

# Weakly-Supervised Deep Shape-from-Template

SARA LUENGO-SANCHEZ<sup>1</sup>, DAVID FUENTES-JIMENEZ<sup>1</sup>, CRISTINA LOSADA-GUTIERREZ<sup>1</sup>, DANIEL PIZARRO<sup>1</sup>, ADRIEN BARTOLI<sup>2</sup>.

<sup>1</sup>Department of Electronics, Universidad de Alcalá (UAH), E-28871 Alcalá de Henares, Spain

<sup>2</sup>EnCoV, 63000 Clermont-Ferrand, France

Corresponding author: Sara Luengo-Sanchez (e-mail: sara.luengo@uah.es).

**ABSTRACT** We propose WS-DeepSfT, a novel deep learning-based approach to the Shape-from-Template (SfT) problem, which aims at reconstructing the 3D shape of a deformable object from a single RGB image and a template. WS-DeepSfT addresses the limitations of existing SfT techniques by combining a weakly-supervised deep neural network (DNN) for registration and a classical As-Rigid-As-Possible (ARAP) algorithm for 3D reconstruction. Unlike previous deep learning-based SfT methods, which require extensive synthetic data and depth sensors for training, WS-DeepSfT only requires regular RGB video of the deforming object and a segmentation mask to discriminate the object from the background. The registration model is trained without synthetic data, using videos where the object undergoes deformations, while ARAP does not require training and infers the 3D shape in real-time with minimal overhead. We show that WS-DeepSfT outperforms the state-of-the-art, in both accuracy and robustness, without requiring depth sensors or synthetic data generation. WS-DeepSfT thus offers a robust, efficient, and scalable approach to SfT, bringing it closer to applications such as augmented reality.

**INDEX TERMS** Non-rigid, Shape-from-Template, weak-supervision, registration, wide-baseline, template-based, 3D reconstruction

## I. INTRODUCTION

In Shape-from-Template (SfT) [1], the objective is to infer the 3D shape of a deformable object from a single input image and the registration of this image to an object template. The template consists of a reference 3D shape in a known position and deformation, usually represented by a 3D mesh, along with a texture map that describes its appearance. Solving SfT involves two main tasks: 1) registration, where a transformation or warp function is found that matches points on the reference 3D shape with pixel positions in the input image, and 2) reconstruction, where the deformed 3D shape is recovered in the camera coordinate system. SfT has potential applications in computer graphics by enabling the digital scanning of object deformations to be used as digital 3D assets [2]–[4]. It is also relevant in the industry, where it may be used to analyse deformations in materials or structures to improve their design, safety, or understanding, such as for aerospace manufacturing [5], [6], landing mats in gymnastics [7], clothes [8], and in medicine, for analysing deformations in organs [9], [10] and human injuries [11]. SfT is also crucial for developing Augmented Reality (AR) appli-

cations, where a computer-generated 3D object is rendered and overlaid onto a live image stream. SfT enables AR by estimating the object deformation in the camera coordinate system, allowing augmentations decided on the reference model to be rendered and overlaid in the real images. AR can be used in medicine to design Computer-Aided Interventions (CAI) tools, helping surgeons conduct minimally invasive surgeries [12]–[15]. These medical operations are performed with an endoscope to limit the size of incisions and reduce recovery time. However, using these tools diminish surgeon perception, creating a highly challenging scenario, even for experienced surgeons. To help overcome these challenges, AR displays 3D representations of the organ's surface and tumours, enhancing the surgeon's perception [16], [17].

Solving SfT is highly challenging due to the demanding conditions in real settings. Specifically, solving the registration task requires finding visual correspondences between the template texture map and a single input image. These two domains are usually significantly distinct, mainly due to projection distortion, surface deformations, occlusions, self-occlusions, and lighting conditions. Solving the SfT

reconstruction task is also highly complex, since an infinite number of 3D shapes share the same 2D projection and thus, are compatible with the estimated warp. A deformation constraint is commonly applied to limit the space of possible solutions. There exist multiple deformation models, either purely geometrical, such as isometry (preserving geodesic distances) [18], [19], conformity (preserving surface angles) [1], [20], equiareality (preserving surface areas) [21], or physically inspired, such as material elasticity [22], [23]. We consider that the object of interest undergoes approximate isometric deformations, which is accurate for many real objects and deformations, and represents the most frequently used deformation model in SfT.

Existing methods for solving isometric SfT can be categorised into classical methods and deep learning methods. Classical methods use feature matching to solve the registration step and non-convex optimisation to reconstruct the deformed shape. These methods fail when facing imaging or texture challenging conditions. Furthermore, the non-convex optimisation process requires a good initialisation and usually limits its deployment in real-time applications. Recently, differentiable physics simulators and renderers have been incorporated to SfT methods, increasing the accuracy of the 3D reconstructions at the expense of even higher optimisation time. On the other hand, deep learning methods solve SfT by training a regression function to solve the registration and reconstruction tasks in real time. However, these methods require a large amount of labeled data, which is extremely complicated to obtain for real scenes. Therefore, they rely on generating synthetic data to train these models, downgrading performance in real settings.

The first complete learning pipeline for solving SfT, DeepSfT, was proposed in [24]. DeepSfT achieves more accurate results than previous SfT methods while running in real-time on GPU-based hardware using a Deep Neural Network (DNN). However, DeepSfT suffers from major drawbacks. It requires large amounts of labelled synthetic data and real data captured with RGB-D sensors. Its performance depends significantly on the quality of the synthetic data, making it crucial for the geometry and deformation simulations to closely match the real scenes in order to overcome the render gap. Therefore, an expert-level knowledge of advanced physical simulators is needed to generate such data. A fine-tuning step is also necessary when applying this method to real scenes, where the labelling obtained from the depth sensor is used to train the DNN along with a photometric loss. The convergence of this training process is affected by external environmental conditions, such as lighting and a lack of rich texture in the template. Finally, a post-processing step is required to find the complete deformed surface, as the DNN model only recovers a depth map of the surface's visible part.

We propose WS-DeepSfT, a new SfT solution based on a hybrid architecture, where the registration step is solved with a DNN model and the reconstruction step is based on a classic mesh processing algorithm. We use a weakly-

supervised approach to train the registration DNN model using only RGB videos of the object undergoing deformations, not requiring synthetic data generation or depth sensors. The only requirement is that the object can be segmented from the background and that some frames in the training videos show the object in its reference shape, *i.e.* that of the template. This is an important step in making SfT methods useful in real applications by allowing training with virtually any object that can be recorded in video. For the reconstruction step we use the As-Rigid-As-Possible (ARAP) [25] algorithm to infer the 3D shape of the object from the registration solution, removing the necessity of depth sensors or training. Contrary to other classical methods, ARAP can be executed in real time and only requires a few iterations to reach convergence. We show that the proposed WS-DeepSfT is comparable to or better than DeepSfT, even when the latter is supervisory trained, and achieves real-time processing. We show our results in two new synthetic and three new real datasets to evaluate the performance of the proposed solution and compare it with the existing approaches.

This paper is divided into the following sections. Section II reviews the start-of-the-art methods in SfT. Section III presents the SfT problem and the proposed solution principles. Sections IV and V elaborate on the proposed registration and reconstruction steps. Section VI states the experimental setup and evaluation methods. It discusses the choice of WS-DeepSfT's hyperparameters in detail. Finally, Section VII reports conclusions and proposes future lines of research.

## II. PREVIOUS WORK

We review previous SfT works by distinguishing between classical methods in Section II-A and learning-based methods in Section II-B. Classical methods are based on optimization and use deformation models. In contrast, learning-based methods train a neural network to estimate the SfT outputs from its inputs. Table 1 summarizes the characteristics of the different methods presented throughout this section and compares them with the proposed solution.

### A. CLASSICAL METHODS

This section covers analytical methods, which model the reconstruction step as Partial Differential Equations (PDEs) after assuming registration is solved, followed by Energy-based methods, based on minimising a cost function for registration and reconstruction, either separately (decoupled methods) or simultaneously (integrated methods). Finally, recent integrated methods based on differentiable renderers and physics simulators are discussed.

#### 1) Analytical Methods

Analytical methods tackle the SfT problem by formulating constraints as PDEs, yielding well-posed convex solutions in a single step. However, these methods are often less accurate than others and require a refinement step using iterative methods. As a result, they are typically used as an initialisation

SfT method	Baseline capability	Solves registration	Solves reconstruction	Needs training	Needs labels	Accuracy	Inference time	References
Analytical methods	Wide	No	Yes	No	No	Low	Low	[1], [18] [21], [26], [27]
Energy-based methods (integrated)	Short	Yes	Yes	No	No	Depends on conditions	Low	[28]–[31]
Energy-based methods (decoupled)	Wide	Yes	Yes	No	No	Depends on conditions	Medium	[19], [23] [22], [32]–[37]
Physics simulation methods	Wide	Yes	Yes	Yes	No	High	High	[38]–[40]
Deep learning methods	Wide	Yes	Yes	Yes	Yes	Depends on synthetic data	Low	[24], [41]–[44]
WS-DeepSfT	Wide	Yes	Yes	Yes	No, needs keyframes	High	Low	Proposed

TABLE 1: Summary of the state-of-the-art methods presented in Section II.

step for non-convex, energy-based approaches. These methods are commonly classified according to the assumptions made about the deformation model. Specifically, [1], [18], [26], [27] address isometric deformations by solving a system of non-linear first-order PDEs. While isometry has been the most extensively studied model, there exist other deformation models, such as conformity [1], preserving local angles, or equiareality [21], preserving local areas. Both the conformal and equiareal models introduce ambiguities that must be resolved by introducing other image cues or boundary conditions, raising major concerns on their practical applicability.

## 2) Energy-based Methods

Energy-based methods minimise a non-convex cost function through iterative optimisation, typically consisting of a data term that ensures photometric or reprojection consistency and a regularisation term based on deformation priors. These methods are categorised into two groups: 1) integrated methods [28]–[31], where the registration and reconstruction are estimated jointly, and 2) decoupled methods [19], [22], [23], [32]–[35], where registration and reconstruction are solved independently.

Integrated methods [28]–[31] perform registration and reconstruction simultaneously. They are mainly used in short-baseline scenarios, such as continuous video streams [28]–[30], where they minimise a non-convex function by deforming the 3D reference template until its projection aligns with the input image. However, due to their non-convexity, these methods may converge to incorrect local minima and fail when the short-baseline conditions are not met. Despite this drawback, these approaches are highly effective in handling complex, high-frequency deformations and can often do so densely and in real time. Nevertheless, they require a good initialisation [31] and additional information, such as point correspondences [28], pixel-level photo-consistency models

[29], [30], or temporal consistency [31].

Decoupled methods [19], [22], [23], [32]–[37] estimate the registration and reconstruction functions separately. While they use well-established registration techniques, they tend to produce sub-optimal results by not fully exploiting the interdependencies that bind between registration and reconstruction. These methods typically begin by solving registration using keypoint-based feature matching algorithms such as SIFT [35], [36], often incorporating filtering [22], [37] or mismatch removal [35] to minimise errors. These approaches operate on individual images and do not require temporal consistency, performing well even under significant deformations. However, when the feature-matching algorithm fails due to challenging image or texture conditions, its errors propagate to the registration and reconstruction of the object. Moreover, these methods use an optimisation process to improve accuracy, which limits their application in real-time wide-baseline scenarios to simple objects and simple deformations. In order to expand the scope to more complex deformations, RobuSfT [35] uses multi-core parallelisation and removes ambiguities by preserving the neighbourhood structure of matches. It improves previous methods in robustness to occlusions, large deformations and fast motions, although its accuracy still relies on the precision and number of the established point matches.

## 3) Physics Simulation Methods

Classical SfT techniques rely on deformation models to constrain the range of potential solutions. However, the assumptions underlying these models are often not entirely met, resulting in inaccuracies in the final reconstructions. A more recent research direction within integrated methods incorporates differentiable physics simulations and renderers [38]–[40]. These approaches aim to improve the solution quality by constraining an object’s motion based on physically mean-

ingful parameters. They provide smoother, higher-quality 3D reconstructions and reduce reprojection errors compared to classical methods. Nevertheless, these techniques typically require several minutes or even hours to optimise a single scene, making them unsuitable for real-time applications.

The method proposed in [38] is the first to integrate a differentiable physics simulator and renderer to optimise forces and material properties. Using accurate elastic deformation priors, it addresses challenging cloth deformations and simulates physical processes to resolve complex local folds. By minimising the pixel-wise photometric error between the input image and the estimated surface reprojection, it effectively uses texture information without being constrained by the mesh resolution. While this approach achieves high physical realism in surface deformations, it struggles with capturing small, high-frequency wrinkles.

In contrast, [39] introduces a physics-based neural network that infers the shape and physical properties of cloth in monocular video sequences, accounting for inertia and force interactions. This neural network provides stable simulations of cloth dynamics, ensuring physically plausible reconstructions. It is trained in a self-supervised manner by minimising the photometric error between rendered and real-world video frames. This method significantly reduces computation time while achieving results comparable to [38].

The method in [40] introduces contour and perspective constraints into a physics simulator to restrict the template's movement range, improving convergence and optimisation accuracy by reducing shape inference errors in non-visible areas. The method initially relies on [35] to estimate the template's position, although this can be prone to errors due to insufficient feature points or mismatches. Following this, a mass-spring mechanical simulator is used to enforce material properties and deformation laws on the estimate, while bilateral mesh denoising ensures smoothness. This approach provides a marked improvement over RobuSfT in scenarios involving complex bending and deformation.

## B. DEEP LEARNING METHODS

Deep learning methods learn a mapping function between the input image and the SfT solution. These approaches solve registration and reconstruction simultaneously in a single feed-forward pass, significantly improving runtime performance compared to energy-based methods. However, these approaches typically require annotated data, which is almost impossible to acquire for registration in real-world scenes. Additionally, most methods are limited to flat templates or regular meshes with a restricted and fixed number of vertices. These methods differ mainly in how they parameterise the 3D coordinates and in their learning strategies. Moreover, most are texture- and template-specific, meaning that a model only handles one object, hence with limited ability to generalise to novel shapes, although recent research has shown promising results in generalising across shape families. Based on this,

deep learning methods can be categorised into object-specific and generic approaches.

### 1) Object Specific

The first solutions using DNN-based approaches [41], [42] were limited to low-vertex, regular, and rectangular templates, with performance decreasing as the number of vertices increased due to the corresponding growth in network size. These methods are fully supervised and thus require labelled data, which is notoriously difficult to obtain for real-world scenes. Consequently, they rely on synthetically generated data, which reduces their accuracy in real scenes due to the rendering gap between simulated and real environments.

DeformNet [41] addresses SfT by iteratively predicting the position of each 3D template vertex in the image. The estimated 2D coordinate of the vertex is progressively refined in three iterations. A depth estimation network is then used to reconstruct the 3D mesh. However, the mesh is limited to  $10 \times 10$  vertices arranged regularly, restricting the method's broader applicability.

HDM-net [42] uses three-channel 2D maps to model the 3D coordinates of a regular  $73 \times 73$  vertex mesh. The loss function is the difference between the network's outputs and the reconstruction labels from a synthetic dataset.

DeepSfT [24] is trained for a specific template (in terms of shape and texture map) by embedding this information into the DNN's weights. The network outputs pixel-level depth and registration maps of the template's visible surface in the input image, allowing it to represent 3D objects of arbitrary and complex shapes. A post-processing step using the ARAP model is required to recover the hidden parts of the surface. Initially trained on synthetic data, DeepSfT is fine-tuned using real data with a photometric loss and a supervised depth loss, requiring an RGB-D sensor for this step. Its key strengths are its high accuracy and ability to recover shapes with any number of vertices.

### 2) Object Generic

IsMo-GAN [43] builds on the work of [42] by using adversarial learning to train both an object-detection network and a discriminator. This approach improves reconstruction accuracy in real images compared to earlier methods and performs well on textureless surfaces.

Texture-generic DeepSfT [44] extends the work of [24], making the method adaptable to new texture maps at runtime without the need for fine-tuning, thus achieving texture invariance. The architecture comprises two neural networks: a segmentation DNN to detect the template and an SfT module to solve registration and reconstruction. Both networks require the template's texture map as input to facilitate texture generalisation. Similarly to its predecessor, a post-processing step using the ARAP model is needed to recover the full 3D shape. This method achieves comparable results to previous DNN texture-specific approaches and performs well under challenging deformation and imaging conditions.



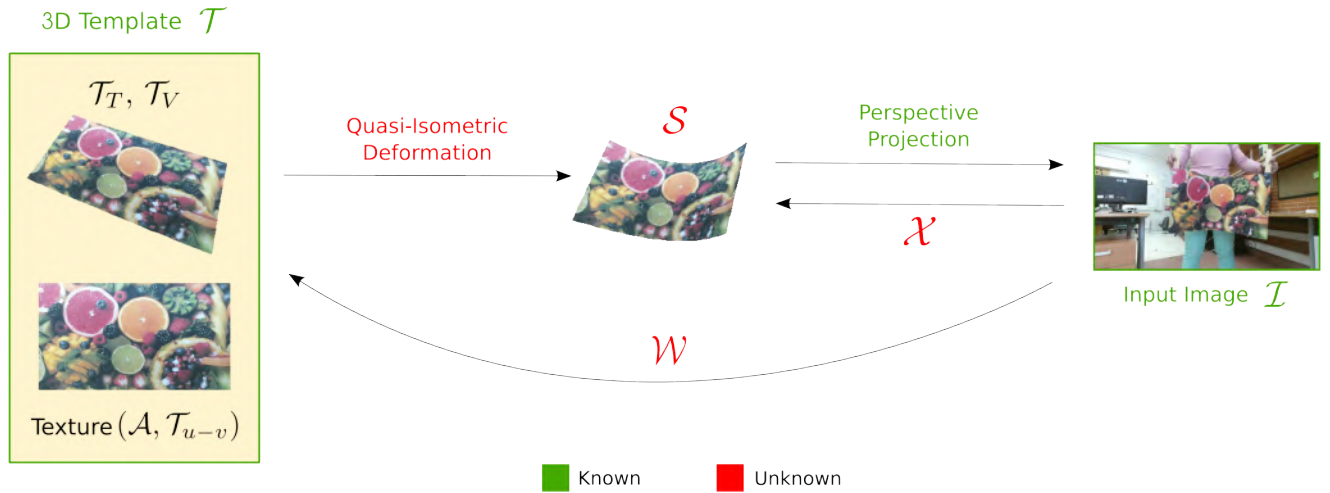


FIGURE 1: Geometric modelling of SfT. Data written in green is known a priori, meanwhile data written in red needs to be estimated to solve SfT.

### III. METHODOLOGY

#### A. PROBLEM DESCRIPTION

This section covers the basic concepts that define the SfT problem. Figure 1 depicts the main elements and functions involved in SfT.

The template  $\mathcal{T}$  is composed of two primary components: a 3D mesh and its associated texture. The 3D mesh is defined by a set of vertices ( $\mathcal{T}_V$ ) and their connectivity information ( $\mathcal{T}_T$  for triangles). The texture is defined by an RGB image  $\mathcal{A}$  of size  $H_{\mathcal{T}} \times W_{\mathcal{T}}$ , along with a set of UV coordinates  $\mathcal{T}_{u-v}$  that map each vertex (and thus each triangle) to specific locations within the texture image. The deformed surface  $\mathcal{S}$  is also represented as a 3D mesh that shares the topology and appearance of the template. It is defined by a set of vertices  $\mathcal{S}_V$ , which differ from those of the template while maintaining the template triangulation  $\mathcal{T}_T$  and texture  $\mathcal{A}$ . The deformed surface  $\mathcal{S}$  and the template  $\mathcal{T}$  are related by a quasi-isometric deformation (see Section V). The input image  $\mathcal{I}$  is an  $H \times W$  RGB image showing the projection of  $\mathcal{S}$  through a pinhole camera with known intrinsic parameters. Due to self and external occlusions, only a subset of  $\mathcal{S}$  is visible in the image. We define the visibility in the input image as a binary mask  $\mathcal{M}$ , the same size as  $\mathcal{I}$ . The image points showing the surface projection are labelled as “1” in the mask and “0” otherwise.

The objective of SfT is to find  $\mathcal{S}$  from  $\mathcal{I}$  and  $\mathcal{T}$ . Solving SfT involves a registration step and a reconstruction step. The registration step consists in finding correspondences between the template  $\mathcal{T}$  and the image  $\mathcal{I}$ . The warp  $\eta : (u, v) \rightarrow (u_{\mathcal{T}}, v_{\mathcal{T}})$  defines the mapping between image coordinates  $(u, v)$  and texture coordinates  $(u_{\mathcal{T}}, v_{\mathcal{T}})$  in the template. We approximate  $\eta$  as a discrete two-channel image  $\mathcal{W}$ , which has the same size as the input image:

$$\mathcal{W}(u, v) = \begin{cases} \eta(u, v) & \mathcal{M}(u, v) = 1 \\ (-1, -1) & \mathcal{M}(u, v) = 0 \end{cases} \quad (1)$$

$$(u, v) \in [0, \dots, H-1] \times [0, \dots, W-1] \quad (2)$$

The coordinates  $(-1, -1)$  lie outside the template texture and indicates that the surface is not visible at that particular position, incorporating the mask  $\mathcal{M}$  into  $\mathcal{W}$ . The reconstruction step involves finding the 3D shape of the surface, defined by the set of vertices  $\mathcal{S}_V$ , from the registration warp and the template. We define the surface embedding function  $\mathcal{X} : (u, v) \rightarrow (x, y, \rho)$ , which maps image coordinates  $(u, v)$  to 3D points  $(x, y, \rho)$  that belong to the surface with respect to the camera’s projection origin. The only unknown in recovering  $\mathcal{X}$  is the surface depth  $\rho$ , as the other two components, *i.e.*,  $x$  and  $y$ , can be obtained from the image coordinates  $(u, v)$ , the intrinsic camera parameters, and  $\rho$  using the following expressions:

$$x(u, v) = \frac{\rho(u, v)}{f_x}(u - u_0) \quad (3)$$

$$y(u, v) = \frac{\rho(u, v)}{f_y}(v - v_0) \quad (4)$$

where  $f_x$  and  $f_y$  are the horizontal and vertical focal lengths, and  $u_0$  and  $v_0$  are the image coordinates of the principal point. We define the surface depth map as the following single-channel image  $\mathcal{D}$ :

$$\mathcal{D}(u, v) = \begin{cases} \rho(u, v) & \mathcal{M}(u, v) = 1 \\ -1 & \mathcal{M}(u, v) = 0 \end{cases} \quad (5)$$

$$(u, v) \in [0, \dots, H-1] \times [0, \dots, W-1] \quad (6)$$

In this step, the mask  $\mathcal{M}(u, v)$  is inferred from the registration warp  $\mathcal{W}(u, v)$ , where points not belonging to the template are assigned a  $(-1, -1)$  value.

Existing SfT methods compute the depth map and then recover the rest of the surface that is not visible in the input image. In contrast, we compute the surface directly from the warp using a mesh processing step.

### B. PROPOSED SOLUTION METHOD

Figure 2 shows a general diagram of the proposed SfT method. We use a DNN encoder-decoder architecture to solve the registration step. This DNN takes the input image  $\mathcal{I}$  and the segmentation mask  $\mathcal{M}$  as inputs and estimates the registration warp image  $\mathcal{W}$ . We propose a weakly-supervised learning process to train the registration DNN, removing the necessity for synthetic image generation and RGB-D sensors, which are required by DeepSfT.

Our training data consist of unlabelled, raw real video sequences gathered and saved frame-by-frame without pre-processing. The training sequences must fulfill three main requirements: 1) the first frame of each sequence shows a rigid transformation of the template reference shape, 2) the deformation and pose changes between consecutive frames is mild, allowing optical flow methods to generate point tracks, and 3) the object can be segmented from the background. This process is further explained in Section IV.

Once the registration has been solved, we apply the ARAP algorithm to solve the reconstruction problem from the estimated warp. This method takes the previously obtained registration map  $\mathcal{W}$  and the template  $\mathcal{T}$  as input and estimates the deformed surface  $\mathcal{S}$ . Since ARAP does not need training, it can be directly used with real data. This process is further explained in Section V.

## IV. REGISTRATION STEP

This section expands on the proposed registration DNN model and weakly-supervised training method.

### A. NETWORK ARCHITECTURE

Figure 3 shows the encoder-decoder architecture used to build the registration DNN. It is based on the Main Block used in the DeepSfT DNN [24], so it uses convolutional and deconvolutional residual feed-forward structures based on the ResNet50 model. This design propagates the information to deeper layers while preserving the spatial high-frequency details. We have removed the output dedicated to the depth map in DeepSfT, which is not needed in our design, and added the segmentation mask  $\mathcal{M}$  as an additional input. We model the registration DNN as the function  $\overline{\mathcal{W}} = D_W(\mathcal{I}, \mathcal{M}, \theta_w)$ , where  $\theta_w$  are the network weights and  $\overline{\mathcal{W}}$  is the estimated warp image. The estimated warp  $\overline{\mathcal{W}}$  is filtered after inference using the mask  $\mathcal{M}$  to remove invalid points.

### B. WEAKLY-SUPERVISED TRAINING

We propose a new weakly-supervised training method for the registration DNN that does not require direct supervision and can be trained directly on real data.

#### 1) Principles and Training Data

The training data must meet several conditions. First, the data should be acquired as video sequences where the surface deforms gradually from its rest shape, which corresponds to the template shape, ensuring that the baseline conditions between frames are minimal. The core idea is to use optical flow methods to compute correspondences between adjacent frames in the sequence, which then serve as weak supervision. Second, we require some frames in the sequence to display the template shape, subject to an unknown rigid transformation that can be computed later using robust rigid tracking methods [45] [46]. We refer to these frames as “keyframes,” and assume they are identified in the sequence, e.g., the first frame of each video sequence is a keyframe. Third, segmentation of the object of interest from the background is required for all frames in the sequence. To achieve this, we use foundational segmentation models to segment the input images with minimal user interaction. We use Track Anything [47], a tool based on the Segment Anything Model (SAM) [48], which extends the segmentation from SAM across the full length of a video sequence. Track Anything allows us to segment an entire input sequence based on the results of applying SAM to just the first frame of the sequence. This simplifies the segmentation process during training and makes it accessible to non-expert users. It is important to highlight that the previous requirements only apply to the training sequences. During inference, the method works independently for each single image and does not require optical flow methods, which makes it able to cope with wide-baseline conditions.

Based on the above data, we propose to train the registration DNN given in Section IV-A with a compound loss as shown in (7), involving three terms: the matching loss  $L_M$ , enforcing the estimated optical flow between frames, the rigidity loss  $L_R$ , including the pose estimation in the keyframes, and a regularisation term  $L_S$ , enhancing the smoothness of the solution. Each term in the equation will be introduced in the following sections:

$$L = \lambda_M L_M + \lambda_R L_R + \lambda_S L_S \quad (7)$$

#### 2) Matching Loss

We construct the matching loss by computing the optical flow between pairs of consecutive images  $\mathcal{I}_i$  and  $\mathcal{I}_j$  with  $j = i+1$ . Given a position  $(u, v)$  in image  $\mathcal{I}_i$ , the optical flow allows us to find its correspondence in image  $\mathcal{I}_j$  at  $(u + \delta_u, v + \delta_v)$ . The correspondence  $(u, v) \rightarrow (u + \delta_u, v + \delta_v)$  is valid if it is a visible point on the surface in both images, meaning  $\mathcal{M}_i(u, v) = 1$ , and  $\mathcal{M}_j(u + \delta_u, v + \delta_v) = 1$ . Using the registration DNN, we compute the two warp images  $\overline{\mathcal{W}}_i$  and  $\overline{\mathcal{W}}_j$  corresponding to  $\mathcal{I}_i$  and  $\mathcal{I}_j$ . Since the SfT warp maps points from the image plane to template coordinates, a correspondence between two images should point to the same template coordinate (see Figure 4 for a graphical depiction). Thus, we aim to minimise the following Euclidean distance

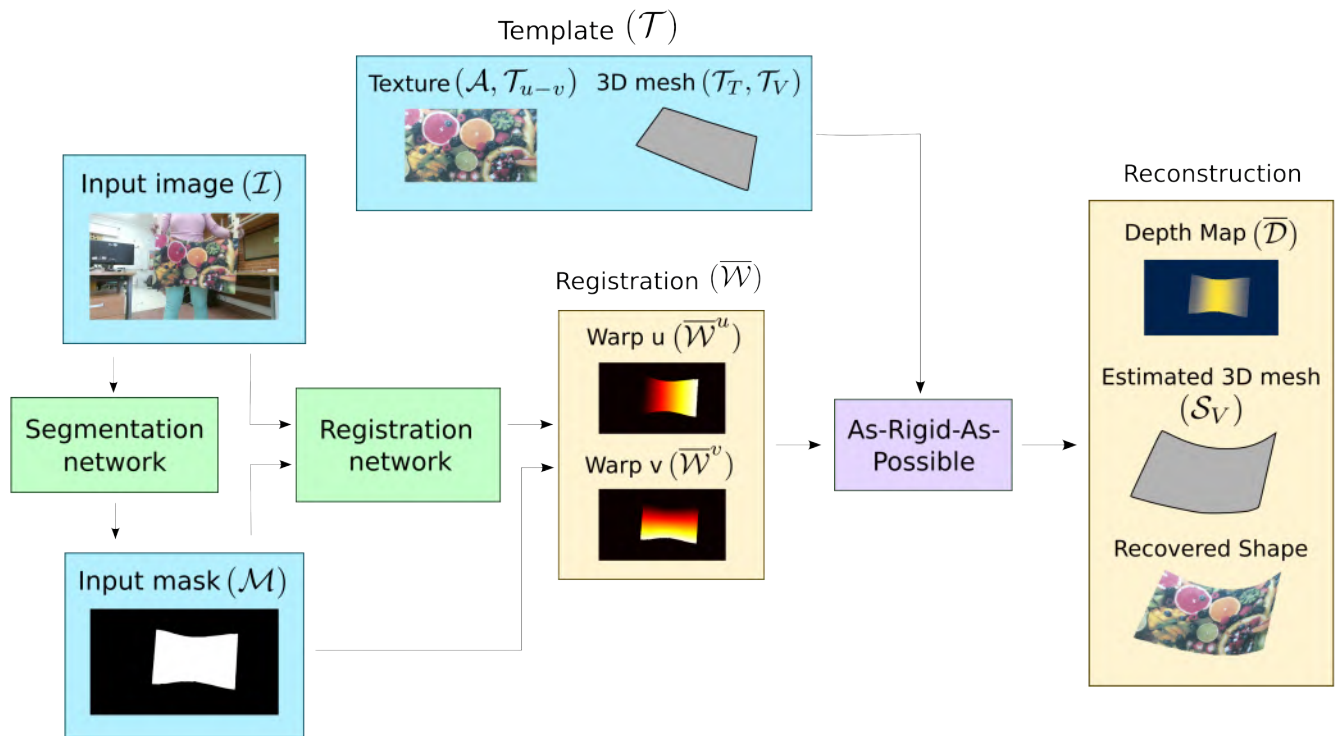


FIGURE 2: Proposed Shape-from-Template method architecture. The inputs of the networks are coloured in blue, the learning-based algorithms are in green meanwhile the non-learning ones are in purple, and the outputs of the system are in yellow.

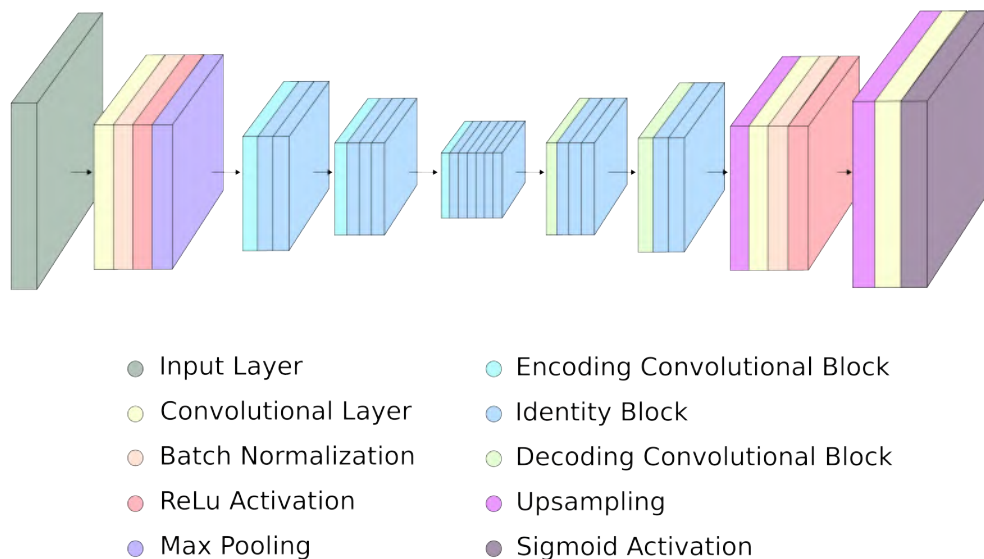


FIGURE 3: Registration Network Architecture: We adopt a modified version of the architecture proposed in [24], a Fully Convolutional Network (FCN) based on the ResNet50. In our version, we introduce the image mask as an additional input, remove the depth output, and replace the final activation layer with a sigmoid layer.

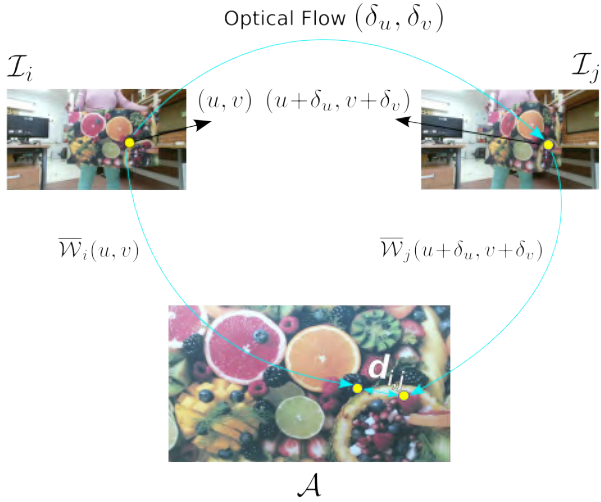


FIGURE 4: Graphical depiction of the matching loss  $L_M$ . The optical flow  $(\delta_u, \delta_v)$  is computed using PWC-Net. The registration maps  $\overline{\mathcal{W}}_i$  and  $\overline{\mathcal{W}}_j$ , estimated by the registration network introduced in Section IV-A, are used to establish correspondences between frames.  $\mathcal{I}_i$  and  $\mathcal{I}_j$  represent the input images, and  $\mathcal{A}$  denotes the known texture map of the template.

during training:

$$d_{i,j}(u, v) = \|\overline{\mathcal{W}}_i(u, v) - \overline{\mathcal{W}}_j(u + \delta_u, v + \delta_v)\|_2. \quad (8)$$

The matching loss is designed so that the distance in (8) is minimised across pairs of frames in the training sequences and over all valid correspondences. To construct the loss, we generate a batch of images grouped into  $N_g$  randomly selected sets of frames. Each group  $k$  consists of  $N_f$  frames  $\mathcal{I}_{k,i}$  for  $i = 1, \dots, N_f$ , taken from the same sequence and sorted chronologically. Frames are spaced within the group following a uniform distribution with a maximum frame gap, which is a training parameter tuned in our experiments. Keeping a small gap is important to ensure that the optical flow method produces accurate flow estimates. We compute the optical flow between consecutive frames in each group, resulting in  $N_f - 1$  pairs of optical flow maps. We denote  $d_{i,j}^k(u, v)$  as the distance defined in (8) for frames  $i$  and  $j$  in group  $k$ , corresponding to image coordinates  $(u, v)$ . The loss is defined as follows:

$$L_M = \gamma \sum_{k=1}^{N_g} \sum_{i=1}^{N_f-1} \sum_{(u,v) \in \mathcal{R}_{i,j}^k} w(d_{i,i+1}^k(u, v), c_M), \quad (9)$$

where  $\mathcal{R}_{i,j}^k$  represents the region of valid correspondences for the pair of frames  $i, j$  in group  $k$ , and  $\gamma$  is a normalisation factor depending on the number of points used in the loss. The function  $w$  is the Welsch Loss [49]:

$$w(x, c) = 1 - \exp\left(-\frac{1}{2} \left(\frac{x}{c}\right)^2\right), \quad (10)$$

where  $c = c_M$  is a scalar hyperparameter that controls the influence of residuals and sensitivity to large errors in the optical flow estimation that might affect the training process. The diagram in Figure 5 visually represents the explained methodology. In our experiments, we use PWC-Net [50] to compute the optical flow although any other modern method would be valid. More details are given in Section IV-C.

### 3) Rigidity Loss

The matching loss introduced in the previous section cannot be used alone to train the registration DNN, as it suffers from degeneracy. The optimisation process can indeed reach a global minimum of  $L_M$  by mapping all image positions to the same coordinates in the template, resulting in  $L_M = 0$ . Therefore, it is necessary to complement this loss with an additional loss that prevents the solution from “shrinking” to a single template coordinate. To address this, we design the rigidity loss, a function only applicable to “keyframes”—frames in the training sequence where the object’s shape is known, corresponding to the template’s shape but with an unknown rigid transformation. We automatically determine this transformation using standard rigid tracking methods. With the template shape, we can compute the warp image  $\hat{\mathcal{W}}$  for that specific frame and use it for direct supervision.

The first step is to find point matches between the template texture map  $\mathcal{T}$  and the input image  $\mathcal{I}$ , corresponding to a keyframe. We use the state-of-the-art point matcher Light-Glue [51], an improved version of SuperGlue [52]. This algorithm extracts local features from both the input image and the texture map using SuperPoint [53] or DISK [54]. After that, it uses self and cross-attention layers to reject non-matchable points. The Sinkhorn algorithm is used to establish correspondences between the feature sets and perform the pair assignments. For planar templates, such as those used in the experimental results section, we compute a homographic transformation between the template texture map and the input image using a robust estimator based on RANSAC. The warp image is then obtained according to this transformation as follows:

$$\hat{\mathcal{W}}(u, v) = \begin{cases} \hat{u}_T(u, v), \hat{v}_T(u, v) & \mathcal{M}(u, v) = 1 \\ (-1, -1) & \mathcal{M}(u, v) = 0 \end{cases} \quad (11)$$

$$\hat{u}_T(u, v) = \frac{h_1 u + h_2 v + h_3}{h_7 u + h_8 v + h_9} \quad (12)$$

$$\hat{v}_T(u, v) = \frac{h_4 u + h_5 v + h_6}{h_7 u + h_8 v + h_9}, \quad (13)$$

where  $h_i$  for  $i = 1, \dots, 9$  are the coefficients of the homographic transformation between the keyframe and the template. For non-planar objects, a robust pose estimation method can be used to find the rigid transformation (*i.e.*, composed of a rotation matrix  $R$  and a translation vector  $T$ ) between the camera coordinates and the 3D template. The computation of  $\hat{\mathcal{W}}(u, v)$  becomes more complex than in



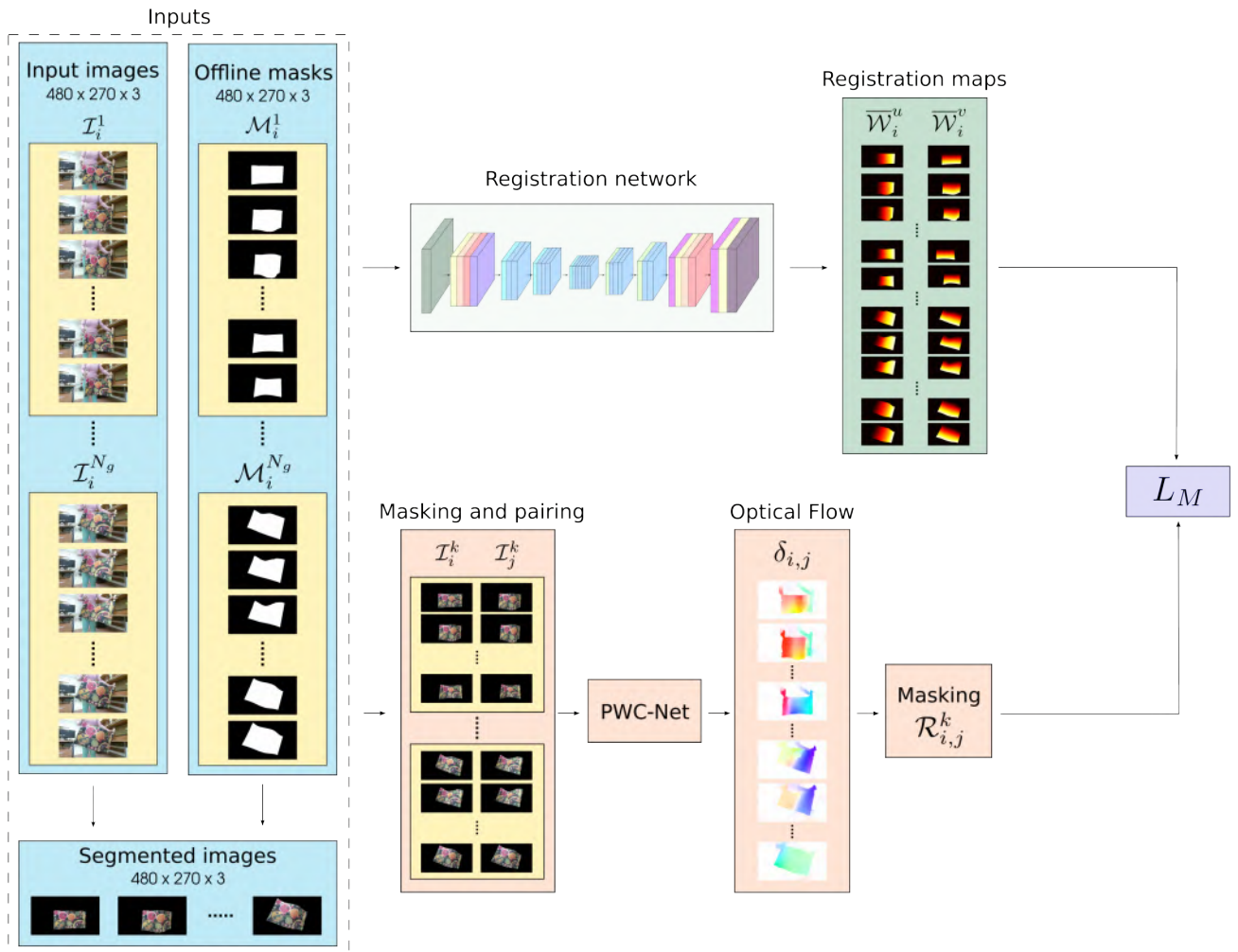


FIGURE 5: Proposed self-supervised matching loss  $L_M$ . This loss uses the spatiotemporal association between frames to match pixels corresponding to the same point in the template, enforcing their registration maps to coincide. Given a batch of  $N_g \times N_f$  input frames and its masks, we first compute the segmented images. Then, we conduct two tasks: 1) we use the input images and masks to obtain the registration maps  $\overline{\mathcal{W}}_i^u$  and  $\overline{\mathcal{W}}_i^v$  through the registration network (top branch), 2) we pair up consecutive segmented images in each group  $k$ , run them through the optical flow network PWC-Net and filter out invalid pairs of pixels (bottom branch). Once both tasks have been completed successfully, we compute the loss  $L_M$  using (9).

(11), as it requires using a rasterisation method to determine the barycentric coordinates of each image coordinate in the template, using the rigid transformation estimated earlier and the template's triangular mesh.

The rigidity loss is then computed as:

$$L_R = \frac{1}{N_k} \sum_{i=1}^K w(\|\overline{\mathcal{W}}_i - \hat{\mathcal{W}}_i\|, c_R), \quad (14)$$

where  $K$  is the set of keyframes in the batch,  $\hat{\mathcal{W}}_i$  is the  $i$ -th keyframe estimated warp using rigidity and  $\overline{\mathcal{W}}_i$  is the estimated warp using the registration DNN. We use  $w$  for the Welsch Loss [49] and  $c_R$  for its hyperparameter, as shown previously in (10).

#### 4) Smoothing Loss

Finally, we add an edge-aware smoothing term to reduce the high-frequency noise in the registration output while preserving the boundaries in the registration map  $\overline{\mathcal{W}}$ . For this purpose, we use the bilateral filtering loss from [55]. Given a batch of  $N_b$  images, where  $\mathcal{I}_{i,c}$  denotes the  $i$ -th image and  $c \in \{0, 1, 2\}$  represents its colour channel, and  $\overline{\mathcal{W}}_i$  corresponds to its estimated warp image, the smoothing loss is defined as:

$$L_S = \frac{1}{N_b} \sum \exp\left(-\frac{\sigma}{3} \sum_c \left|\frac{\partial \mathcal{I}_c}{\partial u}\right|\right) \left|\frac{\partial \overline{\mathcal{W}}}{\partial u}\right| + \exp\left(-\frac{\sigma}{3} \sum_c \left|\frac{\partial \mathcal{I}_c}{\partial v}\right|\right) \left|\frac{\partial \overline{\mathcal{W}}}{\partial v}\right|, \quad (15)$$

where  $\sigma$  modifies the edge weight in the smoothing term. Its value is determined in Section IV-B. The first-order partial derivatives of (15) are implemented using finite differences to make them differentiable with respect to the network parameters.

### C. IMPLEMENTATION DETAILS

We use the Stochastic Gradient Descent (SGD) optimiser to train our network with a learning rate of 0.0001. Additionally, we use a weighted sum given by (7) to balance the three different losses to avoid degeneration of the gradient due to ambiguities. We use the rigidity loss as a reference and assign it a weight of  $\lambda_R = 1$  and  $c_R = 0.01$  for the Welsch Loss hyperparameter. Then, we experimentally select a weight of  $\lambda_S = 25$  for the smoothing loss and a weight of  $\lambda_M = 1.25$  for the matching loss, with a value of  $\sigma = 150$  for the edge weight and a value of  $c_M = 0.005$  for the Welsch Loss hyperparameter respectively. Some of these values will be addressed forward in Section VI-F. The network was trained for 200 epochs although the matching and smoothing losses were not activated until epoch 20 for convergence purposes. We use a mini-batch of 9 images and divide it into  $N_g = 3$  groups of  $N_f = 3$  frames. Only the first group contains a keyframe ( $K = 1$ ) and each group belongs to a different sequence. In each iteration, the time step between frames is picked from a uniform distribution, with a minimum of 1 and a maximum that depends on the deformations present in the dataset. We choose a time step for each dataset that produces a meaningful mean displacement value of 5 pixels, which is sufficient to avoid large displacements while minimising noise influence. The registration network achieves an inference time of 0.058 seconds and a frame rate of 17.2625 fps on an Nvidia GTX 3090 GPU. Consequently, it is compatible with real-time applications.

## V. RECONSTRUCTION STEP

We recover the deformed surface's 3D vertices  $\mathcal{S}_V$ , from the template mesh  $\mathcal{T}$  and the warp image  $\overline{\mathcal{W}}$  obtained in the registration step. We propose a reconstruction method based on solving the following minimisation:

$$\mathcal{S}_V = \arg \min_{\mathcal{S}_V} (E(\mathcal{S}_V)) \quad (16)$$

$$E(\mathcal{S}_V) = E_d(\mathcal{S}_V, \mathcal{W}) + \lambda_a E_{iso}(\mathcal{S}_V) + \lambda_s E_s(\mathcal{S}_V), \quad (17)$$

where  $E_d$  is a reprojection data term that ensures the reconstruction is coherent with the estimated warp,  $E_{iso}$  enforces isometry, and  $E_s$  enforces surface smoothness. The hyperparameters  $\lambda_a$  and  $\lambda_s$  control the weights of the isometry and smoothness terms, respectively. All terms involve the known template mesh  $\mathcal{T}$ , which is composed of vertices  $\mathcal{T}_V$  and triangles  $\mathcal{T}_T$ .

### A. THE DATA TERM

The data term is obtained by sampling  $\overline{\mathcal{W}}$  to extract  $N_d$  point correspondences  $p_i \rightarrow q_i$ , where  $i = 1, \dots, N_d$ , ensuring

that they correspond to visible points on the surface using the mask, *i.e.*  $\mathcal{M}(p_i) = 1$ ,  $i = 1, \dots, N_d$ . We compute the barycentric coordinates  $\alpha_i, \beta_i, \gamma_i$  of each point  $q_i$  within the template mesh, identifying the mesh triangle that contains the point. Since  $\mathcal{T}$  and  $\mathcal{S}$  share the same topology, the barycentric coordinates of a point are preserved. Consequently,  $p_i$  in the image corresponds to a 3D point  $Q_i$  in  $\mathcal{S}$ , defined as  $Q_i = \alpha_i V_{1,i} + \beta_i V_{2,i} + \gamma_i V_{3,i}$ , where  $V_{k,i}$  are the vertices of the triangle in  $\mathcal{S}_V$  associated with point  $i$ . We propose the following reprojection data term:

$$E_d(\mathcal{S}_V) = \sum_{i=1}^{N_d} \frac{\|P_i \times K Q_i\|^2}{\|P_i\|^2}, \quad (18)$$

where  $P_i^\top = (p_i^\top, 1)$ ,  $K$  is the camera's intrinsic  $3 \times 3$  matrix, and  $\times$  denotes the cross product. The term  $E_d$  is quadratic in  $\mathcal{S}_V$ .

### B. THE ISOMETRIC TERM

We follow the approach of [56], [57] to construct the isometric energy term, which encourages each triangle in the mesh to transform quasi-rigidly. Given that  $\mathcal{T}_T$  has  $N_F$  triangles, we define the 3 vertices of face  $f$  as  $\mathcal{T}_{V,f}$  in the template, and  $\mathcal{S}_{V,f}$  in the deformed surface. The isometric energy term is defined as the following sum:

$$E_{iso}(\mathcal{S}_V) = \sum_{f=1}^{N_F} E_{euc}(\mathcal{S}_{V,f}, \mathcal{T}_{V,f}) \quad (19)$$

$$E_{euc}(X, Y) = \min_{R,t} \sum_{i=1}^3 \|R Y_i + t - X_i\|^2, \quad (20)$$

where  $E_{euc}(X, Y)$  involves computing the closest rotation  $R \in SO(3)$  and translation  $t \in \mathbb{R}^3$  that aligns the template triangle  $Y = (Y_1, Y_2, Y_3)$  to the deformed surface triangle  $X = (X_1, X_2, X_3)$ , with  $X_i, Y_i \in \mathbb{R}^3$  for  $i = 1, 2, 3$ , and then computing the alignment error. The term  $E_{iso}$  is non-convex due to the involvement of rotation matrices and the nested optimisation required for  $E_{euc}$ . To address this, we follow [56] to convert it into a convex quadratic energy by using a first-order approximation of the rotation term,  $R \approx (I + \text{skew}(r))R_0$ , where  $\text{skew}(r)$  is a  $3 \times 3$  skew-symmetric matrix,  $r \in \mathbb{R}^3$  is a rotation vector, and  $R_0 \in SO(3)$  is an approximation of  $R$ , computed as the best-fitting rotation between the template triangle  $Y$  and the previous available estimate of  $X$  (*i.e.*, the result from the last optimisation iteration). By replacing  $R$  with  $(I + \text{skew}(r))R_0$  in (19),  $E_{euc}(X, Y)$  admits a closed-form solution that is quadratic in  $X$ . Therefore, the energy term  $E_{iso}$  becomes quadratic in  $\mathcal{S}_V$ . This approximation requires that  $R_0$  is computed accurately and updated at each iteration, as described in Section V-D.

### C. THE SMOOTHING TERM

We use the thin-shell bending energy term to define  $E_s(\mathcal{S}_V)$ , which is quadratic in  $\mathcal{S}_V$  and is designed to penalise high changes in the template curvature. We use the discrete form

described in [58]. This method works by taking the four vertices from every pair of neighbouring triangles and measuring the degree to which their motion deviates from affine motion. The result is a bending matrix,  $B$ , that is used to compute  $E_s$ :

$$E_s(\mathcal{S}_V) = \|B \text{vec}(\mathcal{S}_V)\|_2^2, \quad (21)$$

where  $\text{vec}(\mathcal{S}_V)$  is a column vector containing the coordinates of all vertices in  $\mathcal{S}_V$ .

#### D. OPTIMISATION ALGORITHM

To optimise (16), we first reduce the dimensionality of the solution by expressing the set of vertices  $\text{vec}(\mathcal{S}_V)$  as a linear basis  $\text{vec}(\mathcal{S}_V) = A_b \Phi$ , where  $A_b$  is obtained as the  $n_b$  last singular vectors of the bending matrix  $B$  used in (21), and  $\Phi \in \mathbb{R}^{n_b}$  is the unknown parameter vector. We transform (16) to use  $\Phi$  as the unknown parameter:

$$\Phi = \arg \min_{\Phi} (E(A_b \Phi)). \quad (22)$$

We initialise the solution by projecting the template surface onto the basis  $A_b$ :

$$\Phi_0 = A_b^\top \text{vec}(\mathcal{T}_V) \quad (23)$$

At each iteration  $k$ , the algorithm computes  $\Phi_k$  by minimising (22) which is quadratic and convex on  $\Phi_k$ . We then compute the set of mesh vertices by computing  $\text{vec}(\mathcal{S}_V)_k = A_b \Phi_k$ . Importantly, we use the solution to the last iteration  $\text{vec}(\mathcal{S}_V)_k$  to linearise the isometric energy term. We stop the optimisation after a few iterations or if  $\|\Phi_k - \Phi_{k-1}\| < \epsilon$ . In practice, we set  $\lambda_s$  to a very small value, since  $n_b$  also controls the smoothness of the solution, leaving  $\lambda_{iso}$ ,  $n_b$  and the number of iterations as the main hyperparameters of the reconstruction step. Further implementation details are provided in Section V-E. Once  $\mathcal{S}_V$  is estimated, we can easily obtain the associated depth map by using a z-buffer rasterization method. This is used in the experiments to compare our reconstruction results with other methods, such as DeepSfT, that recover the depth map.

#### E. IMPLEMENTATION DETAILS

The template meshes in our dataset are composed of  $N_V = 12 \times 12$  equally spaced vertices, which form a total of  $N_F = 242$  triangular faces. In our experiments, we sample  $N_d = 10^4$  image positions, which is sufficient given the complexity of these templates. We find that  $n_b \approx 250$  bases and  $\lambda_s = 10^{-5}$  represent large deformations accurately, keeping a smooth surface and significantly reducing the computation time needed. We set  $\lambda_{iso} = 10$ , achieving a correct balance between the isometric and reprojection constraints. Finally, we use  $\epsilon = 10^{-5}$  and a maximum of  $N_{iters} = 25$  iterations.

## VI. EXPERIMENTAL RESULTS

### A. DATASETS

Due to the dataset requirements explained in Section III-B, none of the existing publicly available SfT datasets are suitable for training our network. Therefore, we have created

five new datasets with thin-shell templates to evaluate our method and compare its results with the state-of-the-art. The datasets have been generated from three example texture map images: 1) “Fruits”, an image with bright colours and texture variety (Figure 6), 2) “Zaius”, exhibiting darker colours and poorer texture (Figure 7), and 3) “Cloth”, with a simpler texture but fine details (Figure 8). For texture maps 1) and 2), we generated a synthetic dataset using a bending paper simulation and record a real dataset by printing the texture map in a piece of paper. For 3), we used a cloth bag with a printed texture. Since the deformations of this material differ from those observed in paper, we did not create a synthetic dataset, as it would not faithfully represent its real behaviour. The texture map resolution and shape information for each template are shown in Table 2.



FIGURE 6: “Fruits” dataset texture map.



FIGURE 7: “Zaius” dataset texture map.



FIGURE 8: “Cloth” dataset texture map.

Each dataset consists of 70 video sequences, with 300 frames each, making 21000 images in total. The first frame of each sequence is a keyframe, showing the template shape and is then deformed gradually through the rest of the sequence. Each dataset has been split into three parts: the first 50 se-

		Texture map (px)		Template (mm)	
		Width	Height	Width	Height
Synthetic	Zaius	480	270	480	270
	Fruits	480	270	480	270
Real	Zaius	689	384	288	159
	Fruits	2186	1217	590.81	328.92
	Cloth	2237	2222	247.5	246.0

TABLE 2: Templates' dimensions.

quences are used to train the network, the next 13 sequences are used to determine the best hyperparameterisation of the network, and the remaining sequences are used to compare the proposed solution with state-of-the-art methods. The selection of hyperparameters will be further discussed in Section VI-F.

### 1) Synthetic Datasets

We have created a synthetic dataset for each planar template described in Table 2. Although our weakly-supervised methodology is able to work directly with real scenes, synthetic data allow us to generate registration and reconstruction ground truth, not available in real data. The datasets were generated by digitally simulating quasi-isometric deformations of a piece of paper folding around the surface of a cylinder of variable radius. The paper's position and orientation relative to the cylinder surface varies slightly between frames, as well as the radius of the cylinder, to generate different deformations. The rendering process mimics the Kinect 2 camera intrinsic parameters, along with an initial random camera pose and random camera displacements throughout the sequence. The resolution of the generated images is  $480 \times 270$  pixels. To avoid frames where the object is out of sight, the relative motion between the camera and the surface is limited, so at least three-quarters of the template is always in the line of sight of the camera. In addition, we constrain the minimum radius of the cylinder to ensure that the back of the texture map remains unseen in all frames and the paper does not bend into itself. The initial values and the possible range of the parameters used to generate the dataset are shown in Table 3. We obtain registration and reconstruction ground truth to quantitatively compare different methods and optimise the model hyperparameters. During training with synthetic scenes, we use the registration ground truth for the keyframes, instead of the pose estimation process required in real scenes. This allows us to better evaluate the performance of the matching loss in our ablation studies.

### 2) Real Datasets

We have recorded three real datasets based on the "Fruits", "Zaius" and "Cloth" templates, using a Microsoft Kinect v2 which provides RGB and depth data at 30 frames per second (fps). The depth maps are aligned with the RGB images that have been resized to  $480 \times 270$  pixels to fit the requirements of our registration DNN. The depth data

gathered is only used to evaluate reconstruction accuracy and is not used during training, and the registration labels are not provided. To create real thin-shell templates for paper-like datasets, we print the texture maps with sizes A2 for "Fruits" and A4 for "Zaius", cut them to maintain the texture map proportions, and attach two handles to both sides of the paper to perform manual deformations in the template without producing occlusions. For "Cloth", we use a squared cloth bag with the texture map printed in its center and hold it with a hanger to perform manual deformations. As explained in Section IV-B3, we need to generate an estimate of the ground-truth data for the keyframes in order to implement the rigidity loss and avoid the uniform solution of the matching loss. To obtain a registration map of rigid frames, we used a state-of-the-art pose estimation method, as explained in Section IV-B3.

### 3) Data Augmentation

We perform data augmentation to increase the generalisation of the network during the training phase. We perform affine transformations of the image and add noise, so the intrinsic characteristics of the template (such as colour and structure) are not modified. We apply the data augmentation with random transformation parameters among certain limits to avoid invalid frames. First, the input image is shifted by a maximum of 0.2 times the size of the input image and rotated by a maximum of  $60^\circ$ . There is an exception with the real "Zaius" dataset which is not rotated at all due to the template falling out of the images in some frames. After that, we apply blur with a probability of 0.33, which can be motion blur (with a kernel of size 3) or Gaussian blur ( $\mu = 0$ ,  $\sigma = [0.0001, 0.0004]$ ), with equal likelihood.

## B. COMPARED METHODS

We compare the proposed technique with three state-of-the-art methods that are compatible with the generated datasets. They are described next. We add a clipping stage to the output of these methods to avoid registration values of valid regions to point at positions outside of the template.

### 1) LightGlue + BS

The first method estimates the registration between the template and the input image by applying a sparse point matching algorithm. We use the same state-of-the-art method as in the rigidity loss, LightGlue [51], to extract point matches and remove mismatches as explained in Section IV-B3. Then, we apply a Bicubic B-spline (BBS) algorithm with the ground-truth mask as the region of interest to densify the sparse 2D correspondences according to the properties of the template and obtain a dense registration map. Finally, we utilise the input mask to remove invalid points that might fall outside of the template due to errors in the estimation process. Once the registration maps have been obtained, we apply a 2D to 3D reconstruction method to recover the 3D deformed shape. To make a fair comparison, we apply ARAP, the same algorithm as in our method that was explained earlier in Section V.



Parameter	Initial value	Range of values	Standard deviation	Maximum Variation
Rotation around axis X (rad)	0* / last sequence	[-0.5, 0.5)	0.03	-
Rotation around axis Y (rad)	0* / last sequence	[-0.5, 0.5)	0.03	-
Rotation around axis Z (rad)	0* / last sequence	[-0.5, 0.5)	0.03	-
Displacement in axis X (mm)	[-120, 120)	[-120, 120)	20	[-48, 48)
Displacement in axis Y (mm)	[-67.5, 67.5)	[-67.5, 67.5)	10	[-27, 27)
Displacement in axis Z (mm)	[486, 491)	[432, 648)	10	[-100, 100)
Radius of the cylinder (mm)	$\infty$	[216, $\infty$ )	260	-
Rotation angle of template to axis Z (rad)	0*	$[-\pi/2, \pi/2)$	0.1	-
Number of control mesh points	8	-	-	-
Sequence length (frames)	300	-	-	-

TABLE 3: Generation parameters for the synthetic dataset. \*: Rotations start at 0 in the first sequence of the dataset, but do not reset afterwards. The rest of the parameters reset to a value inside the initial value range at the beginning of the sequence.

## 2) RAFT

We also compare our registration DNN with RAFT [59], a state-of-the-art DNN optical flow method. RAFT estimates the displacement of the pixels between the template and the image, which is equivalent to performing dense point matching between both images and thus, can be used to estimate the warp image. However, optical flow algorithms do not work properly in a wide-baseline scenario. Since there is a substantial disparity in the object size between the input image and the template, we need to add a pre-processing step that adjusts the template's size or otherwise the RAFT DNN is unable to find the optical flow between the images. Thus, we add padding to the texture map, so the template in the reference image has a similar occupancy ratio as the object in the target images. Due to this fact, two post-processing steps are needed: one to turn the optical flow (relative coordinates) into a registration map (absolute coordinates), and a second one to transform the coordinates from the padded reference template into the original one. We have chosen a padding value of -1 to automatically label invalid optical flow predictions that lead to points outside of the template in the padded reference. After that, the ground-truth mask is applied to remove artefacts generated by the optical flow during the registration process. Finally, ARAP is applied to obtain the 3D reconstruction from the registration maps.

## 3) DeepSfT and DeepSfT+ARAP

Lastly, we compare our method with DeepSfT [24]. In our experiments, the DeepSfT DNN has been trained in the synthetic dataset of each corresponding template, using both registration and depth ground-truth supervision. This method could not be tested on the "Cloth" dataset due to the lack of a synthetic dataset required for training. In the original implementation, DeepSfT also includes a fine-tuning step to adapt it to real data that requires depth data. We do not add this step to DeepSfT since it would be unfair to compare it with our method which does not require a depth sensor. We test the performance of DeepSfT, both registration and

depth outputs, and DeepSfT+ARAP, where we apply ARAP reconstruction to the registration solution, discarding the depth output.

## C. EVALUATION METRICS

First, we evaluate the performance of all methods using synthetic data, where we have registration and reconstruction labels. We use the following metrics:

- The **Root Mean Squared Error (RMSE)** of the registration and depth map:

$$E_{\mathcal{W}} = \sqrt{\gamma \sum_{(u,v) \in \mathcal{R}} \|\overline{\mathcal{W}}(u,v) - \mathcal{W}(u,v)\|^2} \text{ (mm)}, \quad (24)$$

$$E_{\mathcal{D}} = \sqrt{\gamma \sum_{(u,v) \in \mathcal{R}} \|\overline{\mathcal{D}}(u,v) - \mathcal{D}(u,v)\|^2} \text{ (mm)}, \quad (25)$$

where  $\mathcal{W}$ ,  $\mathcal{D}$  are the ground-truth maps and  $\overline{\mathcal{W}}$ ,  $\overline{\mathcal{D}}$  are the estimated maps. The region  $\mathcal{R}$  contains the pixel positions that lie in the intersection of the estimated mask and the ground-truth mask. Factor  $\gamma$  is the normalization factor equal to the inverse of the number of evaluated pixel positions.

- The **3D error** ( $d_{3D}$ ) is the distance between the estimated 3D position and the closest point in the ground truth:

$$d_{3D} = \sqrt{\gamma \sum_{(u,v) \in \mathcal{R}} \|\overline{\mathcal{X}}(u,v) - \mathcal{X}(u_{nb}, v_{nb})\|_2^2} \text{ (mm)}, \quad (26)$$

where  $\mathcal{X}(u_{nb}, v_{nb})$  is the nearest neighbour point in the ground-truth image embedding to the estimation  $\overline{\mathcal{X}}$ , as described in Section III-A.

- The **Intersection over Union (IoU)**, measures the discrepancies between the detection area of the prediction and the ground-truth mask. The detection area encapsulates points where the evaluated method has produced a valid registration prediction. In the case of DeepSfT, proper depth prediction must be considered as part of this area. When the estimated IoU is lower than

70%, we consider that the system cannot recover the registration and reconstruction maps properly, and we classify the prediction as a **catastrophic failure**. Those cases have been removed from the metric estimations and are counted separately. The IoU threshold has been experimentally established to be the value at which ARAP performance drops due to the lack of accurate registration.

- The **Photometric Error** ( $E_{Ph}$ ) is evaluated as the mean L2 error between the intensity functions of the input image  $\mathcal{I}$  and the synthesised input image  $\bar{\mathcal{I}}$ , which is obtained by warping the template's texture map into the input image using the estimated warp, as explained in [24]. This representation of the network prediction allows us to directly compare the output of the network with the inputs. However, this measurement is not as accurate as the registration error since it may contain noise introduced by illumination changes between the template and the input image:

$$E_{Ph} = \sqrt{\gamma \sum_{(u,v) \in \mathcal{R}} \|\bar{\mathcal{I}}(u,v) - \mathcal{I}(u,v)\|_2^2} \text{ (n.u.)} \quad (27)$$

In addition, a few graphical examples of the method's performance in medium to difficult frames are shown in Tables 6, 7, 8 and 9. The visual depictions include the synthesised image and the registration, depth and 3D error maps estimated as the absolute difference between the ground truth and the prediction. We also compare the projection of 3D meshes into the camera coordinates to obtain a clearer picture of the reconstruction of the shape, and the synthesised image with the input image by overlaying the former over the latter by applying (28):

$$\mathcal{I}_{overlay} = \frac{\mathcal{I} + \bar{\mathcal{I}}}{2} \quad (28)$$

For the real datasets, we keep the depth and photometric errors and the IoU-related metrics. In this case, we remove points from the ground truth whose depth is lower than 10 mm, as they come from inaccuracies of the depth sensor and would interfere with the evaluation process. We also add the following metrics between  $\mathcal{I}$  and the synthesised image  $\bar{\mathcal{I}}$  to further evaluate the results:

- **Peak Signal-to-Noise Ratio (PSNR)** (dB) is a measurement commonly used in computer vision to quantify the discrepancy of a distorted image with respect to the original image.
- **Structural Similarity Index Measure (SSIM)** estimates the perceived variations in structural information, luminance and contrast between a target image and a reference image considered of perfect quality.

We also include some graphical examples of the results in medium to difficult frames for the real datasets in Tables 12, 13, 14, 15, 16 and 17. We use the same representations of the methods' performance as in the synthetic case, with the exception of the registration error that is replaced by the photometric error since its ground truth is not available.

#### D. EVALUATION WITH SYNTHETIC TEMPLATES

In this section, we analyse the performance metrics and qualitative results achieved by the different methods in the synthetic datasets. The numerical results obtained by each method are presented in Table 5, while a few graphical examples of the method's performance in medium to difficult frames are shown in Tables 6, 7, 8 and 9. The input images corresponding to the frames used for this qualitative evaluation are presented in Table 4.

As shown in Table 5, our model WS-DeepSfT outperforms its competitors in both datasets and across all metrics, achieving robust predictions even in challenging conditions. LightGlue+BS achieves the second-best ranking in registration and photometric error in the "Fruits" dataset by a close margin. Nonetheless, its performance drops significantly in "Zaius" due to the lack of features in the texture, where it is overtaken by RAFT, which shows robustness on texture-poor surfaces. However, RAFT's weaknesses, similar to other optical flow methods, become apparent in the presence of large displacements, as in the "Fruits" dataset, where it ranks as the third-best method. Finally, DeepSfT is the lowest-performing method in "Fruits" and second-to-last in "Zaius". This behaviour is attributed to the dataset's variability and size, as DeepSfT heavily relies on the availability of a large amount of training data with a wide range of deformations and lighting conditions. Although WS-DeepSfT uses a similar DNN, our training strategy successfully overcomes these drawbacks by reducing the amount of required training data and minimising outliers and noise, especially along the template boundaries, as shown in Tables 6, 7, 8 and 9.

The results obtained in registration are directly related to the depth and 3D errors since ARAP uses the former combined with the template information predict the latter, as we observe in Table 5. The only exception is DeepSfT, which predicts its own depth map without ARAP and thus reduces its dependency on registration accuracy. Nevertheless, WS-DeepSfT still outperforms DeepSfT in 3D metrics since the registration errors are lower. On the other hand, LightGlue+BS and DeepSfT+ARAP fail to accomplish an accurate reconstruction of the 3D mesh through ARAP due to the presence of large errors in the registration maps, which results in high depth and 3D errors. It must be noted that DeepSfT also produces its own segmentation instead of using an input mask. This feature leads to an increase of errors in the boundaries of the template and a higher number of gross errors, which results in larger overall errors in registration and reconstruction. This behaviour is observed in the registration error maps of Tables 6, 7, 8 and 9, where the predictions of this method achieve much lower errors in the middle of the template compared to the boundaries.

Aside from blunders, most of these methods fail to make predictions for some pixels inside the region of interest defined by the input mask, as seen in the boundaries of DeepSfT predictions, in the predictions of LightGlue+BS in Tables 7, and 9 or in the left side of the RAFT prediction in 9. WS-DeepSfT consistently produces better and more stable

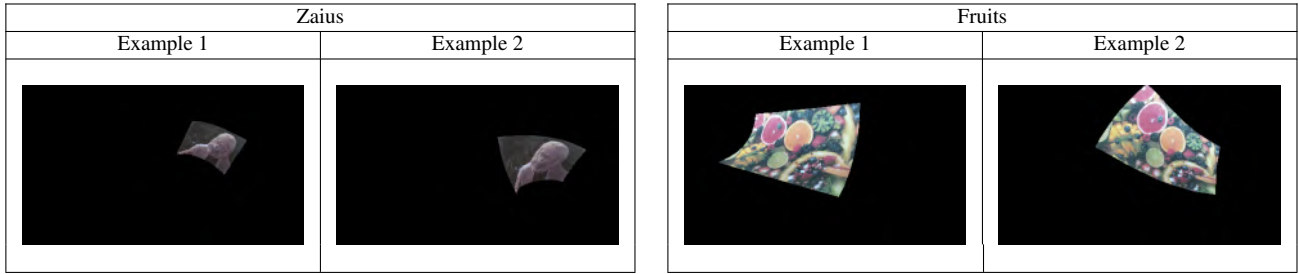


TABLE 4: Example frames for the qualitative results in synthetic datasets.

Zaius						
	$E_{\mathcal{W}}$ (mm)	$E_{Ph}$ (n.u.)	$E_{\mathcal{D}}$ (mm)	$d_{3D}$ (mm)	IoU	Failures
LightGlue+BS	76.0006	0.1905	850.2050	779.8587	0.9167	284
RAFT	17.1158	0.0957	60.9973	45.8062	0.9824	<b>0</b>
DeepSfT	54.6259	0.1205	58.1608	35.8589	0.9591	<b>0</b>
DeepSfT+ARAP	54.6260	0.1205	766.3440	667.1372	0.9515	<b>0</b>
WS-DeepSfT	<b>7.9275</b>	<b>0.0907</b>	<b>26.8419</b>	<b>20.2382</b>	<b>1.0000</b>	<b>0</b>

Fruits						
	$E_{\mathcal{W}}$ (mm)	$E_{Ph}$ (n.u.)	$E_{\mathcal{D}}$ (mm)	$d_{3D}$ (mm)	IoU	Failures
LightGlue+BS	3.7701	0.1960	48.5261	42.4192	0.9817	<b>0</b>
RAFT	15.9341	0.1896	19.1145	12.8020	0.9287	104
DeepSfT	29.5810	0.2018	20.1214	11.7167	0.9636	191
DeepSfT+ARAP	51.2252	0.2097	117.5677	84.9531	0.9552	33
WS-DeepSfT	<b>3.3347</b>	<b>0.1721</b>	<b>6.0404</b>	<b>4.3278</b>	<b>1.0000</b>	<b>0</b>

TABLE 5: Quantitative evaluation results on synthetic test data on the “Zaius” and “Fruits” datasets.

Model	$I'$	$I_{over}$	Registration Error (mm)	Depth Error (mm)	3D Error (mm)	3D Mesh Comparison
LightGlue+BS			 $E_{\mathcal{W}} = 48.85$	 $E_{\mathcal{D}} = 728.14$	 $d_{3D} = 621.13$	
RAFT			 $E_{\mathcal{W}} = 17.09$	 $E_{\mathcal{D}} = 94.10$	 $d_{3D} = 68.18$	
DeepSfT			 $E_{\mathcal{W}} = 56.21$	 $E_{\mathcal{D}} = 96.57$	 $d_{3D} = 42.69$	
DeepSfT+ARAP			 $E_{\mathcal{W}} = 56.21$	 $E_{\mathcal{D}} = 872.94$	 $d_{3D} = 778.27$	
WS-DeepSfT			 $E_{\mathcal{W}} = 8.80$	 $E_{\mathcal{D}} = 73.20$	 $d_{3D} = 49.60$	

TABLE 6: Qualitative results in example 1 of the “Zaius” synthetic dataset are shown. Lighter pixels correspond to higher errors. The colour representation saturates at 30 mm for the registration error and at 250 mm for the depth and 3D distance maps. In the 3D Mesh Comparison column, the ground-truth surface is coloured blue, and the predicted mesh is coloured red.

Model	$I'$	$I_{over}$	Registration Error (mm)	Depth Error (mm)	3D Error (mm)	3D Mesh Comparison
LightGlue+BS			 $E_{\mathcal{W}} = 214.23$	 $E_{\mathcal{D}} = 823.17$	 $d_{3D} = 697.64$	
RAFT			 $E_{\mathcal{W}} = 10.89$	 $E_{\mathcal{D}} = 16.52$	 $d_{3D} = 10.82$	
DeepSfT			 $E_{\mathcal{W}} = 41.90$	 $E_{\mathcal{D}} = 55.09$	 $d_{3D} = 35.57$	
DeepSfT+ARAP			 $E_{\mathcal{W}} = 41.90$	 $E_{\mathcal{D}} = 545.88$	 $d_{3D} = 419.03$	
WS-DeepSfT			 $E_{\mathcal{W}} = 11.90$	 $E_{\mathcal{D}} = 31.16$	 $d_{3D} = 20.45$	

TABLE 7: Qualitative results in example 2 of the “Zaius” synthetic dataset are shown. Lighter pixels correspond to higher errors. The colour representation saturates at 30 mm for the registration error and at 250 mm for the depth and 3D distance maps. In the 3D Mesh Comparison column, the ground-truth surface is coloured blue, and the predicted mesh is coloured red.

Model	$I'$	$I_{over}$	Registration Error (mm)	Depth Error (mm)	3D Error (mm)	3D Mesh Comparison
LightGlue+BS			 $E_{\mathcal{W}} = 3.55$	 $E_{\mathcal{D}} = 2.67$	 $d_{3D} = 2.07$	
RAFT			 $E_{\mathcal{W}} = 15.37$	 $E_{\mathcal{D}} = 15.79$	 $d_{3D} = 10.55$	
DeepSfT			 $E_{\mathcal{W}} = 37.45$	 $E_{\mathcal{D}} = 32.28$	 $d_{3D} = 14.25$	
DeepSfT+ARAP			 $E_{\mathcal{W}} = 37.45$	 $E_{\mathcal{D}} = 177.37$	 $d_{3D} = 83.36$	
WS-DeepSfT			 $E_{\mathcal{W}} = 6.44$	 $E_{\mathcal{D}} = 12.03$	 $d_{3D} = 7.93$	

TABLE 8: Qualitative results in example 1 of the “Fruits” synthetic dataset are shown. Lighter pixels correspond to higher errors. The colour representation saturates at 30 mm for the registration error and at 250 mm for the depth and 3D distance maps. In the 3D Mesh Comparison column, the ground-truth surface is coloured blue, and the predicted mesh is coloured red.





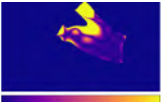





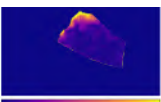


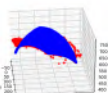

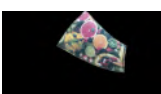
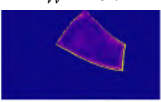

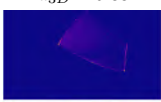
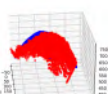


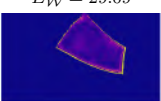

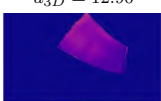



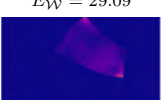
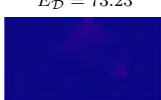
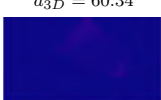
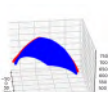
Model	$I'$	$I_{over}$	Registration Error (mm)	Depth Error (mm)	3D Error (mm)	3D Mesh Comparison
LightGlue+BS			 $E_{\mathcal{W}} = 72.31$	 $E_{\mathcal{D}} = 358.28$	 $d_{3D} = 292.00$	
RAFT			 $E_{\mathcal{W}} = 15.72$	 $E_{\mathcal{D}} = 13.25$	 $d_{3D} = 9.38$	
DeepSfT			 $E_{\mathcal{W}} = 29.09$	 $E_{\mathcal{D}} = 22.11$	 $d_{3D} = 12.90$	
DeepSfT+ARAP			 $E_{\mathcal{W}} = 29.09$	 $E_{\mathcal{D}} = 73.23$	 $d_{3D} = 60.34$	
WS-DeepSfT			 $E_{\mathcal{W}} = 3.57$	 $E_{\mathcal{D}} = 6.41$	 $d_{3D} = 4.50$	

TABLE 9: Qualitative results in example 2 of the “Fruits” synthetic dataset are shown. Lighter pixels correspond to higher errors. The colour representation saturates at 30 mm for the registration error and at 250 mm for the depth and 3D distance maps. In the 3D Mesh Comparison column, the ground-truth surface is coloured blue, and the predicted mesh is coloured red.

predictions, even in the most challenging areas or frames where other methods fail. This fact has a high impact on the metrics since invalid pixels and frames are not computed in the mean of the error, therefore boosting the metrics of the other methods. Despite this, our method outperforms the other methods and achieves lower registration and reconstruction errors while avoiding catastrophic failures even in the most demanding circumstances.

Thus, our system is robust against the complexity of the database whereas other methods fail in challenging textures (LightGlue+BS) or large shapes (RAFT). Furthermore, using a separate network to compute the segmentation allows us to reduce the number of blunders on the boundary of the template, which constitutes a great hindrance in methods such as DeepSfT. The robustness of the network enables ARAP to obtain a more accurate 3D reconstruction and more visually appealing synthesised results. These reasons validate our proposal as it improves the actual state of the art in SfT for synthetic templates.

### E. EVALUATION WITH REAL OBJECTS

Real results follow a similar pattern to the synthetic scenario, as we observe in Tables 11, 12, 13, 14, 15, 16 and 17. In this case, the frames used for the qualitative evaluation are presented in Table 10. The results are consistent with the expected outcomes given the performance shown in synthetic data. It is noteworthy that the render gap between synthetic

and real data has a large impact on the results as there is a significant increase in the error and the number of failed frames for most methods in the “Zaius” and “Fruits” datasets, although it is especially remarkable on the former.

Comparing the synthetic and real results, we observe that WS-DeepSfT outmatches the other models in the “Fruits” and “Cloth” datasets and performs extremely well in the “Zaius” dataset, either as the best or second best method, depending on the scope of the evaluation metric. It excels at metrics that evaluate the entire image, such as PSNR or SSIM, while also achieving excellent results in the others. This is because the other methods fail to make valid estimations of the registration map in challenging pixels, leading to their removal in the calculation of per-pixel metrics such as  $E_{\mathcal{D}}$  or  $E_{Ph}$  and increasing their performance in these metrics. Consequently, our approach yields higher perceived quality in the synthesised images generated from registration estimations. Moreover, this property makes our solution the only method that avoids catastrophic failures across all datasets.

RAFT achieves similar results as our method in the “Zaius” dataset but its performance drops in the “Fruits” and “Cloth” datasets due to large displacements, as it happens in the synthetic “Fruits” dataset. LightGlue+BS also delivers similar results in the “Fruits” dataset given its rich texture, but is not able to obtain accurate results on registration or reconstruction metrics in the “Zaius” dataset due to failures

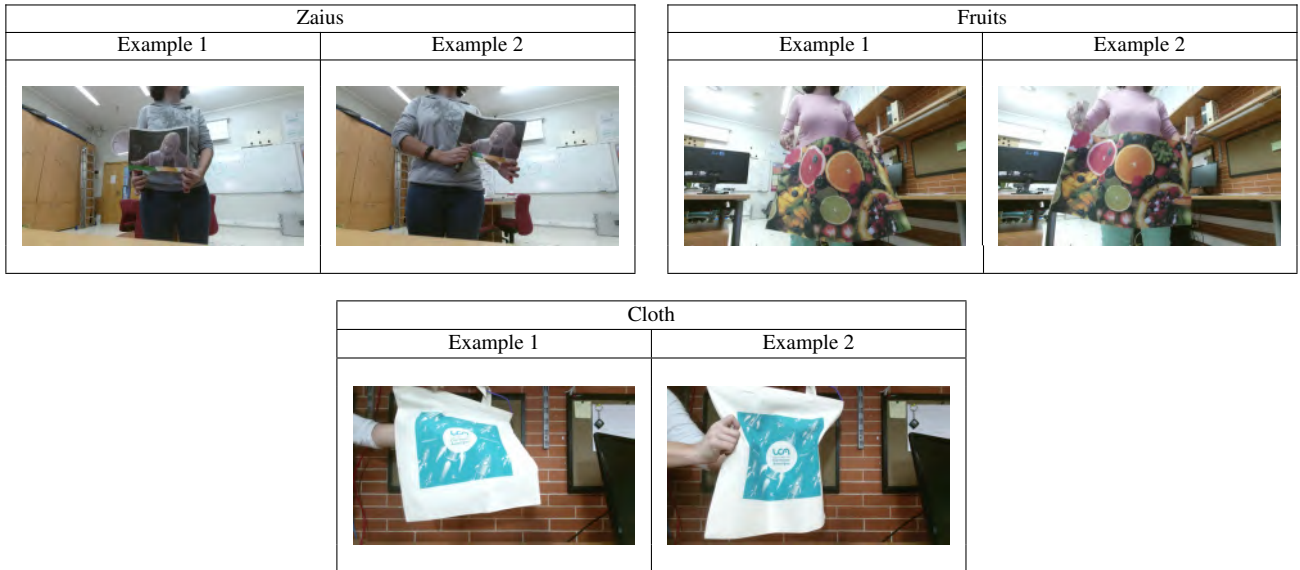


TABLE 10: Example frames for the qualitative results in real datasets.

Zaius						
	$E_D$ (mm)	$E_{Ph}$ (n.u.)	PSNR (dB)	SSIM	IoU	Failures
LightGlue+BS	537.3366	0.1500	29.0467	0.9600	0.9330	45
RAFT	<b>61.7937</b>	<b>0.1197</b>	31.1017	0.9753	0.9748	140
DeepSfT	373.4796	0.1362	29.2424	0.9571	0.8689	255
DeepSfT+ARAP	266.9596	0.1362	29.2424	0.9571	0.8690	255
WS-DeepSfT	69.9365	0.1332	<b>31.8683</b>	<b>0.9773</b>	<b>1.0000</b>	<b>0</b>

Fruits						
	$E_D$ (mm)	$E_{Ph}$ (n.u.)	PSNR (dB)	SSIM	IoU	Failures
LightGlue+BS	15.3206	0.2569	20.6800	0.8710	0.9678	<b>0</b>
RAFT	36.0014	<b>0.2486</b>	20.0392	0.8764	0.9325	1
DeepSfT	189.2932	0.2990	18.6272	0.8120	0.8947	721
DeepSfT+ARAP	178.8990	0.2990	18.6273	0.8120	0.8947	721
WS-DeepSfT	<b>14.6969</b>	0.2557	<b>21.1394</b>	<b>0.8874</b>	<b>1.0000</b>	<b>0</b>

Cloth						
	$E_D$ (mm)	$E_{Ph}$ (n.u.)	PSNR (dB)	SSIM	IoU	Failures
LightGlue+BS	51.4537	0.3206	20.6393	0.9001	0.9566	144
RAFT	35.4192	<b>0.3000</b>	21.6649	0.9253	0.9784	<b>0</b>
WS-DeepSfT	<b>17.4831</b>	0.3116	<b>22.2931</b>	<b>0.9255</b>	<b>1.0000</b>	<b>0</b>

TABLE 11: Quantitative evaluation results on real test data on the “Zaius”, “Fruits” and “Cloth” datasets.

in the feature matching algorithm. It also achieves mediocre results in the “Cloth” dataset due to its lack of texture in larger areas. Finally, DeepSfT and DeepSfT+ARAP are unable to overcome the render gap between synthetic and real in the “Zaius” and “Fruits” datasets. These methods rely on training on a very large (over 100K images) and complex synthetic dataset with varying lighting and deformations to learn the task. Since the synthetic datasets used for the training are more simplistic, these methods fail to achieve comparable results to the other methods. Furthermore, these

methods could not be trained in the “Cloth” dataset since the synthetic data generator could not reproduce the complexity of deformations in this new material.

These experiments also assess the validation of ARAP as a separate reconstruction method in real data to remove the necessity of using a depth sensor for supervised training. As observed in Tables 11, 12, 13, 14, 15, 16 and 17 this method is able to properly reconstruct the template, even in difficult frames within a reasonable error.

Model	$I'$	$I_{over}$	Photometric Error (n.u.)	Depth Error (mm)	3D Error (mm)	3D Mesh Comparison
LightGlue+BS			 $E_{Ph} = 0.12$	 $E_D = 93.00$	 $d_{3D} = 49.19$	
RAFT			 $E_{Ph} = 0.09$	 $E_D = 186.42$	 $d_{3D} = 102.68$	
DeepSfT			 $E_{Ph} = 0.12$	 $E_D = 413.19$	 $d_{3D} = 376.65$	
DeepSfT+ARAP			 $E_{Ph} = 0.12$	 $E_D = 339.31$	 $d_{3D} = 242.54$	
WS-DeepSfT			 $E_{Ph} = 0.10$	 $E_D = 99.61$	 $d_{3D} = 63.71$	

TABLE 12: Qualitative results in example 1 of the “Zaius” real dataset are shown. Lighter pixels correspond to higher errors. The colour representation saturates at 0.5 for the photometric error and at 250 mm for the depth and 3D distance maps. In the 3D Mesh Comparison column, the ground-truth surface is coloured blue, and the predicted mesh is coloured red.

Model	$I'$	$I_{over}$	Photometric Error (n.u.)	Depth Error (mm)	3D Error (mm)	3D Mesh Comparison
LightGlue+BS			 $E_{Ph} = 0.13$	 $E_D = 50.29$	 $d_{3D} = 32.27$	
RAFT			 $E_{Ph} = 0.10$	 $E_D = 51.15$	 $d_{3D} = 33.02$	
DeepSfT			 $E_{Ph} = 0.14$	 $E_D = 397.23$	 $d_{3D} = 374.40$	
DeepSfT+ARAP			 $E_{Ph} = 0.14$	 $E_D = 191.07$	 $d_{3D} = 154.25$	
WS-DeepSfT			 $E_{Ph} = 0.12$	 $E_D = 58.35$	 $d_{3D} = 38.33$	

TABLE 13: Qualitative results in example 2 of the “Zaius” real dataset are shown. Lighter pixels correspond to higher errors. The colour representation saturates at 0.5 for the photometric error and at 250 mm for the depth and 3D distance maps. In the 3D Mesh Comparison column, the ground-truth surface is coloured blue, and the predicted mesh is coloured red.

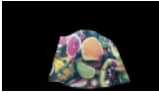

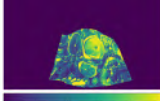


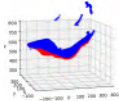

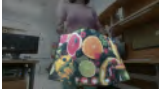
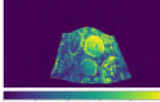
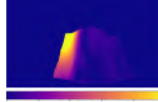
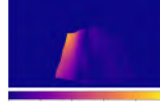
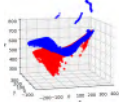

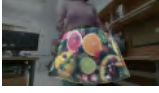
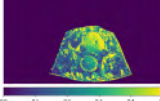
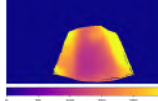
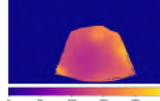
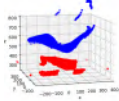

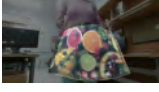
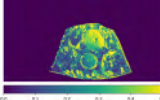
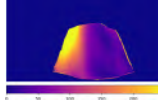
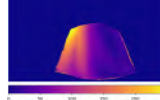
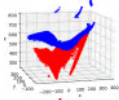

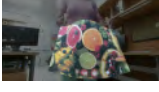
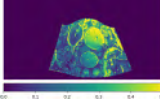
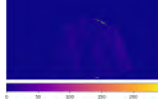
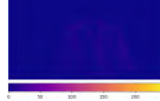
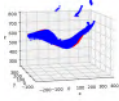
Model	$I'$	$I_{over}$	Photometric Error (n.u.)	Depth Error (mm)	3D Error (mm)	3D Mesh Comparison
LightGlue+BS			 $E_{Ph} = 0.27$	 $E_D = 29.31$	 $d_{3D} = 18.59$	
RAFT			 $E_{Ph} = 0.24$	 $E_D = 88.00$	 $d_{3D} = 30.70$	
DeepSfT			 $E_{Ph} = 0.30$	 $E_D = 188.57$	 $d_{3D} = 139.24$	
DeepSfT+ARAP			 $E_{Ph} = 0.30$	 $E_D = 130.71$	 $d_{3D} = 74.17$	
WS-DeepSfT			 $E_{Ph} = 0.25$	 $E_D = 17.69$	 $d_{3D} = 3.92$	

TABLE 14: Qualitative results in example 1 of the “Fruits” real dataset are shown. Lighter pixels correspond to higher errors. The colour representation saturates at 0.5 for the photometric error and at 250 mm for the depth and 3D distance maps. In the 3D Mesh Comparison column, the ground-truth surface is coloured blue, and the predicted mesh is coloured red.


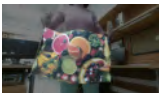
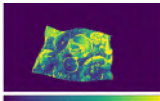
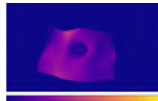
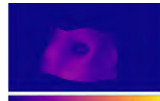
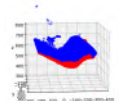


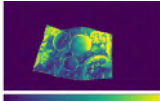


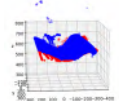

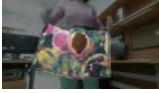
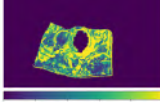
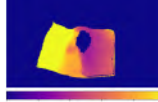
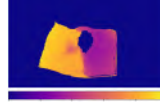
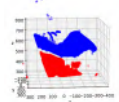


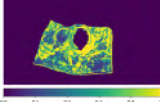
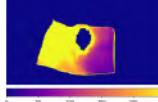
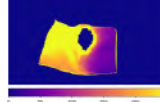
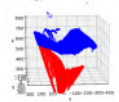


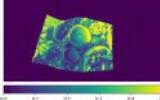
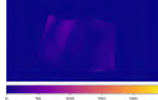
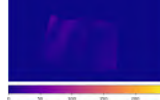
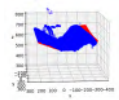
Model	$I'$	$I_{over}$	Photometric Error (n.u.)	Depth Error (mm)	3D Error (mm)	3D Mesh Comparison
LightGlue+BS			 $E_{Ph} = 16.34$	 $E_D = 53.20$	 $d_{3D} = 38.34$	
RAFT			 $E_{Ph} = 0.25$	 $E_D = 22.39$	 $d_{3D} = 16.34$	
DeepSfT			 $E_{Ph} = 0.38$	 $E_D = 211.28$	 $d_{3D} = 158.35$	
DeepSfT+ARAP			 $E_{Ph} = 0.38$	 $E_D = 267.33$	 $d_{3D} = 195.65$	
WS-DeepSfT			 $E_{Ph} = 0.26$	 $E_D = 13.85$	 $d_{3D} = 7.49$	

TABLE 15: Qualitative results in example 2 of the “Fruits” real dataset are shown. Lighter pixels correspond to higher errors. The colour representation saturates at 0.5 for the photometric error and at 250 mm for the depth and 3D distance maps. In the 3D Mesh Comparison column, the ground-truth surface is coloured blue, and the predicted mesh is coloured red.





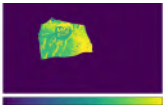


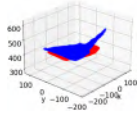


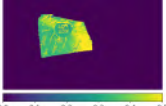
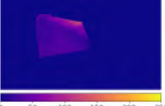
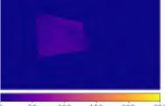
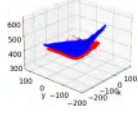


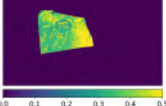
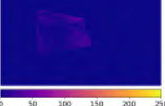
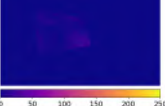
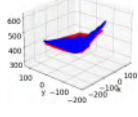
Model	$I'$	$I_{over}$	Photometric Error (n.u.)	Depth Error (mm)	3D Error (mm)	3D Mesh Comparison
LightGlue+BS			 $E_{Ph} = 0.43$	 $E_D = 36.47$	 $d_{3D} = 18.52$	
RAFT			 $E_{Ph} = 0.42$	 $E_D = 39.26$	 $d_{3D} = 23.46$	
WS-DeepSfT			 $E_{Ph} = 0.42$	 $E_D = 13.08$	 $d_{3D} = 6.45$	

TABLE 16: Qualitative results in example 1 of the “Cloth” real dataset are shown. Lighter pixels correspond to higher errors. The colour representation saturates at 0.5 for the photometric error and at 250 mm for the depth and 3D distance maps. In the 3D Mesh Comparison column, the ground-truth surface is coloured blue, and the predicted mesh is coloured red.



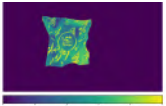

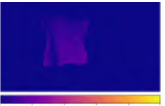
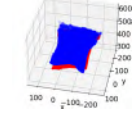


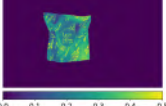
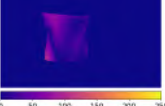
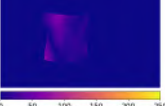
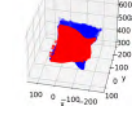

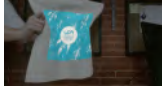
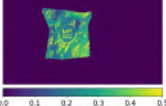
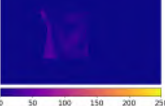
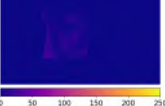
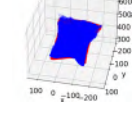
Model	$I'$	$I_{over}$	Photometric Error (n.u.)	Depth Error (mm)	3D Error (mm)	3D Mesh Comparison
LightGlue+BS			 $E_{Ph} = 0.30$	 $E_D = 30.74$	 $d_{3D} = 20.74$	
RAFT			 $E_{Ph} = 0.28$	 $E_D = 35.01$	 $d_{3D} = 20.02$	
WS-DeepSfT			 $E_{Ph} = 0.30$	 $E_D = 14.04$	 $d_{3D} = 7.33$	

TABLE 17: Qualitative results in example 2 of the “Cloth” real dataset are shown. Lighter pixels correspond to higher errors. The colour representation saturates at 0.5 for the photometric error and at 250 mm for the depth and 3D distance maps. In the 3D Mesh Comparison column, the ground-truth surface is coloured blue, and the predicted mesh is coloured red.

## F. ABLATION STUDY

We evaluate the network to make design choices about its hyperparameters. We compare different parametrisations on the challenging synthetic dataset “Zaius”, to quantitatively analyse its performance in complicated environments and templates. For this study, the DNN is trained in the training split and evaluated on a 500-image subset of the validation split; thus the test split is only used to compare against the other state-of-the-art methods and it has not been seen previously. We use the registration ( $E_W$ ) and photometric errors ( $E_{Ph}$ ) to analyse the different versions of the model. We present the results of the ablation study in Table 18.

### 1) Calibrating The Gradient Descent

The system has been trained with SGD, since other optimisers presented worse convergence. Furthermore, the introduction of the optical flow and smoothing losses is delayed 20 epochs, to avoid an early degradation of the gradient due to noise and undesired local minima. We also explore the learning rate parameter space to achieve an optimal gradient descent. We have found that a learning rate of  $10^{-4}$  obtains the best results in RMSE and photometric metrics, with an improvement of up to 18.5%.

Model Hyperparameters									
Learning Rate	$\lambda_R$	$c_R$	$\lambda_M$	$c_M$	$\lambda_S$	Max. Step	Delay	Registration Error (mm)	PhotometricError (n.u.)
0.0010	1	0.010	0.25	0.015	0	5	0	73.090	0.16060
0.0010	1	0.010	0.25	0.015	0	5	20	8.557	0.08949
0.0050	1	0.010	0.25	0.015	0	5	20	7.255	0.08158
0.0001	1	0.010	0.25	0.015	0	5	20	<b>7.222</b>	<b>0.08124</b>
0.0005	1	0.010	0.25	0.015	0	5	20	7.808	0.08359
0.0001	1	0.010	0.25	0.001	0	5	20	8.184	0.08290
0.0001	1	0.010	0.25	0.005	0	5	20	<b>6.618</b>	<b>0.07966</b>
0.0001	1	0.010	0.25	0.010	0	5	20	6.785	0.07974
0.0001	1	0.010	0.25	0.050	0	5	20	7.665	0.08288
0.0001	1	0.010	0.00	-	0	-	-	8.901	0.08576
0.0001	1	0.010	0.05	0.005	0	5	20	8.555	0.08592
0.0001	1	0.010	0.10	0.005	0	5	20	7.347	0.08199
0.0001	1	0.010	0.75	0.005	0	5	20	6.141	0.07818
0.0001	1	0.010	1.25	0.005	0	5	20	<b>5.797</b>	<b>0.07624</b>
0.0001	1	0.010	1.50	0.005	0	5	20	5.924	0.07642
0.0001	1	0.010	2.00	0.005	0	5	20	5.817	0.07932
0.0001	1	0.010	1.25	0.005	10	5	20	6.318	0.08285
0.0001	1	0.010	1.25	0.005	25	5	20	<b>5.696</b>	0.07767
0.0001	1	0.010	1.25	0.005	50	5	20	5.836	<b>0.07658</b>
0.0001	0.1	0.010	1.25	0.005	25	5	20	304.000	-
0.0001	1	0.010	1.25	0.005	25	5	20	<b>5.696</b>	<b>0.07767</b>
0.0001	10	0.010	1.25	0.005	25	5	20	9.049	0.08596
0.0001	1	0.001	1.25	0.005	25	5	20	302.100	-
0.0001	1	0.010	1.25	0.005	25	5	20	<b>5.696</b>	<b>0.07767</b>
0.0001	1	0.100	1.25	0.005	25	5	20	21.540	0.12950

TABLE 18: Results of the ablation study. In the first section, we evaluate the optimum learning rate and the importance of delaying the introduction of the matching and smoothing loss. Next, we test different configurations for the parameters and weights of the matching loss. Afterwards, we assess the impact of a regularisation loss. Finally, we validate the chosen values for they rigidity loss hyperparameters. We use the registration error and the photometric error to measure the performance of each variation of the model. Missing photometric errors correspond to predictions where the model collapsed and, therefore, the synthesized image could not be generated.

## 2) Matching Loss Parametrisation

We select the Welsch Loss function to compute the matching loss  $L_M$ , aiming to prevent errors from the optical flow prediction from propagating through the network training. The Welsch function includes the parameter  $c$  to control the penalty applied to mismatches, which is crucial to be managed carefully. Low values remove important information, worsening the performance from a registration error of 7.222 mm to 8.184 mm in our experiments. On the other hand, excessively high values reduce the weight of useful information and increase the influence of mismatches, leading to an additional error of 0.443 mm compared to the initial value of  $c_M = 0.015$ . After carefully analysing this parameter range, we set  $c_M$  to 0.005, achieving an 8.36% reduction in the registration error and an 8.8% reduction in the photometric error compared to the initial value, and a 19.1349% and 3.9083% reduction, respectively, compared to the worst-case scenario.

Next, we evaluate the balance between the matching loss  $L_M$  and the rigidity loss  $L_R$ . We set  $\lambda_R = 1$  for the rigidity loss with  $c_R = 0.01$ , and then analyse the model's performance with different weights for the matching loss.

The best results are achieved with  $\lambda_M = 1.25$ , which reduces the error by 34.85% compared to the baseline model that uses only the rigidity loss. Although the rigidity loss achieves acceptable error rates, a closer examination shows it fails to accurately estimate deformations, producing good results only on flat parts of the surface. The proposed loss function effectively reduces the error by focusing on learning from the deformed sections of the object. Lower values of  $\lambda_M$  increase the influence of the rigidity loss, which does not perform well with deformations, resulting in higher average errors. Conversely, higher  $\lambda_M$  values do not improve performance because mismatches in the optical flow prediction dominate the rigidity loss information, introducing more noise and leading to poorer predictions.

## 3) The Benefits of Regularisation

Finally, we add a smoothing loss to introduce regularisation and increase the performance of the system. This loss aids us to prevent overfitting and reduce the impact of noisy parameters. We achieve a reduction of 1.74% of the registration error with  $\lambda_S = 25$  for the smoothing loss. We select  $\lambda_S = 25$  over the other options due to its lower registration error, since this

metric is more precise than the photometric error, which is influenced by noise.

#### 4) The Critical Role of the Rigidity Loss

The hyperparameterisation of the rigidity loss is crucial for the system's performance. A low weight, such as  $\lambda_R = 0.1$ , fails to prevent gradient degradation, whereas higher values like  $\lambda_R = 10$  negatively affect the model's ability to adapt to deformations, resulting in suboptimal results. Through our experiments, we found that  $\lambda_R = 1$  provides the best balance, offering stability without compromising the model's performance under deformation. The parameter  $c_R$ , which regulates the Welsch loss, also has a significant impact on the solution's effectiveness. Smaller values, such as  $c_R = 0.001$ , lead to loss saturation and prevent the model from learning properly, causing training to become stuck in a local minimum with no viable predictions. Conversely, larger values like  $c_R = 0.1$  stretch the Welsch function along the  $x$ -axis, resulting in a lower rigidity loss but ultimately producing suboptimal performance due to poor loss balancing.

#### 5) Strengths and Weaknesses of the Proposed Method

The proposed solution addresses several limitations of current state-of-the-art methods, resulting in significantly reduced errors in the estimated registration and reconstruction. Our wide-baseline approach proves more robust than short-baseline methods, such as RAFT, when coping with variations in illumination and deformation, and it does not require temporal coherence during inference. Furthermore, the adopted network architecture performs well even in regions with limited texture, where feature-based algorithms, such as LightGlue+BS, struggle to detect or accurately match scene features to those of the template. We also present a novel weakly supervised strategy that can be trained directly on real-world, unlabelled data, avoiding the need to synthesise training datasets as required by DeepSfT. Since real images are inherently more complex than synthetic ones, previous deep neural learning-based SfT approaches often encountered difficulties when real-world conditions deviated from the training domain. By training solely on real data, our method avoids these domain gaps and, in so doing, both improves performance and reduces the overall data requirements.

The proposed method nonetheless has weaknesses. Our weakly supervised approach presupposes the availability of keyframes in the training data, which imposes constraints on how the data is recorded. Ensuring accurate pose estimation between the template and these keyframes is critical to prevent degeneracy during training via the rigidity loss. In addition, the main supervision signal arises from optical flow estimation and the associated matching loss function, making highly accurate optical flow calculations essential for successful registration. Finally, the method relies on semantic segmentation to identify the template within the image. While we assume this step to be largely addressed by foundational models such as SAM—demonstrably effective

across a wide range of objects and scenarios—this reliance introduces an additional external dependency in the pipeline.

## VII. CONCLUSIONS

We introduced WS-DeepSfT, the very first dense, wide-baseline, real-time SfT solution for deformable surfaces based on deep learning that can be directly trained on non-annotated real data. WS-DeepSfT uses optical flow and pose estimation to exploit the temporal and template information from video sequences to solve SfT accurately. The proposed approach is the first DNN SfT method to predict high-resolution meshes without requiring special sensors or synthetic labelled data, thus avoiding the render gap by training directly in real images. Furthermore, it provides higher robustness compared to state-of-the-art methods by learning a dense representation of the object, thus not relying on appearance-specific features or spatiotemporal correlation during inference.

Experiments on synthetic and real thin-shell datasets show that it outperforms the existing methods and achieves a great reduction of catastrophic failures, gross errors and outliers, which leads to a sizable boost in performance metrics. WS-DeepSfT is especially desirable for critical applications due to its capacity to produce highly reliable predictions. Future works include additional self-supervision methodologies exploiting other image cues such as colour or shading, extending the method to work with generic objects of different topologies and adding a pre-processing step to enhance the quality of the input frames.

## ACKNOWLEDGMENT

This work was supported by the Spanish Ministry of Education under the grants for Training of University Teachers (FPU19/04500) and by the Spanish Ministry of Science and Innovation MCIN/AEI/10.13039/501100011033 through project ATHENA (PID2020-115995RB-I00) and project METAMORPH (PID2023-151295OB-I00).

The code and datasets used in this research are publicly available at <https://github.com/saraluengo-uah/WSDeepSfT> to ensure the reproducibility of our findings and facilitate future research.

## REFERENCES

- [1] A. Bartoli, Y. Gérard, F. Chadebecq, T. Collins, and D. Pizarro, "Shape-from-template," *IEEE transactions on pattern analysis and machine intelligence*, vol. 37, no. 10, pp. 2099–2118, 2015.
- [2] M. Zollhöfer, J. Thies, P. Garrido, D. Bradley, T. Beeler, P. Pérez, M. Stamminger, M. Nießner, and C. Theobalt, "State of the art on monocular 3d face reconstruction, tracking, and applications," in *Computer graphics forum*, vol. 37, no. 2. Wiley Online Library, 2018, pp. 523–550.
- [3] Y. Feng, H. Feng, M. J. Black, and T. Bolkart, "Learning an animatable detailed 3d face model from in-the-wild images," *ACM Transactions on Graphics (ToG)*, vol. 40, no. 4, pp. 1–13, 2021.
- [4] G. Gafni, J. Thies, M. Zollhofer, and M. Nießner, "Dynamic neural radiance fields for monocular 4d facial avatar reconstruction," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 8649–8658.
- [5] Y. Liu, Z. Ge, Y. Yuan, X. Su, X. Guo, T. Suo, and Q. Yu, "Wing deformation measurement using the stereo-vision methods in the presence

- of camera movements,” *Aerospace Science and Technology*, vol. 119, p. 107161, 2021.
- [6] Z. Yang, Y. Wang, Y. Jiang, H. Zhang, and C. Yang, “Deformernet based 3d deformable objects shape servo control for bimanual robot manipulation,” in *2024 IEEE International Conference on Industrial Technology (ICIT)*. IEEE, 2024, pp. 1–7.
  - [7] M. T. Pain, C. Mills, and M. R. Yeadon, “Video analysis of the deformation and effective mass of gymnastics landing mats,” *Medicine and science in sports and exercise*, vol. 37, no. 10, pp. 1754–1760, 2005.
  - [8] T. Peng, W. Wu, J. Liu, L. Li, J. Miao, X. Hu, R. He, and L. Li, “Pgn-cloth: Physics-based graph network model for 3d cloth animation,” *Displays*, vol. 80, p. 102534, 2023.
  - [9] N. S. Phatak, Q. Sun, S.-E. Kim, D. L. Parker, R. Kent Sanders, A. I. Veress, B. J. Ellis, and J. A. Weiss, “Noninvasive determination of ligament strain with deformable image registration,” *Annals of biomedical engineering*, vol. 35, pp. 1175–1187, 2007.
  - [10] S. Shimada, V. Golyanik, P. Pérez, and C. Theobalt, “Decaf: Monocular deformation capture for face and hand interactions,” *ACM Transactions on Graphics (ToG)*, vol. 42, no. 6, pp. 1–16, 2023.
  - [11] N. Haouchine, P. Juvekar, W. M. Wells III, S. Cotin, A. Golby, and S. Frisken, “Deformation aware augmented reality for craniotomy using 3d/2d non-rigid registration of cortical vessels,” in *Medical Image Computing and Computer Assisted Intervention—MICCAI 2020: 23rd International Conference, Lima, Peru, October 4–8, 2020, Proceedings, Part IV 23*. Springer, 2020, pp. 735–744.
  - [12] L. Maier-Hein, P. Mountney, A. Bartoli, H. Elhawary, D. Elson, A. Groch, A. Kolb, M. Rodrigues, J. Sorger, S. Speidel, and D. Stoyanov, “Optical techniques for 3d surface reconstruction in computer-assisted laparoscopic surgery,” *Medical image analysis*, vol. 17, no. 8, pp. 974–996, 2013.
  - [13] Y.-H. Su, K. Lindgren, K. Huang, and B. Hannaford, “A comparison of surgical cavity 3d reconstruction methods,” in *2020 IEEE/SICE International Symposium on System Integration (SII)*. IEEE, 2020, pp. 329–336.
  - [14] J. J. G. Rodríguez, J. M. Montiel, and J. D. Tardós, “Nr-slam: Non-rigid monocular slam,” *IEEE Transactions on Robotics*, 2024.
  - [15] L. Xu, H. Zhang, J. Wang, A. Li, S. Song, H. Ren, L. Qi, J. J. Gu, and M. Q.-H. Meng, “Information loss challenges in surgical navigation systems: From information fusion to ai-based approaches,” *Information Fusion*, vol. 92, pp. 13–36, 2023.
  - [16] J. Shan, Y. Li, and H. Wang, “Endo-slam: A dense endoscopic slam with neural implicit representation,” in *2023 China Automation Congress (CAC)*. IEEE, 2023, pp. 8417–8422.
  - [17] Z. Yang, J. Dai, and J. Pan, “3d reconstruction from endoscopy images: A survey,” *Computers in Biology and Medicine*, p. 108546, 2024.
  - [18] A. Chhatkuli, D. Pizarro, A. Bartoli, and T. Collins, “A stable analytical framework for isometric shape-from-template by surface integration,” *IEEE transactions on pattern analysis and machine intelligence*, vol. 39, no. 5, pp. 833–850, 2016.
  - [19] F. Brunet, A. Bartoli, and R. I. Hartley, “Monocular template-based 3d surface reconstruction: Convex inextensible and nonconvex isometric methods,” *Computer Vision and Image Understanding*, vol. 125, pp. 138–154, 2014.
  - [20] D. T. Ngo, S. Park, A. Jorstad, A. Crivellaro, C. D. Yoo, and P. Fua, “Dense image registration and deformable surface reconstruction in presence of occlusions and minimal texture,” in *Proceedings of the IEEE International Conference on Computer Vision*, 2015, pp. 2273–2281.
  - [21] D. Casillas-Perez, D. Pizarro, D. Fuentes-Jimenez, M. Mazo, and A. Bartoli, “Equiareal shape-from-template,” *Journal of Mathematical Imaging and Vision*, vol. 61, pp. 607–626, 2019.
  - [22] A. Malti, R. Hartley, A. Bartoli, and J.-H. Kim, “Monocular template-based 3d reconstruction of extensible surfaces with local linear elasticity,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2013, pp. 1522–1529.
  - [23] A. Malti, A. Bartoli, and R. Hartley, “A linear least-squares solution to elastic shape-from-template,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 1629–1637.
  - [24] D. Fuentes-Jimenez, D. Pizarro, D. Casillas-Pérez, T. Collins, and A. Bartoli, “Deep shape-from-template: Single-image quasi-isometric deformable registration and reconstruction,” *Image and Vision Computing*, vol. 127, p. 104531, 2022. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S0262885622001603>
  - [25] O. Sorkine and M. Alexa, “As-rigid-as-possible surface modeling,” in *Symposium on Geometry processing*, vol. 4. Citeseer, 2007, pp. 109–116.
  - [26] M. Gallardo, D. Pizarro, T. Collins, and A. Bartoli, “Shape-from-template with curves,” *International Journal of Computer Vision*, vol. 128, pp. 121–165, 2020.
  - [27] D. Casillas-Perez, D. Pizarro, D. Fuentes-Jimenez, M. Mazo, and A. Bartoli, “The isowarp: the template-based visual geometry of isometric surfaces,” *International Journal of Computer Vision*, vol. 129, no. 7, pp. 2194–2222, 2021.
  - [28] D. T. Ngo, J. Östlund, and P. Fua, “Template-based monocular 3d shape recovery using laplacian meshes,” *IEEE transactions on pattern analysis and machine intelligence*, vol. 38, no. 1, pp. 172–187, 2015.
  - [29] T. Collins, A. Bartoli, N. Bourdel, and M. Canis, “Robust, real-time, dense and deformable 3d organ tracking in laparoscopic videos,” in *International Conference on Medical Image Computing and Computer-Assisted Intervention (MICCAI)*. Springer, 2016, pp. 404–412.
  - [30] Q. Liu-Yin, R. Yu, L. Agapito, A. Fitzgibbon, and C. Russell, “Better together: Joint reasoning for non-rigid 3d reconstruction with specularities and shading,” *arXiv preprint arXiv:1708.01654*, 2017.
  - [31] R. Yu, C. Russell, N. D. Campbell, and L. Agapito, “Direct, dense, and deformable: Template-based non-rigid 3d reconstruction from rgb video,” in *Proceedings of the IEEE international conference on computer vision (ICCV)*, 2015, pp. 918–926.
  - [32] A. Malti and C. Herzet, “Elastic shape-from-template with spatially sparse deformation forces,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 3337–3345.
  - [33] E. Özgür and A. Bartoli, “Particle-sft: A provably-convergent, fast shape-from-template algorithm,” *International Journal of Computer Vision (IJCV)*, vol. 123, pp. 184–205, 2017.
  - [34] N. Haouchine and S. Cotin, “Template-based monocular 3d recovery of elastic shapes using lagrangian multipliers,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 4095–4103.
  - [35] M. Shetab-Bushehri, M. Aranda, E. Özgür, Y. Mezouar, and A. Bartoli, “Robusft: Robust real-time shape-from-template, a c++ library,” *Image and Vision Computing*, vol. 141, p. 104867, 2024.
  - [36] D. G. Lowe, “Distinctive image features from scale-invariant keypoints,” *International Journal of Computer Vision (IJCV)*, vol. 60, no. 2, pp. 91–110, 2004.
  - [37] V. Badrinarayanan, A. Kendall, and R. Cipolla, “Segnet: A deep convolutional encoder-decoder architecture for image segmentation,” *IEEE transactions on pattern analysis and machine intelligence*, vol. 39, no. 12, pp. 2481–2495, 2017.
  - [38] N. Kairanda, E. Tretschk, M. Elgharib, C. Theobalt, and V. Golyanik, “f-sft: Shape-from-template with a physics-based deformation model,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 3948–3958.
  - [39] D. Stotko, N. Wandel, and R. Klein, “Physics-guided shape-from-template: monocular video perception through neural surrogate models,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024, pp. 11 895–11 904.
  - [40] D. Tan, H. Yang, Z. Jiang, W. Shi, J. Qin, and F. Qu, “Improved shape-from-template method with perspective space constraints for disappearing features,” *Complex & Intelligent Systems*, pp. 1–14, 2024.
  - [41] A. Pumarola, A. Agudo, L. Porzi, A. Sanfeliu, V. Lepetit, and F. Moreno-Noguer, “Geometry-aware network for non-rigid shape prediction from a single view,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 4681–4690.
  - [42] V. Golyanik, S. Shimada, K. Varanasi, and D. Stricker, “Hdm-net: Monocular non-rigid 3d reconstruction with learned deformation model,” in *Virtual Reality and Augmented Reality: 15th EuroVR International Conference, EuroVR 2018, London, UK, October 22–23, 2018, Proceedings 15*. Springer, 2018, pp. 51–72.
  - [43] S. Shimada, V. Golyanik, C. Theobalt, and D. Stricker, “Ismo-gan: Adversarial learning for monocular non-rigid 3d reconstruction,” in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition workshops*, 2019, pp. 0–0.
  - [44] D. Fuentes-Jimenez, D. Pizarro, D. Casillas-Perez, T. Collins, and A. Bartoli, “Texture-generic deep shape-from-template,” *IEEE Access*, vol. 9, pp. 75 211–75 230, 2021.
  - [45] M. A. Fischler and R. C. Bolles, “Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography,” *Communications of the ACM*, vol. 24, no. 6, pp. 381–395, 1981.



- [46] S. Li, C. Xu, and M. Xie, "A robust  $o(n)$  solution to the perspective- $n$ -point problem," *IEEE transactions on pattern analysis and machine intelligence*, vol. 34, no. 7, pp. 1444–1450, 2012.
- [47] J. Yang, M. Gao, Z. Li, S. Gao, F. Wang, and F. Zheng, "Track anything: Segment anything meets videos," *arXiv preprint arXiv:2304.11968*, 2023.
- [48] A. Kirillov, E. Mintun, N. Ravi, H. Mao, C. Rolland, L. Gustafson, T. Xiao, S. Whitehead, A. C. Berg, W.-Y. Lo *et al.*, "Segment anything," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2023, pp. 4015–4026.
- [49] J. T. Barron, "A general and adaptive robust loss function," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2019, pp. 4331–4339.
- [50] D. Sun, X. Yang, M.-Y. Liu, and J. Kautz, "Pwc-net: Cnns for optical flow using pyramid, warping, and cost volume," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 8934–8943.
- [51] P. Lindenberger, P.-E. Sarlin, and M. Pollefeys, "Lightglue: Local feature matching at light speed," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2023, pp. 17 627–17 638.
- [52] P.-E. Sarlin, D. DeTone, T. Malisiewicz, and A. Rabinovich, "Superglue: Learning feature matching with graph neural networks," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020, pp. 4938–4947.
- [53] D. DeTone, T. Malisiewicz, and A. Rabinovich, "Superpoint: Self-supervised interest point detection and description," in *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, 2018, pp. 224–236.
- [54] M. Tyszkiewicz, P. Fua, and E. Trulls, "Disk: Learning local features with policy gradient," *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 33, pp. 14 254–14 265, 2020.
- [55] C. Tomasi and R. Manduchi, "Bilateral filtering for gray and color images," in *Sixth international conference on computer vision (IEEE Cat. No. 98CH36271)*. IEEE, 1998, pp. 839–846.
- [56] T. Collins and A. Bartoli, "[poster] realtime shape-from-template: System and applications," in *2015 IEEE International Symposium on Mixed and Augmented Reality*. IEEE, 2015, pp. 116–119.
- [57] O. Sorkine and M. Alexa, "As-rigid-as-possible surface modeling," in *Symposium on Geometry processing*, vol. 4. Citeseer, 2007, pp. 109–116.
- [58] R. W. Sumner and J. Popović, "Deformation transfer for triangle meshes," *ACM Transactions on graphics (TOG)*, vol. 23, no. 3, pp. 399–405, 2004.
- [59] Z. Teed and J. Deng, "Raft: Recurrent all-pairs field transforms for optical flow," in *Computer Vision—ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part II 16*. Springer, 2020, pp. 402–419.



SARA LUENGO received a B.S. degree and M.S. degree in telecommunications engineering from the University of Alcalá, Alcalá de Henares, Madrid, Spain, in 2018 and 2020 respectively, where she is currently pursuing a Ph.D. degree in artificial intelligence.

She has been a member of the GEINTRA research group since 2016. Her research interests are computer vision, people detection and re-identification, image registration and 3D object

reconstruction.

Ms. Luengo was awarded an excellence prize for her studies in 2016 by the Madrid regional government. She received the José Ramón López Villares and the CEI Intelligent Energy prizes in 2019. She was granted the IBM prize for the best academic record from the Official College of Telecommunication Engineers (COIT) and the Spanish Association of Telecommunication Engineers (AEIT) in 2021. She held the University Professor Training Scholarship (FPU) from the Spanish government from 2019 until 2023.



DAVID FUENTES received a B.S. degree in electronics engineering in 2016, an M.S. degree in electrical engineering in 2018, and a Ph.D. degree in artificial intelligence in 2021 from the University of Alcalá, Alcalá de Henares, Madrid, Spain.

He worked as a research scientist in NaturalVox S.A.U. from 2021 to 2023, specializing in face and voice biometry. Since 2023, he has been an assistant professor at the University of Alcalá and a member of the GEINTRA research group. His

researchs interest are computer vision, 3D object reconstruction and people detection with depth cameras.

Dr. Fuentes has been a Nova Talent Member since 2022. He held the Researcher Training Scholarship (FPI) from the University of Alcalá from 2018 until 2021. He received the CEI Intelligent Energy in 2020.



CRISTINA LOSADA received a B.S. degree in telecommunications engineering and a Ph.D. degree in advanced intelligent systems electronics from the University of Alcalá (UAH), Alcalá de Henares, Madrid, Spain, in 2004 and 2010, respectively.

Since 2011, she has been a professor at the Electronic Department of the University of Alcalá and a member of the GEINTRA Research Group. She is also assistant director of the Electronic

department since 2022. His research interests are computer vision and sensor systems applied to intelligent spaces and human behaviour analysis.



DANIEL PIZARRO received his B.S. and Ph.D. degrees from the University of Alcalá, Alcalá de Henares, Madrid, Spain, in 2003 and 2008, respectively.

He was an Assistant Professor and member of the GEINTRA research group at the Department of Electronics at the University of Alcalá from 2005 to 2012. From 2012 to 2014, he joined the Advanced Laparoscopy and Computer Vision Group (AICoV) and was an Associate Professor

at the Université d'Auvergne, Clermont-Ferrand, France. Since 2015, he has been a professor at the University of Alcalá and a member of the GEINTRA research group, which is currently leading since 2023. He is also an associated member of the Endoscopy and Computer Vision (ENCoV) Research Group. His research interests include image registration, rigid and non-rigid reconstruction, and its applications to minimally invasive surgery.

Dr. Pizarro was awarded the Young Researcher Prize in the area of Experimental Sciences in 2015 and the prize for excellence in Ph.D. supervision in 2018 from the University of Alcalá.



ADRIEN BARTOLI received his Ph.D. from the Perception Group, Inria Grenoble, Grenoble, France, in 2003, and his Habilitation degree (HDR) from Université Blaise Pascal, Clermont-Ferrand, France, in 2008.

He started as a postdoctoral researcher in the Visual Geometry Group at the University of Oxford under Andrew Zisserman in 2004. He joined the CNRS research scientist at Institut Pascal in 2004 where he led ComSee, the Computer Vision research group. He stayed as a Visiting professor in DIKU at the University of Copenhagen, Copenhagen, Denmark, from 2006 to 2009. He has been a professor of Computer Science at Université Clermont Auvergne, Clermont-Ferrand, France, since 2009. He is the chief scientific officer in SurgAR and leads the Endoscopy and Computer Vision (ENCoV) Research Group. His research interests include image registration and Shape-from-X in rigid and non-rigid environments, and its applications to computer-aided medical interventions, particularly endoscopy.

Dr. Bartoli has been a member of the Institut Universitaire de France since 2016. He received the Grenoble-INP Ph.D. thesis prize in 2004, the CNRS médaille de bronze in 2008 and a research prize from Université d'Auvergne in 2016. He held an ERC Consolidator Grant from 2013 to 2018 and an ERC Proof-of-Concept Grant from 2018 to 2019. He is on the editorial board of the International Journal of Computer Vision (IJCV) and the Journal of Artificial Intelligence Research (JAIR).

...