mathbf{M}^{(t)T}\mathbf{M}^{(t)}(\mathbf{x}^{(t-1)} + \Delta\mathbf{x}^{(t)}) + \nabla\mathbf{C}(\mathbf{x}^{(t-1)})^T \lambda = \mathbf{0} \\ \mathbf{C}(\mathbf{x}^{(t-1)}) + \nabla\mathbf{C}(\mathbf{x}^{(t-1)})\Delta\mathbf{x}^{(t)} = \mathbf{0} \end{cases}, \tag{9}$$

where λ is the vector of Lagrange multipliers. We recall that this loss function is not minimized during an offline training process but instead during inference for each of the inputs.

For the initialization of $\mathbf{x}^{(0)}$, we use the template mesh $\mathcal{T}$. Note that the optimization of the first step is expected to take longer and yield a larger $\Delta\mathbf{x}^{(1)}$ than other time steps, where the motion between neighboring images in time is in general small provided the frame rate is high enough.

3.3. Neural approach

Representing the state of a deformable object alone is an open challenge (Yin et al., 2021; Montagnat et al., 2001). Most SfT methods represent objects using meshes, but an alternative is to use the neural parametric representation introduced by Groueix et al. (2018). Such a representation encodes a surface in the weights of a neural network representing a mapping from 2D to 3D. Parametric representations are a general way to express a surface, and their great flexibility allows to control deformations with intuitive parameters (Arriola-Rios et al., 2020). Given a parametric surface, we can analytically compute the first and second fundamental forms, Gaussian curvature, and surface normals (Bednarík et al., 2020).

Implicit neural representations are another method that has recently emerged as a promising alternative to classical discretized representations of signals (Yüce et al., 2022). Both parametric and implicit neural representations can generate continuous surfaces, which amounts to having meshes with an arbitrary resolution. However, an advantage of using parametric representations in front of the popular implicit representations such as NeRFS (Mildenhall et al., 2020) or SDFs (Park et al., 2019) is that the co-domain of a parametric representation is the surface itself. Therefore, if we want to generate a mesh, it is enough to sample the domain at desired locations of the vertices. Instead, implicit representations require using marching cubes on the outputs obtained by sampling many more points on $\mathbb{R}^3$ in the case of SDFs (and the sampled points will only lie on the surface if in a specific level set), and sampling rays for NeRFs (with rays that may or may not hit the object).

Given that both considered datasets only consist of rectangular surfaces, we choose the parametrization $\mathcal{P}$ to be $\mathcal{P} := [0, 1]$ for practical purposes. Other approaches (Bednarík et al., 2020; Groueix et al., 2018) set $\mathcal{P}$ as the interior of the unit square. However, to naturally represent the surface edges, we consider the closure of such a domain. The choice of $\mathcal{P}$ to be the unit square allows representing all developable surfaces, *i.e.*, smooth surfaces that we can flatten into a plane, *e.g.*, cylinders, cones, and toruses.

To model non-developable surfaces like the sphere, we can choose the interior of the unit square. In case of having surfaces with other topologies or with holes, we could obtain $\mathcal{P}$ by conformal flattening of $\mathcal{T}$ (Bartoli et al., 2015), using a texture map (Bartoli et al., 2012), inferring the parametrization domain from data (Lei et al., 2020), or combining different parametric surfaces to create an atlas (Groueix et al., 2018).

A usual assumption in the SfT problem is that the template $\mathcal{T}$ and the tracked surface at time t, $\mathcal{S}^{(t)}$, have the same topology, which implies that they also share a parametrization space (Bartoli et al., 2015). Therefore, we can use the same choice of $\mathcal{P}$ to infer the surface at any point in the sequence.

Proposition 1. *Let $\mathcal{S}$ be a surface that can be parametrized on the unit square. There exists a feed-forward neural network with Softplus non-linearities that can approximate $\mathcal{S}$ with arbitrary precision.*

Proof. The proof follows the same reasoning as in Groueix et al. (2018, Proposition 2.), which states the same for ReLU non-linearities. Instead of relying on the universal representation theorem by Hornik (1991), this proposition requires invoking the theorem by Kidger and Lyons (2020), which works with other activation functions, including Softplus. □

Assuming we can parametrize the surface of interest with the chosen $\mathcal{P}$, we can leverage Proposition 1 and represent it with a feed-forward neural network φ_θ, where θ are the learnable parameters. Note that the parametrization needs to be twice differentiable so that the loss may incorporate first-order derivatives (Bednarík et al., 2020). The requirement motivates the use of Softplus (Dugas et al., 2000), an approximation of the ReLU function with smooth first and second derivatives (Bednarík et al., 2020).

For the sake of notation, let $\varphi_\star$ be the real parametrization of $\mathcal{S}$. Let $P_\mathcal{V}$ be the set of points from $\mathcal{P}$ corresponding to the vertices of the template mesh and the surface meshes for each time. That is

$$P_\mathcal{V} := \{\mathbf{p} \in \mathcal{P} : \mathbf{p} = \varphi_\star^{-1}(\mathbf{v}) \ \forall \mathbf{v} \in \mathcal{V}\}\,. \tag{10}$$

Similarly to the relaxation of Eq. (2) used in Section 3.2, re-projection consistency can be enforced with

$$\mathcal{L}_{\text{projection}} := \frac{1}{|M_\mathcal{P}^{(t)}|} \sum_{(\mathbf{p},\mathbf{i}) \in M_\mathcal{P}^{(t)}} \left\| \Pi(\varphi_{\theta^{(t)}}(\mathbf{p})) - \mathbf{i} \right\|_2 , \tag{11}$$

where

$$M_\mathcal{P}^{(t)} := \{(\mathbf{p}, \mathbf{i}) : \mathbf{p} \in \mathcal{P}, \mathbf{i} = \Pi(\varphi_\star^{(t)}(\mathbf{p})) \in I^{(t)}\}\,. \tag{12}$$

This term enforces that the recovered surfaces are consistent with their corresponding images. This loss is present in all the SfT solutions, either explicitly or implicitly, since the monocular images provide the only information to recover the current state of the surfaces.

Suppose the matches provide image locations for each vertex. One could displace each 3D vertex location along the line of sight, thus obtaining infinitely many surfaces that attain a zero re-projection loss. Enforcing isometry ideally reduces the set of plausible solutions to a single surface (Bartoli et al., 2012, Theorem 1). To favor surfaces whose Riemannian metric is preserved, we add the loss term

$$\mathcal{L}_{\text{metric}} := \frac{1}{|P_\mathcal{V}|} \sum_{\mathbf{p} \in P_\mathcal{V}} \left\| \mathbf{J}^T_{\varphi_{\theta^{(t)}}} \mathbf{J}_{\varphi_{\theta^{(t)}}}(\mathbf{p}) - \mathbf{J}^T_{\varphi_{\theta^{\text{temp.}}}} \mathbf{J}_{\varphi_{\theta^{\text{temp.}}}}(\mathbf{p}) \right\|_F^2 , \tag{13}$$

where $\|\cdot\|_F$ is the Frobenius norm.

Unlike works approximating the surface metric, we can compute it analytically using surface parametrization (Bednarík et al., 2020). Moreover, we enforce metric preservation in $P_\mathcal{V}$, not only in the visible parts, which potentially helps to recover occluded zones (Brunet et al., 2010).

Another difference with previous works is that we incorporate isometry as a soft constraint, which differs from works imposing the metric to be exactly preserved (Bartoli et al., 2012, 2015), which may be a restrictive assumption in real scenarios. Quasi-isometry, on the other hand, is a relatively mild constraint when manipulating surfaces like clothes, as those are nearly inextensible (Salzmann and Fua, 2011). This approximation is especially suited for robotics contexts, where fine details such as wrinkles are irrelevant (Coltraro et al., 2022).

Finally, the loss also includes a temporal regularization term

$$\mathcal{L}_{\text{time}} := \frac{1}{|P_\mathcal{V}|} \sum_{\mathbf{p} \in P_\mathcal{V}} \left\| \varphi_{\theta^{(t)}}(\mathbf{p}) - \varphi_{\theta^{(t-1)}}(\mathbf{p}) \right\|_2 . \tag{14}$$

Adding a small amount of temporal regularization reduces frame-to-frame flickering (Yu et al., 2015) and can help constrain the corners, which can freely bend isometrically if no point correspondences are found on them (Brunet et al., 2010).

The total loss used to update the parameters $\theta^{(t)}$ then becomes

$$\mathcal{L}_{\text{total}} := \mathcal{L}_{\text{projection}} + \lambda_{\text{metric}}\mathcal{L}_{\text{metric}} + \lambda_{\text{time}}\mathcal{L}_{\text{time}}\,. \tag{15}$$

Similarly to Brunet et al. (2010), we obtain the surface parameters by minimizing a combination of a re-projection loss Eq. (11), a term involving the metric tensor favoring plausible poses Eq. (13), and a term that encourages smooth motions Eq. (14). The key differences with this work are:

Algorithm 1 Deformable Surface Reconstruction

1: **Inputs:** Template $\mathcal{T}$, matchings $\{M_{\mathcal{P}}^{(t)}\}_{t\in[T]}$
2: **Output:** Estimated surfaces described by $\{\varphi_{\theta^{(t)}}\}_{t\in[T]}$
3: Compute $P_{\mathcal{V}}$ according to $\mathcal{T}$
4: $\theta^{\text{temp.}} \leftarrow \underset{\theta^{\text{temp.}}}{\text{argmin}} \frac{1}{|P_{\mathcal{V}}|} \sum_{\mathbf{p}_i \in P_{\mathcal{V}}} \left\| \varphi_{\theta^{\text{temp.}}}(\mathbf{p}_i) - \mathbf{v}_i \right\|_2$ ▷Over-fit to template
5: Store $\mathbf{J}_{\varphi_{\theta^{\text{temp.}}}}(\mathbf{p})^T \mathbf{J}_{\varphi_{\theta^{\text{temp.}}}}(\mathbf{p})\ \forall \mathbf{p} \in P_{\mathcal{V}}$ ▷Metric tensors of the template
6: Let $\varphi_{\theta^{(0)}} \leftarrow \varphi_{\theta^{\text{temp.}}}$ ▷Needed for Eq. (14)
7: **for** $t \in [T]$ **do**
8: $\quad \theta^{(t)} \leftarrow \underset{\theta^{(t)}}{\text{argmin}}\, \mathcal{L}_{\text{total}}$ ▷Compute loss Eq. (15) with stored metric tensors
9: **end for**

- **Re-projection loss:** For a matching $(\mathbf{s}, \mathbf{i}) \in M^{(t)}$, Brunet et al. (2010) enforce that $\mathbf{s} \approx \Pi^{-1}(\mathbf{i})$, which requires knowing the depth. Instead, we check that $\Pi(\mathbf{s}) \approx \mathbf{i}$.
- **Metric preservation loss:** Our method enforces that the metric of the surface at any time is close to that of the template. In contrast, Brunet et al. (2010) set the metric tensor to the identity, which only allows modeling unit squares on $\mathbb{R}^3$ when $\mathcal{P} = [0, 1]$.
- **Smoothing loss:** Brunet et al. (2010) favor non-bending surfaces by minimizing the Frobenius norm of the Hessian of the parametrization. Instead, we consider the difference of surfaces in consecutive frames, which aims at reducing frame-to-frame flickering.

In Algorithm 1, we detail the procedure on how to estimate the surfaces for the whole sequence given the inputs of the SfT problem.

4. Experiments

4.1. Datasets

In this paper, we use two public datasets involving rectangular deformable surfaces:

- **Deformable Surface Tracking (DeSurT)** (Wang et al., 2019): Dataset consisting of 11 video streams, with around 300 images each, displaying different materials, deformations, and lighting conditions. The surfaces are either well-textured (Campus, Cobble, Cushion I, Scene, Newspaper I, and Newspaper II), repetitively textured (Brick, Cloth, and Cushion II), or weakly textured (Stone and Sunset). The surfaces are represented with meshes of size 13×10 for all the sequences except the cushion, which uses an 11×11 mesh.
- **Tracking Surface with Occlusion (TSO)** (Ngo et al., 2015): Set of two image sequences with about 200 images each and meshes of size 13×10. The sequences consist of a poorly textured surface with real occlusions (White paper) and a well-textured surface with artificial occlusions (Classroom).
- **Texture-less Deformable Surfaces (TDS)** (Bednarík et al., 2018): Dataset showing deformable surfaces under various lighting conditions. We use seven sequences (the ones with ground-truth vertex annotations) with around 900 images each, displaying a piece of cloth represented with a 31×31 mesh.

Given that the surfaces of both datasets are rectangular and meshes have equispaced vertices, we define $P_{\mathcal{V}}$ with the coordinates given by a grid on the unit square.

4.2. Baselines

We use the following methods to compare the performance of the proposed technique. Unless specified, we use the publicly available code implemented by the original authors without modifying the algorithms or the hyper-parameters.

- **GP:** Unconstrained GP-LVM (Varol et al., 2012). We exclude the constrained GP-LVM introduced in the same paper from the comparisons as it consistently under-performed the unconstrained version.
- **Lap:** Method based on Laplacian meshes (Magnenat et al., 2015; Ngo et al., 2016; Östlund et al., 2012).
- **Dense:** Dense image registration in Ngo et al. (2015).
- **Graph:** Deformable surface tracking using graph matching (Wang et al., 2019).
- **TexLess:** The model from Bednarík et al. (2018) trained from scratch with ground-truth mesh vertices for each dataset and mesh resolution.
- **Linear (ours):** Constrained linear optimization method outlined in Section 3.2.
- **Neural (ours):** Proposed method described in Section 3.3.

4.3. Implementation details

The sparse linear system in Eq. (9) required by Linear (ours) is solved in the least-squares sense using the SciPy (Virtanen et al., 2020) routine limited to 100 iterations. This procedure is repeatedly applied until the approximation is good enough, as understood by the same criterion of Coltraro et al. (2022).

For the proposed method, we set an early stopping mechanism for the optimizations in Algorithm 1 with different criteria for the overfitting of the template (line 4) and the surface reconstruction (line 8). When learning the template, we set a patience of 200 iterations, meaning that the minimization process halts if the loss does not improve during 200 optimization steps. Since surface reconstruction starts with an initialization close to the solution for high enough frame rates, we set a patience of 20 and restrict the total number of iterations to 200. In both cases, the model that we use is the one that attains the minimum loss in the whole optimization process.

The matches $M^{(t)}$ can be obtained with SIFT (Lowe, 1999), SURF (Bay et al., 2006), Ferns (Ozuysal et al., 2010), or soft graph matching (Wang et al., 2019) followed by an outlier rejection mechanism (Vogler et al., 2007). Obtaining the matches with a dedicated model and then performing reconstruction results in correspondences robust to occlusions, especially for well-textured surfaces (Ngo et al., 2015).

On the one hand, for the Neural (ours), we use matches obtained with CoTracker (Karaev et al., 2023), a model based on transformers (Vaswani et al., 2017) that exploits the correlation across features by tracking dense points in a frame jointly across a video sequence. In this case, the matches were obtained by uniformly sampling points on the faces and borders of the template and tracking them throughout the sequence. Using CoTracker allows tracking texture-less surfaces and obtaining denser matches.

On the other hand, we found that the Linear (ours) method works best when using the correspondences provided by the registration algorithm used in the baseline Lap, based on matching SIFT (Lowe, 1999) features and then applying an outlier rejection mechanism. The Lap algorithm did not terminate and hence could not provide matchings

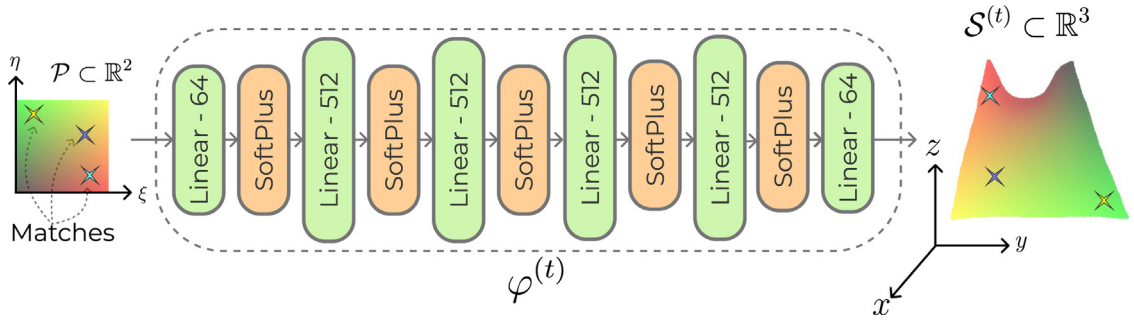


Fig. 3. Model architecture. Given 2D-to-2D matches relating the tracked surface and the input image at time t, we map them to 3D coordinates using $\varphi^{(t)}$. We parametrize the mapping $\varphi^{(t)}$ using a neural network with linear layers and SoftPlus activations. The SoftPlus activations, unlike more popular choices such as ReLUs, make the mapping twice differentiable, a requirement to compute differential geometric quantities. The predicted 3D points are projected to image coordinates and compared to the 2D locations provided by the point tracker. Then, we sample points in a uniform grid on $\mathcal{P}$, transform them using $\varphi^{(t)}$, and enforce metric preservation and temporal consistency on the output, which lies in 3D.

Table 2
Ablation. Design choices of our model and their impact on the performance on the Lr_bottom_edge sequence of the TDS dataset in terms of the mean tracking error (mm). We show modifications belonging to different parts of our pipeline: the input, the loss, and the model. The last row of the table shows the result of our final model, for which results are reported. The last column of this table shows the difference in performance between an experiment and Neural (ours).

Modification	Error (↓)	Difference
Without coordinate normalization	77.11	+8.77
$\lambda_{\text{metric}} = 0$	81.30	+12.96
$\lambda_{\text{time}} = 0$	72.23	+3.89
$\lambda_{\text{metric}} = \lambda_{\text{time}} = 0$	130.58	+68.24
Shallower: 4 layers	66.33	-2.01
Deeper: 10 layers	75.72	+7.38
Narrower: 64 units in hidden layers	67.94	-0.40
Wider: 1024 units in hidden layers	78.31	+9.97
Neural (ours)	68.34	–

for all the frames indicated with (∗) in Table 3. In this case, we used synthetic matches by sampling a random point inside each facet of the mesh. Note that these matches are noise free and cannot be obtained in a real pipeline.

4.4. Ablation

The proposed neural approach offers flexibility regarding the class of models used to represent φ_θ, the loss terms' weighting, and the optimization algorithm updating the model parameters in new frames. In practice, we use a simple multi-layer perceptron to parametrize the surface similarly to NeRFs (Mildenhall et al., 2020).

We use Optuna (Akiba et al., 2019) to set the hyperparameters of the model, namely the number of layers (we test from 3 to 10) and the number of units of each layer (64, 128, 256, or 512 for the first, middle and last layers). We select an architecture based on the average performance across all sequences and show the resulting model architecture in Fig. 3.

We also use the hyperparameter search to determine the learning rate (which we explore in the log-scaled range from 10^{-5} to 10^{-1}) and the amount of time and metric regularization, *i.e.*, λ_{time} and λ_{metric}, respectively (sampling the interval from 0 to 100).

In Table 2, we present an ablation study comparing different design choices. Similarly to Groueix et al. (2018), we processed all the meshes so that the vertices fall in the centered unit ℓ_2 ball. We can see that this normalization results in a substantial increase in performance, as the domain of values allows for better-conditioned gradients and results in overall improved convergence.

Incorporating the metric preservation term into the loss function is the main contribution of this paper and is shown to be the principal driver of performance in Table 2. When not considering the temporal loss term, the performance drop is smaller and attributed to a slight jitter. Removing both temporal and metric losses, the model optimizes only for the projection error, which results in meshes that correctly project to the tracked points but are not metrically consistent nor necessarily coherent with neighboring time steps.

Finally, we test modifications on the model on the same sequence. Making the model larger, both wider and deeper, for this case results in worse performance However, we can see that for this particular sequence, using a shallower model can reduce the error by around 2 mm on average, and using a narrower model can slightly improve the performance.

Overall, the ablation shows that there is room for tiny incremental improvements by tuning the model hyperparameters to specific sequences. Nonetheless, we use the same architecture for all sequences to avoid performing architecture searches for each sequence.

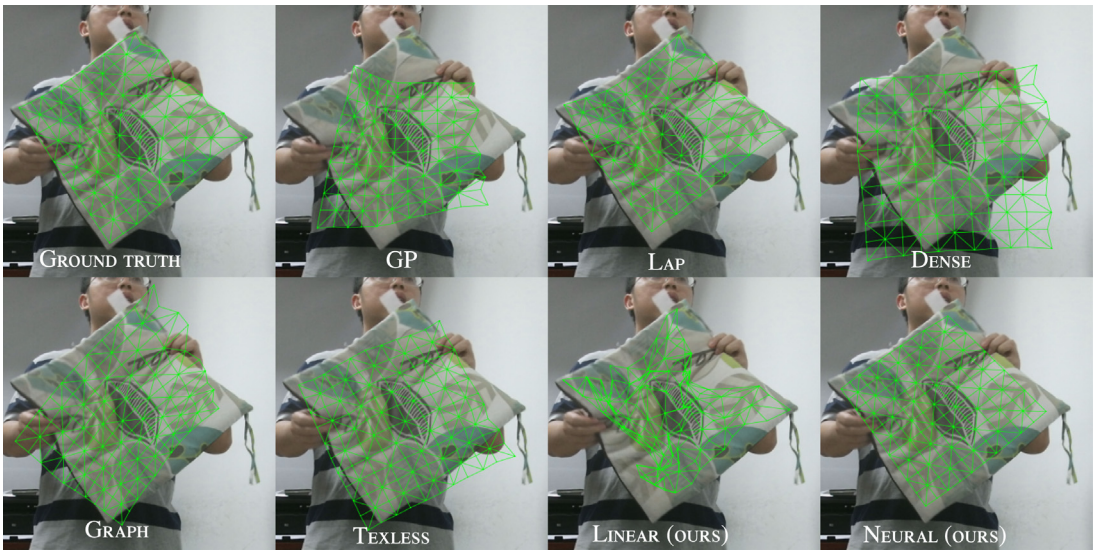
Besides the changes that contributed to the performance of our model, we also tested the following changes that obtained a worse mean tracking error:

- **Batch normalization** (Ioffe and Szegedy, 2015): We experimented with applying batch normalization after every linear layer except the last one. In this case, we removed the biases of all the linear layers and added a bias term to the output layer. However, we found the same unstable training reported in Bednarík et al. (2020).
- **Positional encoding:** Similarly to NeRFs, the inputs to our model are low-dimensional coordinates. Mildenhall et al. (2020) used sines and cosines at different frequencies determined by multiples of the input coordinate to create a higher dimensional representation and allow representing higher frequency functions. In this experiment, we mapped the 2D coordinates to $[-1, 1]^2$ and used the same positional encodings as (Mildenhall et al., 2020). While this change improved the performance on some occasions, it was detrimental in most sequences.
- **Normalized projection:** Given that predicting normalized surface coordinates was beneficial, we tested with predicting image coordinates normalized to $[-1, 1]$. Similarly to the change above, we found slight improvements in some sequences, but overall, it was not beneficial.
- **Skip connections:** We also tried concatenating the inputs to the activations of one of the middle layers to prevent vanishing gradients. This strategy is also used by Mildenhall et al. (2020) but did not provide substantial benefits for our case.

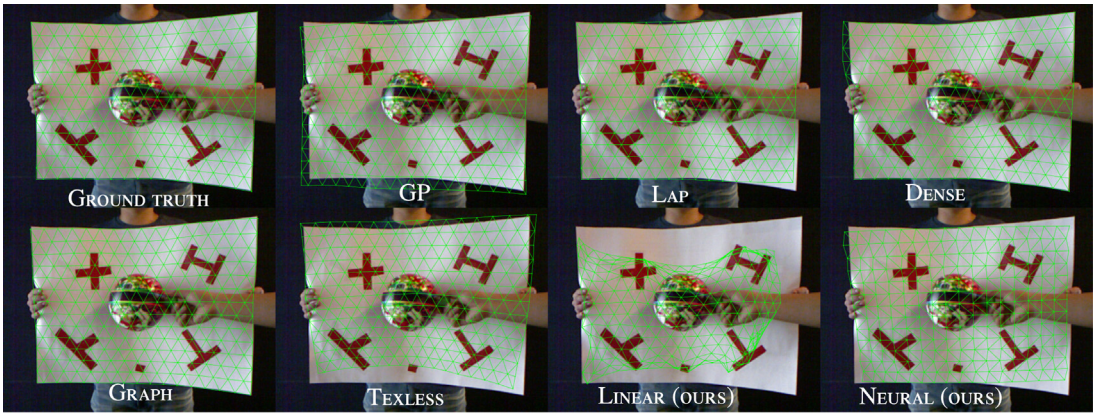
4.5. Evaluation

4.5.1. Ground truth mesh

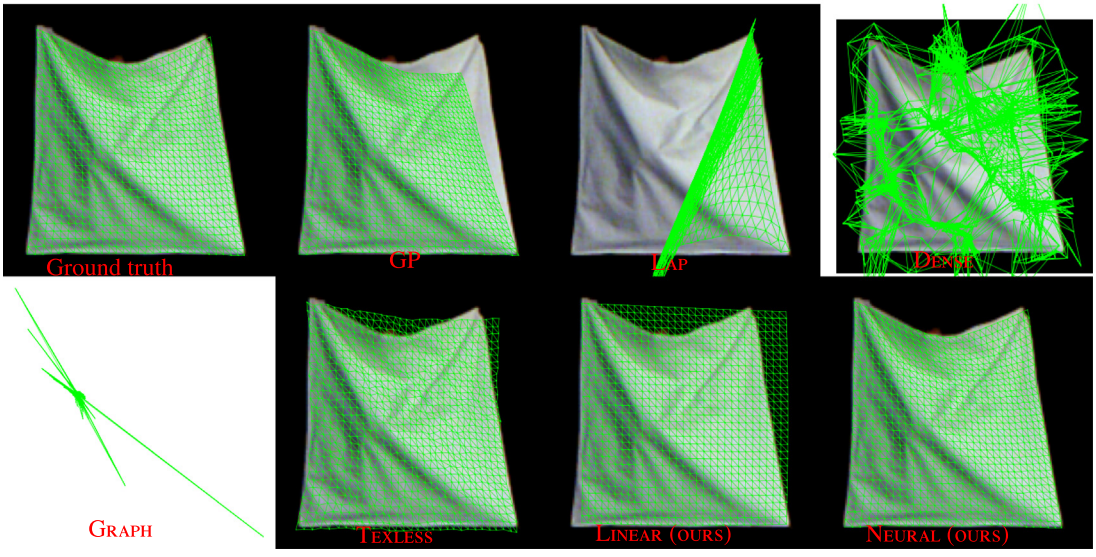
To evaluate the accuracy of the proposed model, we compute the Euclidean distances from vertex to vertex. Concretely, Table 3 reports



(a) **Well-textured reconstruction:** Frame 103 of the Cushion I sequence of the DeSurT dataset.



(b) **Occluded reconstruction:** Frame 145 of the White paper sequence of the TSO dataset.



(c) **Textureless reconstruction:** Frame 75 of the Lr_bottom_edge_tl_corn sequence of the TDS dataset. The white borders of Dense and Graph indicate mesh vertex predictions that project outside of the image.

Fig. 4. Qualitative results. Neural (ours) successfully recovers the tracked surface in all cases, with the most noticeable gap in performance with other methods for the TDS dataset (except our linear approach, which can also successfully estimate the tracked surface). Linear (ours) is sensible to occlusions and diverges in some cases for the other datasets. In this case, Neural (ours) showcases its superiority and robustness to different surfaces.

Table 3
Mean tracking error. This table shows the mean tracking error (mm) obtained when reconstructing a mesh from monocular images. The **best value** for each sequence (the lowest) is shown in boldface, and the runner-up is underlined. (∗) indicate that a method did not terminate, in which case the average error of the meshes obtained before the method crashed is reported. The sequences used to train and validate Texless are indicated with (†) and (‡). Note that we favor Texless by providing it with full sequences and, for the sequences with (†), reporting the values on training examples.

Dataset ↓/Methods→		GP	Lap	Dense	Graph	Texless	Linear (ours)	Neural (ours)
DeSurT	Brick	109.45	44.80	81.56	**25.62**	82.23†	67.20	39.47
	Campus	85.48	85.80	57.00	**22.36**	71.56	66.29	35.62
	Cloth	104.30	460.44	85.58	**46.96**	114.63	899.65	55.30
	Cobble	100.54	41.90	68.90	**21.70**	80.39	56.33	40.92
	Cushion I	104.18	72.55	108.65	44.55	89.75†	250.69	**30.76**
	Cushion II	104.05	157.76	96.91	**38.05**	218.35‡	918.63	46.52
	Newspaper I	83.25	63.23	85.12	**22.20**	72.67†	79.83	34.93
	Newspaper II	79.85	67.66	73.65	**25.50**	186.36	68.46	40.78
	Scene	102.25	57.36	82.37	**21.14**	235.45	69.85	44.12
	Stone	100.62	421.36	90.93	**36.11**	85.55†	1140.23	38.53
	Sunset	107.89	248.29∗	67.24	**29.22**	87.35‡	72.78	40.81
TSO	Classroom paper	49.16	8.22	35.66	**3.13**	47.54†‡	32.92	19.28
	White paper	45.50	54.48	26.20	**9.92**	43.64	737.09	33.35
TDS	Lr_bottom_edge	94.72	882.28∗	85.50	1184.78	86.35†	**66.43**	68.34
	Lr_bottom_edge_tl_corn	55.59	2999.26∗	71.00	1043.35	55.01†	**40.93**	47.64
	Lr_left_edge	78.89	1603.06∗	103.06	1154.88	82.70†	58.58	**52.68**
	Lr_tl_tr_corns	66.19	850.86∗	87.80	1049.56	62.68†	59.09	**58.69**
	Lr_top_edge_1	93.43	509.31∗	93.53	1059.72	81.17†	69.08	**60.16**
	Lr_top_edge_2	74.79	600.83∗	88.89	1108.46	71.83†	63.32	**55.83**
	Lr_top_edge_3	59.50	361.39∗	70.54	1032.46	69.82‡	54.65	**52.04**

the mean of such distances, *i.e.*,

$$\frac{1}{T}\sum_{t\in[T]}\left[\frac{1}{N}\sum_{n\in[N]}\left\|\hat{\mathbf{v}}_n^{(t)}-\mathbf{v}_n^{(t)}\right\|_2\right], \tag{16}$$

the quantity reported in several 3D reconstruction works (Ngo et al., 2015; Pumarola et al., 2018; Varol et al., 2012). Given a parametrization $\varphi_{\theta^{(t)}}$ obtained with the proposed method, one can compute $\hat{\mathbf{v}}_n^{(t)}$ by evaluating the parametrization at the point in $P_{\mathcal{V}}$ corresponding to the $n-$th vertex.

We can see that the Graph method obtains a superior performance on the DeSurT and TDS datasets, with our neural approach, *i.e.*, Neural (ours), being the runner-up in most cases and obtaining the best mean tracking error for the Cushion I sequence. For the TSO sequence, Lap achieves the second-best performance, which is not surprising given the fact that Graph uses the same reconstruction algorithm and only changes the matching algorithm, leading to a close performance in the well-textured classroom surface.

When reconstructing textureless surfaces, we can see that both our approaches consistently outperform all the baselines. The neural approach outlined in Section 3.3 obtains the best results for most sequences, with the approach based on solving a linear program with linear constraints described in Section 3.2 closely following.

It is worth noting that the best performance for all sequences is attained by methods relying on an external method to obtain correspondences. The superiority of algorithms taking matches as input contrasts with the current trend of predicting surfaces directly from images. However, relying on an external registration algorithm is a double-edged sword and becomes the main limitation of the proposed method. The reason is that matching algorithms introduce noise and may fail with repetitive or poorly textured surfaces, especially the classical matching methods. We can see this phenomenon on the TDS dataset, where the Graph method fails at recovering the surfaces.

An alternative is to use dense approaches like Dense, which does not extract features but instead maximizes a similarity measure to perform registration (Ngo et al., 2015; Yu et al., 2015). These approaches usually need consistent illumination and suffer from brightness changes, occlusions, and motion blur (Wang et al., 2019).

Data-based approaches like Texless do not require matches but typically work only with the surface seen during training and require fine-tuning to different templates. Fuentes-Jimenez et al. (2021) attempted training texture-generic neural networks, but their results were worse than the ones obtained with texture-specific methods. Table 3 showcases that data-based approaches did not perform strictly better than other methods for any tested sequences.

Linear (ours) directly incorporates the Riemannian metric preservation constraint proposed by Coltraro et al. (2022) for cloth simulation and achieves one of the best performances for the TDS sequences. Despite the notable results backing up the metric preservation assumption, this approach scales poorly with the number of vertices (being the slowest method in Table 5) and diverges in some cases (*e.g.*, the Stone sequence in Table 3). Neural (ours) obtains very similar results in TDS and consistently outperforms Linear (ours) on DeSurT.

In Fig. 4, we provide a qualitative evaluation of the obtained results. We depict the projection of the reconstructed mesh on top of the input image for each tested method and the ground truth. Recall that, as mentioned above, a perfect vertex projection does not guarantee a correct reconstruction due to the depth ambiguity. Therefore, one must consider both qualitative and quantitative results, the former considering how well the projected mesh matches the image and the latter considering the estimation error in 3D.

4.5.2. Mesh-to-pointcloud

For all the datasets used in this paper, ground truth mesh vertices are provided. In all cases, the process to obtain the ground truth is automated and based on capturing RGBD data and processing the resulting pointcloud. However, this process is not trivial as correspondences are unknown.

For the TDS dataset, the authors learn a neural network that maps RGB images to a mesh using synthetic data. Then, the same model is trained on the real dataset of pointclouds incorporating losses that take into account the segmentation mask and the depth map of the cloth. Nevertheless, the process to obtain the ground truth of the TSO and DeSurT datasets is not specified in the papers that presented them. Particularly for the DeSurT dataset, we noticed that the ground truth was inaccurate in some cases, for example those illustrated in Fig. 5.

For this reason, we report a distance using the original unprocessed point cloud $\mathcal{Z}^{(t)}$ for each time step t. Given that we need a metric between sets of points with different sizes and without a matching between them, we use the mean of the directed Hausdorff distances, defined as:

$$\frac{1}{T}\sum_{t\in[T]}\max_{\hat{\mathbf{v}}\in\mathcal{V}^{(t)}}\left\{\min_{\mathbf{z}\in\mathcal{Z}^{(t)}}\|\hat{\mathbf{v}}-\mathbf{z}\|\right\}, \tag{17}$$

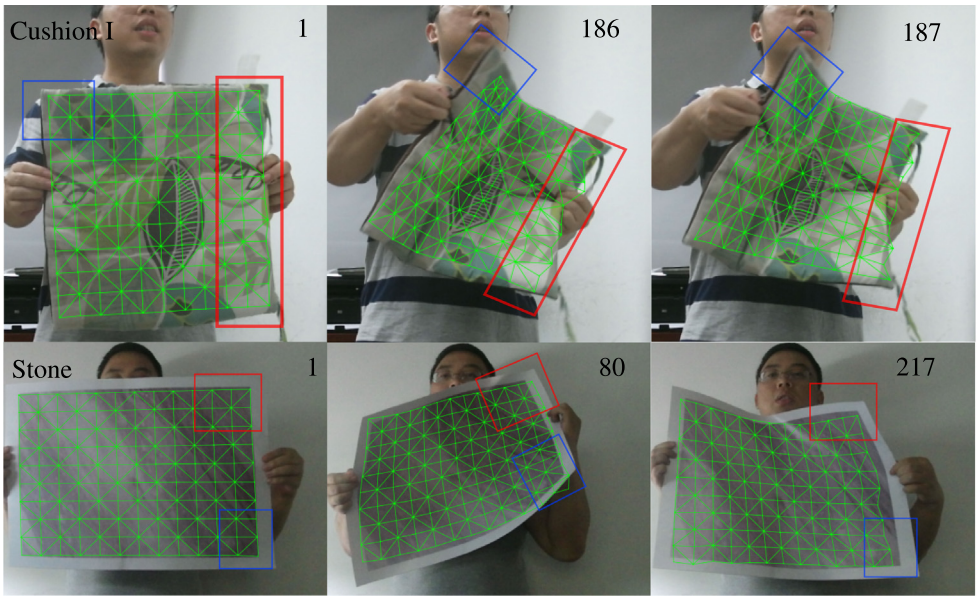


Fig. 5. Inaccuracies in ground truth meshes. Obtaining the vertex locations from RGBD data is a nontrivial procedure that can lead to errors. Such errors can stem from different sources, wrong correspondences from the feature-matching algorithm being the most probable. In this case, methods using the image correspondences obtained with the same procedure used to generate the ground truth will make the same mistakes, resulting in positively biased performance metrics that constitute an unfair and unrealistic evaluation. This figure shows three frames from the Cushion I and Stone sequences from the DeSurT dataset. The first frame, corresponding to the template mesh, shows a near-planar rectangular mesh that covers all the objects. In later frames, the vertices do not correspond to the original surface points of the template, with the zones indicated with blue and red rectangles being the ones where the mismatch is most noticeable.

Table 4
Hausdorff distance. Given that we found some inconsistencies in the ground truth annotations of the meshes, we report the error between the predicted mesh and the input point cloud. Concretely, we report the directed Hausdorff distance (mm) to account for sets of points of different sizes and without correspondences. The **best value** for each sequence (the lowest) is shown in boldface, and the runner-up is underlined. (∗) indicate that a method did not terminate, in which case the average error of the meshes obtained before the method crashed is reported. The sequences used to train and validate Texless are indicated with (†) and (‡). Note that we favor Texless by providing it with full sequences and, for the sequences with (†), reporting the values on training examples. (⊗) indicates that the depth used to obtain the point cloud is given by a monocular depth estimator and the results are thus sensible to prediction errors.

Dataset ↓/Methods→		GP	Lap	Dense	Graph	Texless	Linear (ours)	Neural (ours)
DeSurT⊗	Brick⊗	147.38	83.66	100.65	82.23	101.48†	112.20	**76.35**
	Campus⊗	119.87	141.91	109.03	86.28	98.99	137.13	**83.14**
	Cloth⊗	149.31	556.14	147.36	105.62	129.58	754.33	**98.59**
	Cobble⊗	144.07	93.27	97.69	86.87	98.63	91.72	**81.12**
	Cushion I⊗	128.83	116.16	132.83	85.46	101.45†	272.47	**56.67**
	Cushion II⊗	283.55	215.49	107.43	81.37	98.88‡	836.13	**73.61**
	Newspaper I⊗	133.74	109.52	146.88	**79.30**	116.97†	150.47	97.29
	Newspaper II⊗	133.96	103.61	137.86	**75.24**	217.98	116.18	93.54
	Scene⊗	127.86	97.25	111.68	**76.47**	236.48	123.97	83.34
	Stone⊗	140.90	504.37	136.45	126.56	116.54†	1006.38	**81.65**
	Sunset⊗	162.96	348.11	123.49	97.37	140.38†	149.28	**92.04**
TSO⊗	Classroom paper⊗	80.84	47.72	59.99	44.25	76.50†‡	57.32	**43.07**
	White paper⊗	113.13	112.11	**72.56**	81.45	107.38	628.08	73.58
TDS	Lr_bottom_edge	193.65	1004.33	216.28	1370.55	178.31†	163.59	**135.45**
	Lr_bottom_edge_tl_corn	159.99	3040.47	180.72	1208.29	157.34†	**122.07**	132.98
	Lr_left_edge	186.85	1640.69	235.25	1333.07	189.88†	148.27	**141.27**
	Lr_tl_tr_corns	157.34	884.21	177.79	1219.90	147.03†	145.09	**130.31**
	Lr_top_edge_1	212.57	556.37	216.85	1263.86	184.79†	164.27	**142.74**
	Lr_top_edge_2	183.25	690.91	196.30	1218.45	169.71†	156.97	**138.67**
	Lr_top_edge_3	144.60	490.05	175.51	1042.67	157.63‡	151.59	**126.59**

where $\mathcal{V}^{(t)}$ is the set of estimated vertices of the surface at time t. Note that, since this metric is computed with a dense point cloud $\mathcal{Z}$, it approximates the maximum distance between any point and the true surface. On the one hand, this is complementary to the mean tracking error as it reports an extreme metric instead of an average. On the other hand, reporting a metric on the raw input point cloud circumvents possible errors in the ground truth mesh.

The TDS dataset provides point clouds, but other datasets do not. For the DeSurT dataset we compute segmentation masks using the convex hull of the mesh projected to the image. To account for incorrect mesh vertices, we dilate the resulting mask using image morphology, as most of the error is due to the borders of the mesh being towards the inside of the mask. Finally, we use the SAM segmentation model (Kirillov et al., 2023) to refine the segmentations. For the TSO dataset, segmentation masks are provided for the object and the occlusions. We then obtain a final mask by removing occluded zones.

We obtain depth estimates for DeSurT and TDS using an off-the-shelf monocular depth estimation model. Estimators providing depth maps from monocular RGB images, obtain results that do not necessarily have the same scale as the true unknown values. We tackle this problem by first creating a point cloud from the RGB image, the estimated depth, and the segmentation mask, and then finding the scaling of the

Table 5
Additional quantitative performance indicators. The row Time shows the average time in seconds required to process one image for the TDS sequence Lr_bottom_edge, the dataset with higher resolution meshes (961 vertices). The rows DeSurT, TSO and TDS present the mean and standard deviation of the tracking error in millimeters across the sequences of each dataset. The **best value** for each sequence (the lowest) is shown in boldface, and the runner-up is underlined. (⊗) indicates that the depth used to obtain the point cloud is given by a monocular depth estimator and the results are thus sensible to prediction errors.

Dataset ↓/Methods→	GP	Lap	Dense	Graph	Texless	Linear (ours)	Neural (ours)
Mean tracking error statistics (Table 3)							
DeSurT	98.35 ± 9.90	156.47 ± 146.19	81.63 ± **13.88**	**30.30** ± 9.09	122.03 ± 62.24	335.45 ± 405.84	40.71 ± **6.20**
TSO	47.33 ± **1.83**	31.35 ± 23.13	30.93 ± 4.73	**6.53** ± 3.39	45.59 ± 1.95	385 ± 352.08	26.31 ± 7.03
TDS	74.73 ± 14.33	1115.28 ± 854.54	85.76 ± 10.84	1090.46 ± 55.51	72.79 ± 10.56	58.87 ± 8.63	**56.48 ± 6.23**
Hausdorff distance statistics (Table 4)							
DeSurT⊗	152.04 ± 43.12	215.41 ± 165.81	122.85 ± 17.50	89.34 ± 14.54	132.49 ± 46.72	340.93 ± 328.94	**83.39 ± 11.56**
TSO⊗	96.99 ± 16.15	79.91 ± 32.19	66.27 ± **6.28**	62.85 ± 18.60	91.94 ± 15.44	342.70 ± 285.38	**58.33** ± 15.26
TDS	176.89 ± 22.05	1186.72 ± 836.30	199.81 ± 21.61	1236.68 ± 97.84	169.24 ± 14.74	150.27 ± 13.34	**135.43 ± 5.46**
Time (s)	**0.01**	5.80	26.73	32.52	**0.01**	49.00	0.73

resulting point cloud that better aligns with the ground truth mesh vertices. Doing so addresses the scale ambiguity, but predicted depth values may still contain some errors. In practice, we use ZoeDepth (Bhat et al., 2023) for its zero-shot transfer capabilities to a wide variety of domains.

In Table 4, we report the Hausdorff distance between the estimated point clouds and the meshes provided by each compared method. As hypothesized, the TDS dataset, for which ground truth depth maps are provided and whose ground truth mesh acquisition is reported, presents a very similar performance to that of Table 3. Concretely, the best and runner-up performances are consistent with the mean tracking error, with only one of the sequences showing a flip in the performance ranking order.

However, we can observe that for the DeSurT and TSO sequences, Neural (ours) obtains a much better performance relative to that of the other baselines. Concretely, our method achieves the best performance in 8/11 sequences of the DeSurT dataset in terms of the Hausdorff distance while only achieving the lowest mean tracking error using the ground truth mesh in 1/11 sequences. As expected, the 2/11 sequences of the DeSurT dataset in which Graph obtains the lowest Hausdorff distance show well-textured surfaces. In this case, obtaining correspondences using classical matching algorithms is easier, leading to better surface estimations for Graph and facilitating the ground truth mesh extraction. Note that we obtain the results for DeSurT and TSO with an estimated depth map. Hence, the Hausdorff distance evaluation is affected by possible estimation errors. While TDS offers sensed depth, we use a pseudo-ground truth for the other datasets, circumventing the absence of ground truth depth values and the inaccuracies of the ground truth meshes.

These results point out a possible unfair comparison but should be contrasted as the Hausdorff distance does not consider correspondences. Instead, all points are treated equally, and what matters is that, for each point of the predicted mesh, there is a point in the pointcloud that is close enough. That said, a perfect surface reconstruction would attain zero error in both cases if the ground truth mesh is ideal so the two results are complementary and indicative of the reconstruction performance.

Considering both the mean tracking error and the Hausdorff distance, Neural (ours) is crowned as the best performing method, as shown in Table 5. While Graph obtains the best average performance per dataset in terms of the mean tracking error, our two approaches outperform this method on the TDS dataset. Moreover, in terms of the Hausdorff distance, our neural approach is indisputably the best method. Interestingly, our approach shows a stable performance in the different sequences of each datasets, hence attaining the lowest standard deviation in most cases.

In Table 5 we also report the average time[1] required by each method to process one image and statistics about the performance across all the sequences.

On the one hand, as expected, learning-based methods (GP and Texless) attain the lowest inference time as they only need to evaluate a function. Optimization-based methods require finding the best surface parameters given an image, which requires performing several function evaluations and parameter updates. The proposed method is optimization-based but has significantly lower computational overhead than its competitors. Concretely, the time to compute a forward pass is roughly 1 ms, while the backward pass takes another 7 ms. This makes it feasible to compute several gradient updates and still keep reduced computation times.

On the other hand, the proposed approach attains the best average performance across both tested datasets. Moreover, the standard deviation of mean tracking errors for different sequences is among the lowest, which shows that our method is generally applicable and robust.

5. Conclusions

This work tackles the inherently ill-posed problem of reconstructing deformable surfaces from monocular images by assuming that the Riemannian metric of the manipulated surface is approximately constant. Isometry is a mild hypothesis for a broad range of scenarios since many materials do not perceptibly shrink or stretch when they suffer deformations (Salzmann and Fua, 2009).

The metric preservation constraint can be easily incorporated when using parametric surfaces as they allow for an analytic computation of differential geometric quantities. With this in mind, one could adapt our framework for constraining other properties considered in SfT, *e.g.* the surface area (Salzmann et al., 2007a). Another advantage of learning a parametric surface is that we can approximate a surface with arbitrary precision (Proposition 1) and obtain a continuous representation.

Contrasting to the previous methods for SfT using neural networks, our approach does not require offline training but instead uses an iterative optimization process for each input. Although this solution is slower than data-driven regression methods, such optimization takes orders of magnitude less time than the other optimization-based methods. Moreover, it lifts the requirement of data-driven methods to have a dataset with enough samples to represent the dynamics and appearance of an object. Hence, we can use our approach on new sequences without modification. Additionally, being an optimization-based solution, it incorporates the error-feedback loop that regression methods lack.

[1] Computations performed in a machine with an 11th Gen Intel (R) Core (TM) i9-11900KF @ 3.50 GHz with 16 cores and a NVIDIA GeForce RTX 3090 GPU with 24 GB of VRAM.

CRediT authorship contribution statement

Oriol Barbany: Writing – review & editing, Writing – original draft, Visualization, Validation, Software, Resources, Methodology, Investigation, Formal analysis, Data curation, Conceptualization. **Adrià Colomé:** Writing – review & editing, Supervision. **Carme Torras:** Writing – review & editing, Supervision, Funding acquisition.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Data is taken from previous research and is publicly available. Please refer to the cited papers for access to the datasets.

Acknowledgments

This work is part of the project CLOTHILDE (“CLOTH manIpulation Learning from DEmonstrations”) which has received funding from the European Research Council (ERC) under the European Union’s Horizon 2020 research and innovation program (Advanced Grant agreement No. 741930). O.B. thanks Franco Coltraro and Xavier Gràcia for their insights on differential geometry, and the European Laboratory for Learning and Intelligent Systems (ELLIS) PhD program for support.

Appendix A. Supplementary data

Supplementary material related to this article can be found online at https://doi.org/10.1016/j.cviu.2024.104155.

References

Akiba, T., Sano, S., Yanase, T., Ohta, T., Koyama, M., 2019. Optuna: A next-generation hyperparameter optimization framework. In: The 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining. pp. 2623–2631.

Alet, F., Doblar, D., Zhou, A., Tenenbaum, J.B., Kawaguchi, K., Finn, C., 2021a. Noether networks: Meta-learning useful conserved quantities. In: NeurIPS. pp. 16384–16397.

Alet, F., Kawaguchi, K., Lozano-Pérez, T., Kaelbling, L.P., 2021b. Tailoring: Encoding inductive biases by optimizing unsupervised objectives at prediction time. In: NeurIPS. pp. 29206–29217.

Arriola-Rios, V.E., Guler, P., Ficuciello, F., Kragic, D., Siciliano, B., Wyatt, J.L., 2020. Modeling of deformable objects for robotic manipulation: A tutorial and review. Front. Robotics AI 7 (82), 1–25.

Bartoli, A., Gérard, Y., Chadebecq, F., Collins, T., 2012. On template-based reconstruction from a single view: Analytical solutions and proofs of well-posedness for developable, isometric and conformal surfaces. In: CVPR. pp. 2026–2033.

Bartoli, A., Gérard, Y., Chadebecq, F., Collins, T., Pizarro, D., 2015. Shape-from-template. IEEE Trans. Pattern Anal. Mach. Intell. 37 (10), 2099–2118.

Bay, H., Tuytelaars, T., Van Gool, L., 2006. SURF: Speeded up robust features. In: ECCV. pp. 404–417.

Bednarík, J., Fua, P.V., Salzmann, M., 2018. Learning to reconstruct texture-less deformable surfaces from a single view. In: 3DV. pp. 606–615.

Bednarík, J., Parashar, S., Gundogdu, E., Salzmann, M., Fua, P., 2020. Shape reconstruction by learning differentiable surface representations. In: CVPR. pp. 4716–4725.

Bhat, S.F., Birkl, R., Wofk, D., Wonka, P., Müller, M., 2023. ZoeDepth: Zero-shot transfer by combining relative and metric depth. arXiv:2302.12288.

Bodenhagen, L., Fugl, A.R., Jordt, A., Willatzen, M., Andersen, K.A., Olsen, M.M., Koch, R., Petersen, H.G., Krüger, N., 2014. An adaptable robot vision system performing manipulation actions with flexible objects. IEEE Trans. Autom. Sci. Eng. 11 (3), 749–765.

Brunet, F., Hartley, R., Bartoli, A., Navab, N., Malgouyres, R., 2010. Monocular template-based reconstruction of smooth and inextensible surfaces. In: ACCV. pp. 52–66.

Casillas-Pérez, D., Pizarro, D., Fuentes-Jimenez, D., Mazo, M., Bartoli, A., 2021. The isowarp: The template-based visual geometry of isometric surfaces. IJCV 129 (7), 2194–2222.

Chhatkuli, A., Pizarro, D., Bartoli, A., Collins, T., 2017. A stable analytical framework for isometric shape-from-template by surface integration. IEEE Trans. Pattern Anal. Mach. Intell. 39 (5), 833–850.

Coltraro, F., Amorós, J., Alberich-Carramiñana, M., Torras, C., 2022. An inextensible model for the robotic manipulation of textiles. Appl. Math. Model. 101.

Do Carmo, M.P., 2016. Differential Geometry of Curves and Surfaces: Revised and Updated Second Edition. Courier Dover Publications.

Dugas, C., Bengio, Y., Bélisle, F., Nadeau, C., Garcia, R., 2000. Incorporating second-order functional knowledge for better option pricing. In: NeurIPS. pp. 472–478.

Fuentes-Jimenez, D., Pizarro, D., Casillas-Pérez, D., Collins, T., Bartoli, A., 2021. Texture-generic deep shape-from-template. IEEE Access 9, 75211–75230.

Fuentes-Jimenez, D., Pizarro, D., Casillas-Pérez, D., Collins, T., Bartoli, A., 2022. Deep Shape-from-Template: Single-image quasi-isometric deformable registration and reconstruction. Image Vis. Comput. 127 (104531), 1–19.

Groueix, T., Fisher, M., Kim, V.G., Russell, B., Aubry, M., 2018. AtlasNet: A Papier-Mâché approach to learning 3D surface generation. In: CVPR. pp. 216–224.

Hornik, K., 1991. Approximation capabilities of multilayer feedforward networks. Neural Netw. 4 (2), 251–257.

Ioffe, S., Szegedy, C., 2015. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. In: Proceedings of the 32nd International Conference on Machine Learning, Vol. 37. pp. 448–456.

Kairanda, N., Tretschk, E., Elgharib, M., Theobalt, C., Golyanik, V., 2022. φ-SfT: Shape-from-Template with a Physics-Based Deformation Model. In: CVPR. pp. 3938–3948.

Karaev, N., Rocco, I., Graham, B., Neverova, N., Vedaldi, A., Rupprecht, C., 2023. Cotracker: It is better to track together. arXiv:2307.07635.

Kidger, P., Lyons, T.J., 2020. Universal approximation with deep narrow networks. In: COLT. pp. 2306–2327.

Kirillov, A., Mintun, E., Ravi, N., Mao, H., Rolland, C., Gustafson, L., Xiao, T., Whitehead, S., Berg, A.C., Lo, W.Y., Dollar, P., Girshick, R., 2023. Segment anything. In: ICCV. pp. 4015–4026.

Lei, J., Sridhar, S., Guerrero, P., Sung, M., Mitra, N., Guibas, L.J., 2020. Pix2Surf: Learning Parametric 3D Surface Models of Objects from Images. In: ECCV. pp. 121–138.

Lowe, D., 1999. Object recognition from local scale-invariant features. In: ICCV. pp. 1150–1157.

Luque, A., Parent, D., Colomé, A., Ocampo-Martinez, C., Torras, C., 2024. Model predictive control for dynamic cloth manipulation: Parameter learning and experimental validation. IEEE Trans. Control Syst. Technol. 1–17.

Magnenat, S., Ngo, D.T., Zünd, F., Ryffel, M., Noris, G., Rothlin, G., Marra, A., Nitti, M., Fua, P., Gross, M., Sumner, R.W., 2015. Live texturing of augmented reality characters from colored drawings. IEEE Trans. Vis. Comput. Graphics 21 (11), 1201–1210.

Mildenhall, B., Srinivasan, P.P., Tancik, M., Barron, J.T., Ramamoorthi, R., Ng, R., 2020. NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis. In: ECCV. pp. 405–421.

Montagnat, J., Delingette, H., Ayache, N., 2001. A review of deformable surfaces: topology, geometry and deformation. Image Vis. Comput. 19 (14), 1023–1040.

Ngo, T.D., Östlund, J., Fua, P., 2016. Template-based monocular 3D shape recovery using Laplacian meshes. IEEE Trans. Pattern Anal. Mach. Intell. 38 (1), 172–187.

Ngo, D.T., Park, S., Jorstad, A., Crivellaro, A., Yoo, C.D., Fua, P., 2015. Dense image registration and deformable surface reconstruction in presence of occlusions and minimal texture. In: ICCV. pp. 2273–2281.

Östlund, J., Varol, A., Ngo, D., Fua, P., 2012. Laplacian meshes for monocular 3D shape recovery. In: ECCV. pp. 412–425.

Ozuysal, M., Calonder, M., Lepetit, V., Fua, P., 2010. Fast keypoint recognition using random ferns. IEEE Trans. Pattern Anal. Mach. Intell. 32 (3), 448–461.

Parashar, S., Pizarro, D., Bartoli, A., Collins, T., 2015. As-rigid-as-possible volumetric shape-from-template. In: ICCV. pp. 891–899.

Park, J.J., Florence, P., Straub, J., Newcombe, R., Lovegrove, S., 2019. DeepSDF: Learning continuous signed distance functions for shape representation. In: CVPR. pp. 165–174.

Pumarola, A., Agudo, A., Porzi, L., Sanfeliu, A., Lepetit, V., Moreno-Noguer, F., 2018. Geometry-aware network for non-rigid shape prediction from a single view. In: CVPR. pp. 4681–4690.

Salzmann, M., Fua, P., 2009. Reconstructing sharply folding surfaces: A convex formulation. In: CVPR. pp. 1054–1061.

Salzmann, M., Fua, P., 2011. Linear local models for monocular reconstruction of deformable surfaces. IEEE Trans. Pattern Anal. Mach. Intell. 33 (5), 931–944.

Salzmann, M., Hartley, R., Fua, P., 2007a. Convex optimization for deformable surface 3-D tracking. In: ICCV. pp. 1–8.

Salzmann, M., Lepetit, V., Fua, P., 2007b. Deformable surface tracking ambiguities. In: CVPR. pp. 1–8.

Salzmann, M., Moreno-Noguer, F., Lepetit, V., Fua, P., 2008a. Closed-form solution to non-rigid 3D surface registration. In: CVPR. pp. 581–594.

Salzmann, M., Pilet, J., Ilic, S., Fua, P., 2007c. Surface deformation models for nonrigid 3D shape recovery. IEEE Trans. Pattern Anal. Mach. Intell. 29 (8), 1481–1487.

Salzmann, M., Urtasun, R., 2010. Implicitly constrained Gaussian process regression for monocular non-rigid pose estimation. In: NeurIPS. pp. 2065–2073.

Salzmann, M., Urtasun, R., Fua, P., 2008b. Local deformation models for monocular 3D shape recovery. In: CVPR. pp. 1–8.

Sanchez, J., Corrales, J.A., Bouzgarrou, B.C., Mezouar, Y., 2018. Robotic manipulation and sensing of deformable objects in domestic and industrial applications: a survey. Int. J. Robotics Res. 37 (7), 688–716.

Sorkine, O., Alexa, M., 2007. As-rigid-as-possible surface modeling. In: Geometry Processing. The Eurographics Association, pp. 109–116.

Tretschk, E., Kairanda, N., B. R., M., Dabral, R., Kortylewski, A., Egger, B., Habermann, M., Fua, P., Theobalt, C., Golyanik, V., 2023. State of the art in dense monocular non-rigid 3D reconstruction. In: Computer Graphics Forum (Eurographics State of the Art Reports).

Varol, A., Salzmann, M., Fua, P., Urtasun, R., 2012. A constrained latent variable model. In: CVPR. pp. 2248–2255.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł., Polosukhin, I., 2017. Attention is all you need. In: NeurIPS, Vol. 30. pp. 1–11.

Virtanen, P., Gommers, R., Oliphant, T.E., Haberland, M., Reddy, T., Cournapeau, D., Burovski, E., Peterson, P., Weckesser, W., Bright, J., van der Walt, S.J., Brett, M., Wilson, J., Millman, K.J., Mayorov, N., Nelson, A.R.J., Jones, E., Kern, R., Larson, E., Carey, C.J., Polat, İ., Feng, Y., Moore, E.W., VanderPlas, J., Laxalde, D., Perktold, J., Cimrman, R., Henriksen, I., Quintero, E.A., Harris, C.R., Archibald, A.M., Ribeiro, A.H., Pedregosa, F., van Mulbregt, P., SciPy 1.0 Contributors, 2020. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. Nature Methods 17 (3), 261–272.

Vogler, C., Goldenstein, S., Stolfi, J., Pavlovic, V., Metaxas, D., 2007. Outlier rejection in high-dimensional deformable models. Image Vis. Comput. 25 (3), 274–284.

Wang, T., Ling, H., Lang, C., Feng, S., Hou, X., 2019. Deformable surface tracking by graph matching. In: ICCV. pp. 901–910.

Yin, H., Varava, A., Kragic, D., 2021. Modeling, learning, perception, and control methods for deformable object manipulation. Science Robotics 6 (54), 1–16.

Yu, R., Russell, C., Campbell, N.D.F., Agapito, L., 2015. Direct, dense, and deformable: Template-based non-rigid 3D reconstruction from RGB video. In: ICCV. pp. 918–926.

Yüce, G., Ortiz-Jiménez, G., Besbinar, B., Frossard, P., 2022. A structured dictionary perspective on implicit neural representations. In: CVPR. pp. 19228–19238.

Zienkiewicz, O., Taylor, R., Zhu, J., 2013. Chapter 6 - shape functions, derivatives, and integration. In: Zienkiewicz, O., Taylor, R., Zhu, J. (Eds.), The Finite Element Method: Its Basis and Fundamentals. pp. 151–209.