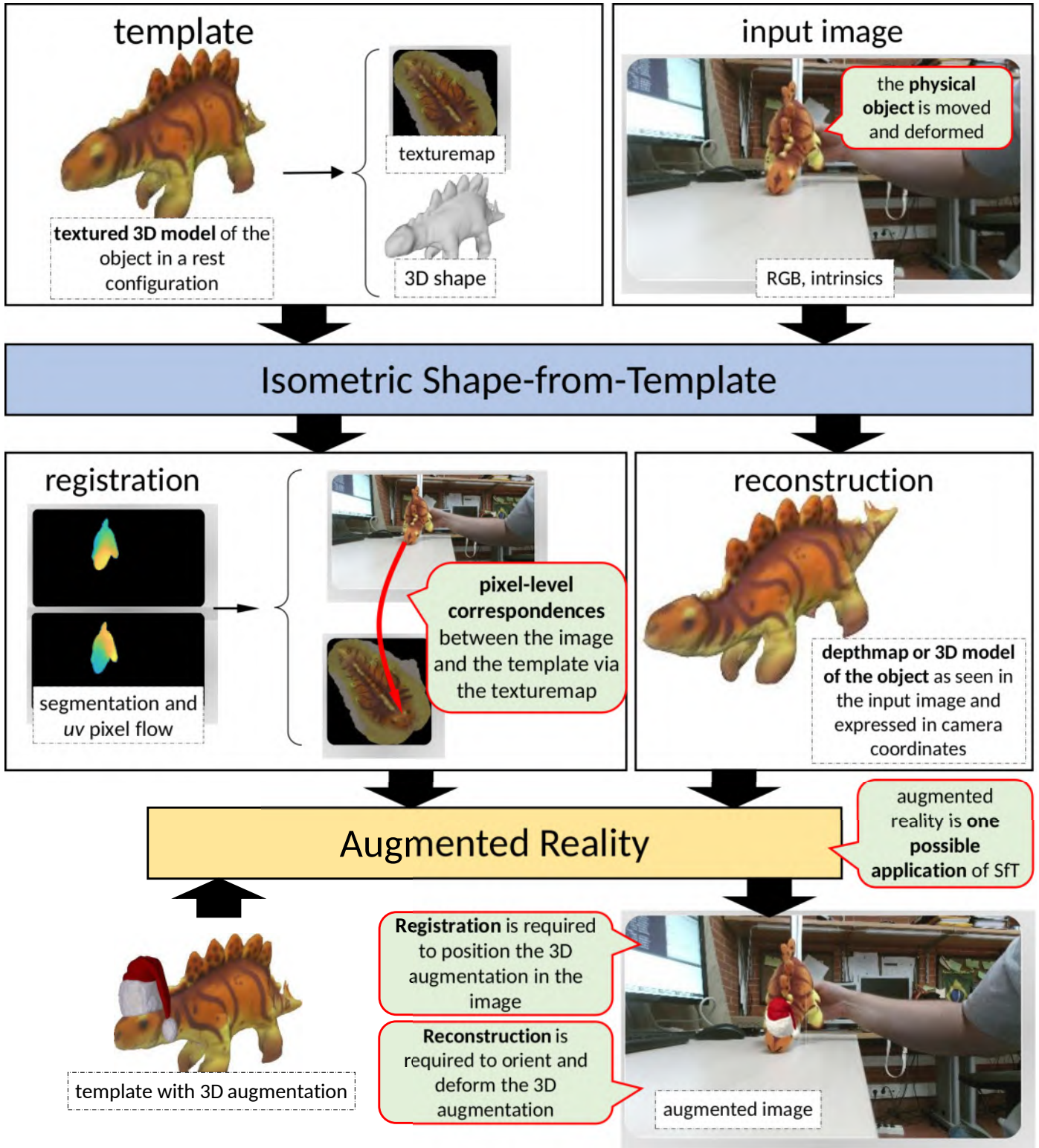


# Graphical Abstract

## Deep Shape-from-Template: Single-image Quasi-isometric Deformable Registration and Reconstruction

David Fuentes-Jimenez, Daniel Pizarro, D. Casillas-Pérez, Toby Collins, Adrien Bartoli



# Highlights

## **Deep Shape-from-Template: Single-image Quasi-isometric Deformable Registration and Reconstruction**

David Fuentes-Jimenez, Daniel Pizarro, D. Casillas-Pérez, Toby Collins, Adrien Bartoli

- We propose DeepSfT, a novel DNN fully-convolutional and based on residual encoder-decoder structures with refining blocks. specifically tailored to SfT.
- We propose a semi-supervised end-to-end training procedure, capable of training from synthetic and real data.
- DeepSfT is a solution to cope with multiple imaging geometries (caused by changing the intrinsics of the physical camera, typically by zooming in or out, or by using a different camera), at training and inference.
- We propose to combine DeepSfT with a physics-based estimation procedure.

# Deep Shape-from-Template: Single-image Quasi-isometric Deformable Registration and Reconstruction

David Fuentes-Jimenez<sup>a</sup>, Daniel Pizarro<sup>a,d</sup>, D. Casillas-Pérez<sup>c,\*</sup>, Toby Collins<sup>b</sup>, Adrien Bartoli<sup>d</sup>

<sup>a</sup>Department of Electronics, Universidad de Alcalá, 28943, Alcalá de Henares. Spain.

<sup>b</sup>IRCAD, Place de l'Hôpital, Strasbourg 67000, France.

<sup>c</sup>Department of Signal Theory and Communications, Universidad Rey Juan Carlos, 28942, Fuenlabrada, Spain.

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

---

## Abstract

Shape-from-Template (SfT) solves 3D vision from a single image and a deformable 3D object model, called a template. Concretely, SfT computes registration (the correspondence between the template and the image) and reconstruction (the depth in camera frame). It constrains the object deformation to quasi-isometry. Real-time and automatic SfT represents an open problem for complex objects and imaging conditions. We present four contributions to address core unmet challenges to realise SfT with a Deep Neural Network (DNN). First, we propose a novel DNN called DeepSfT, which encodes the template in its weights and hence copes with highly complex templates. Second, we propose a semi-supervised training procedure to exploit real data. This is a practical solution to overcome the render gap that occurs when training only with simulated data. Third, we propose a geometry adaptation module to deal with different cameras at training and inference. Fourth, we combine statistical learning with physics-based reasoning. DeepSfT runs automatically and in real-time and we show with numerous experiments and an ablation study that it consistently achieves a lower 3D error than previous work. It outperforms in generalisation and achieves great performance in terms of reconstruction and registration error with wide-baseline, occlusions, illumination changes, weak texture and blur.

*Keywords:* Monocular, 3D Model, Registration, Reconstruction, Wide-baseline, Dense, Deformable, Shape-from-Template

*PACS:* 0000, 1111

*2000 MSC:* 0000, 1111

---

## 1. Introduction

### 1.1. Context and the SfT problem

The tasks of image registration (i.e., the computation of correspondences) and image-based reconstruction (i.e., the computation of depth) are fundamental in computer vision<sup>1</sup>. Solving both tasks is required in applications such as 3D object tracking and augmented reality. To date, there exist mature techniques for rigid objects, such as Structure-from-Motion (SfM) [1]. The case of deformable objects is however largely unresolved. The existing work has considered two main scenarios. In Non-Rigid SfM (NRSfM) [2, 3, 4, 5], the inputs are a set of images and the problem is to find correspondences across images (registration) and depth (reconstruction). In Shape-from-Template (SfT) [6, 7, 8, 9, 10], the inputs are a single image, a 3D object model (template) is known, and the problem is to find correspondences between the model and the image (registration) and depth (reconstruction). Obviously, as the object is deformable, the image is not a photo of the model under some unknown pose: rather, it is a photo of the model taken after some unknown deformation. The most common type of deformation

prior used in NRSfM and SfT is the widely applicable *quasi-isometry*, which prevents significant stretching or shrinking of the object. An illustration of SfT is shown in figure 1. A very important concept in SfT is the *template*, which is the known textured 3D object model. Concretely, the template is a 3D shape (e.g., a triangulated 3D mesh) and a texture map (e.g. an image giving colours for the mesh's facets), which is acquired straightforwardly using a 3D scanner, an RGB-D sensor or SfM.

SfT is a difficult and unresolved problem. The core challenges are related to the object (typically, a rich texture and a flat template shape are easier to deal with), to the imaging conditions (typically, a sharp and well-lit image with strong visibility are easier to deal with) and to the availability of an initial solution guess. The latter is generally available when the input image is extracted from a continuous video, where the solution to the past frame forms a guess for the current frame, and forms the so-called *short-baseline* case. In contrast, the *wide-baseline* case occurs when the input image is processed individually, without having a solution guess. The short-baseline condition (that a solution guess is available) is obviously a strong weakness, as it assumes that camera motion and object deformation are small between frames, and fails if, for instance, the object goes outside the field of view. The wide-baseline case, despite its increased difficulty, is thus very important to achieve

---

\*Corresponding author: David Casillas-Pérez. Department of Signal Theory and Communications, Universidad Rey Juan Carlos, 28942, Fuenlabrada, Spain: david.casillas@urjc.es

<sup>1</sup>By 'image', we always mean an RGB image taken by a regular camera.

highly robust deformable object registration and reconstruction.

SfT has been widely investigated with non-DNN approaches for almost two decades and only recently within the DNN framework. Non-DNN SfT methods fall into two broad categories. Methods in the first category compute registration before reconstruction with existing keypoint-based or dense matching methods [11, 12, 13]. They thus deal with the wide-baseline case but are tremendously limited by the catastrophic failure of registration, for many of the challenging object or imaging conditions (e.g., blur will typically defeat the extraction of keypoints). Methods in the second category compute registration and reconstruction simultaneously [8, 14, 15]. They proceed by numerical optimisation from an initial guess and hence only work in the short-baseline case. They may catastrophically fail for many of the challenging object or imaging conditions. Using the DNN framework to solve SfT is an attractive idea. The general concept is to learn a function that maps the input image to 3D deformation parameters [16, 17, 18]. This solves registration and reconstruction jointly, without iterative optimisation at run-time, and copes with the wide-baseline case. The attempts to develop DNN SfT methods are promising but also bear three important limitations. First, they are very restrictive with the object template, requiring regular rectangular meshes with a relatively small number of vertices (namely,  $73 \times 73$  in [17, 18] and  $10 \times 10$  in [16]). Second, they require labelled registration and reconstruction data for training. This relegates their training to only use synthetic data, affecting their accuracy in real conditions. Third, they require that training and inference are done with images coming from the same camera, which is a strong practical limitation. In spite of the progress brought by these works, there does not currently exist an SfT method capable of handling the wide-baseline case robustly for the challenging object and imaging conditions, densely and in real-time.

### 1.2. Related vision problems

SfT is closely related to some other vision problems, namely optical flow, scene flow, monocular depth reconstruction, pose estimation and Shape-from-Shading. However, SfT is unique in its own right as it has specific inputs-outputs and challenges. As a consequence, existing methods to these related problems either do not apply or cannot compete with specific SfT solutions.

**Optical flow.** Optical flow [14, 19, 20, 21, 22] solves registration between two consecutive images in a video. It differs from SfT in terms of its inputs (which are two images) and because it is only solved in the short-baseline configuration. Additionally, SfT involves reconstruction, while optical flow does not. Applications in AR, for instance, cannot be realised from optical flow only.

**Scene flow.** Scene flow solves registration between consecutive depth maps in an RGB-D video, obtained from an active sensor [23, 24] or stereo [25, 26]. It differs from SfT in terms of its inputs and because it is only solved in the short-baseline configuration. In SfT, the sensor at run-time is a regular camera, whose image cannot be fed to scene flow because of the missing depth channel. Additionally, DNN-based scene-flow meth-

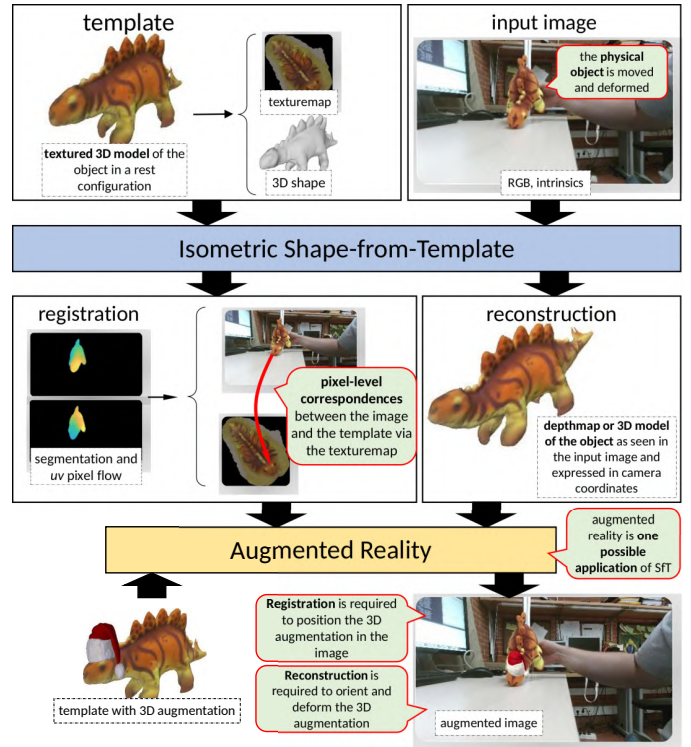


Figure 1: The principle of SfT and its application to augmented reality. Results obtained with DeepSfT proposal over a 2D manifold dinosaur template.

ods [27, 28, 29] have shown limited success using piecewise rigid motion and self-supervised approaches, with very similar limitations to optical flow methods.

**Monocular depth reconstruction.** Monocular depth reconstruction [30, 31, 32, 33, 34] infers depth from a single image for a scene category, such as rooms and road scenes. It is a hard problem with ambiguities due to the wide variability of objects, textures and shapes inside the scene. It is a reconstruction method, not involving registration, in contrast with SfT which involves both. Applications in augmented reality, for instance, cannot be realised from monocular depth reconstruction only.

**Pose estimation.** Pose estimation [35, 36, 37] computes the articulated pose of a person, typically defined by a 3D skeleton model, from a single image. In this sense it computes both registration and reconstruction, as the skeleton model is recovered in 3D. In a way, the skeleton model represents a category-level template. SfT differs from human pose estimation because its template is object-specific and deformation is of a much broader dimensionality. Specifically, a skeleton model typically has about 16 vertices, while an SfT template typically has several thousand vertices (e.g., 36256 vertices for the dinosaur template shown in figure 1).

**Shape-from-Shading.** Shape-from-Shading (SfS) is a reconstruction method which estimates depth and normal maps from an image of a textureless object. SfS does not generally consider a 3D object model and does not solve registration. The recent DNN methods [38, 39] have however solved SfS for object categories, such as pieces of cloth or paper. While [39] does not use an explicit object model, [38] fits a  $31 \times 31$  regular

rectangular mesh to guide reconstruction. It uses a depth sensor for labelling real training data. The experimental setup ensures that the object is easily segmented from a dark background and the illumination is controlled with at least three light sources. Both approaches stick to the classical SfS setting where the object must be mainly textureless, the scene illumination must produce significant shading, and the object must be segmented from the background. They are thus not applicable in the general AR context.

### 1.3. Summary of contributions

We present four contributions to advance the state-of-the-art in SfT within the DNN framework.

First, we propose DeepSfT, a novel DNN specifically tailored to SfT. Technically, DeepSfT is fully-convolutional and based on residual encoder-decoder structures with refining blocks. DeepSfT has an original architecture compared to previous DNN SfT methods [16, 17, 18]. First, in terms of its inputs: DeepSfT only takes the image as input, but not the template. This means that DeepSfT is *object-specific*, as the template is encoded in its weights at training time.

Second, in terms of its outputs: while previous methods output 3D vertices, DeepSfT produces a dense optical flow to represent registration and a dense depth map to represent reconstruction. If required, the full object shape is then obtained from a physics-based model a posteriori. These choices have important practical consequences. First, DeepSfT is an efficient network with real-time inference capability. Second, it is independent of the 3D object model representation, hence capable to exploit models with fine geometric details, complex topology and advanced material and illumination parameters. The computational cost of inference is independent of the number of parameters used to represent the object, such as the number of vertices with a mesh model. It thus solves the problem of limited template complexity of previous DNN SfT approaches [16, 17, 18].

Second, we propose a semi-supervised end-to-end training procedure, capable of training from synthetic and real data. Training from synthetic data is simple, by synthesising images from random quasi-isometric deformations of the template, whose registration and reconstruction parameters are readily available. Training from real data is however a difficult issue in SfT, yet is required to achieve good generalisation and to overcome the so-called *render gap*. Indeed, while the depth label can be obtained by acquiring data with a standard RGB-D sensor, the registration label cannot be obtained. Our procedure first trains DeepSfT from synthetic data with a combination of supervised loss functions that measures the error of the predicted registration and depth in different points of the network, forcing it to a coarse initial output. The refining blocks of the network are then trained from real data, with a combination of a supervised reconstruction loss function and a self-supervised registration loss function, based on image colour photo-consistency. Importantly, the quasi-isometric deformation of the object is learnt by DeepSfT because the training data, whether synthetic or real, exhibit quasi-isometric deformations of the object. Our training procedure thus strongly reduces the

requirement for fully-labelled data of previous DNN SfT approaches [16, 17, 18].

Third, we propose a solution to cope with multiple imaging geometries (caused by changing the intrinsics of the physical camera, typically by zooming in or out, or by using a different camera), at training and inference. A natural idea is to train the network with a variety of imaging geometries and possibly to also have it to output the calibration parameters. This is risky in at least two respects. First, quasi-isometric SfT has been shown to have a unique and well-posed solution in the general case for a calibrated camera, but not for an uncalibrated camera. Second, this will increase the size of the network and decrease its generalisability. Our proposal simply exploits the known camera geometry (intrinsics and distortion parameters) to warp the input image to a standard configuration. With this standardisation, DeepSfT can be trained to a single imaging geometry, and yet, handles any camera in any configuration. Our solution thus resolves the need of using the same camera to acquire or simulate training data and at inference time of previous DNN SfT approaches [16, 17, 18].

Fourth, we propose to combine DeepSfT with a physically inspired estimation procedure. This combination is intrinsically related to the choice of outputs we made for DeepSfT which, recall, is the optic flow field (registration) and the depth map (reconstruction). These outputs only solve SfT on the part of the object visible in the input image. For a 2D manifold object such as a shoe, there are always occluded parts, which, for some applications, may be important to register and reconstruct too. We propose to use the result of SfT for the visible part to solve for the occluded part based with the so-called As-Rigid-As-Possible (ARAP) prior. ARAP is a discrete approximation of the strain and bending energy of an isotropic elastic material that does not tear. It is therefore a physically inspired model for quasi-isometric deformations, which is widely used in graphics. We show how DeepSfT can be directly connected to this solution to recover the full object solution. Importantly, some applications such as AR do not require one to resolve the occluded object part, in which case this last step can be left aside and computation time saved. Our solution guarantees that the recovered occluded part of the object fulfills the physics-based constraints pertaining to the real world. In contrast, in previous DNN SfT approaches [16, 17, 18], these constraints are learnt by the network and thus only hold approximately on the solution.

Table 1 summarises the comparison between DeepSfT, other SfT methods, and related vision methods. We present quantitative and qualitative experimental results showing that DeepSfT outperforms the state-of-the-art in accuracy, robustness and computation time. These results include the wide-baseline case and severe imaging conditions, with strong occlusions, illumination changes, weak texture and blur. We release the data created by the authors through the following Kaggle dataset [40].

## 2. Previous Work

We first review the non-DNN SfT methods, which we call *classical SfT* methods, forming the vast majority of existing

|                  | Methods/problems               | Baseline | Accuracy | Number of vertices | Needs rectangular template | 2D manifold | Solves dense registration | Training needs full supervision | Real time | References                           |
|------------------|--------------------------------|----------|----------|--------------------|----------------------------|-------------|---------------------------|---------------------------------|-----------|--------------------------------------|
| Classical SfT    | Decoupled methods              | Wide     | Med      | Low                | No                         | Yes         | No                        | N/A                             | Yes       | [8, 41]                              |
|                  | Integrated methods             | Short    | High     | High               | No                         | Yes         | Yes                       | N/A                             | Yes       | [42, 43, 44]                         |
| DNN SfT          | Previous methods               | Wide     | High     | Low                | No                         | No          | Yes                       | Yes                             | Yes       | [17, 16, 18]                         |
|                  | <b>DeepSfT</b>                 | Wide     | High     | High               | No                         | Yes         | Yes                       | No                              | Yes       | Proposed                             |
| Related problems | Optical flow                   | Short    | High     | High               | Yes                        | Yes         | Yes                       | Yes                             | Yes       | [14, 19, 20]<br>[21, 22]             |
|                  | Scene flow                     | Short    | High     | High               | No                         | Yes         | Yes                       | Yes                             | Yes       | [23, 24, 25]<br>[27, 28, 29]<br>[26] |
|                  | Monocular depth reconstruction | N/A      | Low      | Low                | No                         | No          | Yes                       | No                              | Yes       | [33, 34, 30]<br>[31, 32]             |
|                  | Pose estimation                | Wide     | Med      | Low                | No                         | No          | Yes                       | Yes                             | Yes       | [36, 37, 35]                         |
|                  | Shape-from-Shading             | N/A      | Med      | High               | No                         | No          | Yes                       | Yes                             | Yes       | [38, 39]                             |

Table 1: Characteristics of existing SfT methods and other related problems with state-of-the-art solutions provided by DNNs. Existing SfT methods are divided into classical (non-DNN) and DNN methods.

work. We start with the *decoupled methods*, which solve registration and reconstruction as independent problems, and then we discuss the *integrated methods* that solve registration and reconstruction jointly. We finally review the DNN SfT methods. We have categorised state-of-the-art methods, their properties, and problems related to SfT in table 1.

### 2.1. Classical SfT decoupled methods

Decoupled methods first compute registration and then reconstruction as two independent and sequential stages [42, 43, 44]. Their main advantages are simplicity, problem decomposition, and to leverage existing mature registration approaches. However, they tend to produce sub-optimal solutions because they do not consider all physical constraints that connect reconstruction and registration. Decoupled methods typically solve wide-baseline registration with an existing method that is not specific to SfT, using feature-based matching with keypoints such as SIFT [45], with filtering to reduce the mismatches [11, 46]. These approaches inherit the advantages of wide-baseline registration: they can deal with individual images and strong deformation without requiring temporal consistency. However, they are fundamentally limited by feature-based registration, which fails when the object has a weak or repetitive texture, or when the imaging conditions are challenging (low image resolution, blur or strong viewpoint distortion). Furthermore, accurate results demand an expensive optimisation process at run-time. Because of these limitations, the existing real-time wide-baseline decoupled methods require simple objects with simple deformations, such as bending sheets of paper.

Various reconstruction methods have been considered in decoupled methods, and they can be classified according to the deformation model. The most popular deformation model is isometry, which approximately preserves geodesic distances. These methods follow one of three main strategies: *i*) using a convex relaxation of isometry called inextensibility [42, 6, 43, 44], *ii*) using local differential geometry [7, 9] and *iii*) minimising a global non-convex cost [44, 47]. Methods in *iii*) are the most accurate but also the most computationally expensive. They require an initial solution found using a method from *i*)

or *ii*). There also exist methods that relax isometry in an attempt to handle elastic deformations. These include the angle-preserving conformal model [7], or simple mechanical models with linear [48, 49] or non-linear elasticity [50, 51, 52, 53]. These models all require boundary conditions in the form of known 3D points, which is a fundamental limitation. The well-posedness of non-isometric methods remains an open research question.

### 2.2. Classical SfT integrated methods

Integrated methods compute both registration and reconstruction jointly. All existing methods are short-baseline, restricted to video data, and may work in real time [8, 14, 19]. They are based on the iterative minimisation of a non-convex cost that deforms the template in 3D so that its projection agrees with the image data. Some methods use keypoint correspondences that can be re-estimated during optimisation [8], and others use pixel-level information [14, 19] and a data cost based on template/image photo-consistency. These latter methods support dense solutions and resolve complex, high-frequency deformations. Their main limitations are two-fold. First, they break down with fast deformation or camera motion. Second, at run-time, they must solve an optimisation process that is highly non-convex and computationally demanding, requiring careful hand-crafted design and a correct balance of data and deformation constraints.

### 2.3. DNN SfT methods

Several DNN-based methods have been recently proposed [16, 17, 18]. These methods assume a flat template, described with a regular mesh. We refer to this special type of template as a *rectangular template*. They all use encoder-decoder neural architectures, and differ in the way the mesh vertex coordinates are parameterised and the learning strategy. [16] first solves registration by regressing many 2D belief maps (three per vertex), giving their likely 2D coordinates in the image. A depth estimation network is then used to reconstruct vertex depth coordinates. This strategy does not scale well to many

vertices, limiting its applicability, as shown by the reported experiments with  $10 \times 10$  vertices or fewer. [17, 18] use three-channel 2D outputs to parameterise the 3D coordinates of the mesh vertices. This strategy allows [17] and [18] to use a rectangular template with a greater number of vertices than [16], showing results with  $73 \times 73$  vertices in both cases. [17] use supervised learning that minimises the mean squared error between the network outputs and reconstruction labels with a synthetic training data base. [18] uses an adversarial learning approach, introducing a discriminator network. The methods of [16, 17, 18] share four important common weaknesses. First, they only work with rectangular templates, limiting their application to e.g. paper sheets or rectangular cloth sections. They cannot be used with non-rectangular templates, such as 2D manifold templates or objects with complex geometries like the shoe of figure 2. Second, they do not scale well for larger meshes, as it increases the network size. Third, the camera used for training and run-time must be the same. Fourth, they are *fully-supervised* methods, requiring fully labeled data. Due to the difficulty of obtaining labels with real data, they rely on simulated data. This strategy limits prediction accuracy in real images due to the render gap between simulated and real data [54]. For instance, [17, 18] use Blender [55] to create synthetic images of a deforming paper sheet or clothing. In all reported experiments the simulated images have controlled background and lighting conditions. In all these previous DNN methods, the experimental results with real data are mostly qualitative and with a controlled environment, to mitigate the render gap between the synthetic and real data.

In summary, the previous DNN SfT methods have shown that SfT can be learnt by a DNN. However, they have not been shown to work in real-world challenging conditions, and suffer four main limitations discussed above. Our proposed approach DeepSfT does not have these limitations, signifying a considerable step forward in SfT research and real-world application.

### 3. Methodology

#### 3.1. Scene geometry

**Template.** Figure 2 shows the geometric model of SfT, including the camera image and template deformation. The template is known and represented by a 3D surface  $\mathcal{T} \subset \mathbb{R}^3$  jointly with an appearance model, described as a *texture map*  $\mathcal{A}_{\mathcal{T}} = (\mathcal{A}, A)$ . The texture map consist of an  $\mathbb{R}^2$  domain  $\mathcal{A} \subset \mathbb{R}^2$  and a function  $A: \mathcal{A} \rightarrow (r, g, b)$  which maps it to the RGB space. The texture map domain  $\mathcal{A}$  is represented as a collection of flattened *texture charts*  $\mathcal{U}_i$  whose union covers the appearance of the whole template [56]. We use normalised texture coordinates for  $\mathcal{A}$ , drawn from the unit square. In our approach the template is not restricted to a specific topology, and can be *thin-shell* or *2D manifold*, without requiring modification to our DNN architecture. Our approach is also not restricted to a specific surface representation. In our experiments section we use mesh representations because of their generality, but this is not a requirement of the DNN. The bijective map between  $\mathcal{A}$  and  $\mathcal{T}$  is known and denoted by  $\Delta: \mathcal{A} \rightarrow \mathcal{T}$ .

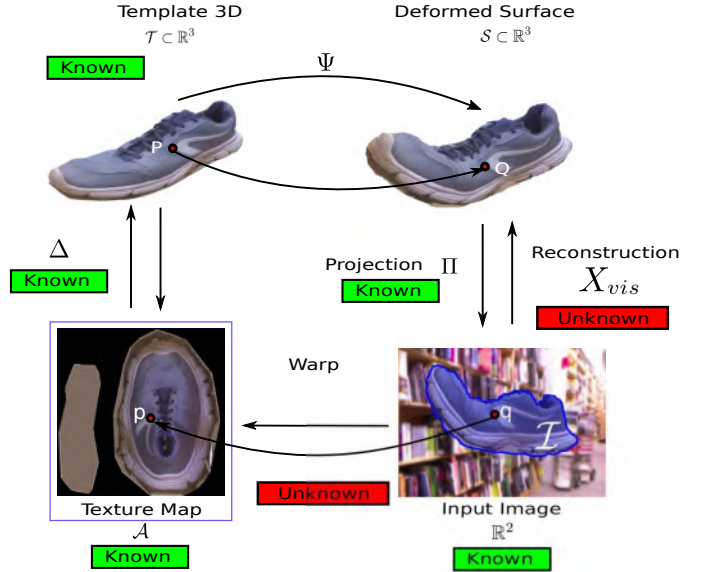


Figure 2: Geometric model of Shape from Template, showing the case of a shoe template. This model show that initially we know the 3D template  $\mathcal{T}$ , the Texture map  $\mathcal{A}$  of the object and the Input image used. What we want to find using an SfT approximation is the registration function  $\eta$  and the reconstruction of the visible part  $X_{vis}$ .

**Deformation.** We assume that  $\mathcal{T}$  is deformed with an unknown quasi-isometric map  $\Psi: \mathcal{T} \rightarrow \mathcal{S}$ , where  $\mathcal{S} \subset \mathbb{R}^3$  denotes the unknown deformed surface. Quasi-isometric maps permit mild extension and compression, common with many real world deformable objects.

**Camera projection.** The input image is modeled as a 3-channel colour intensity function  $I: \mathbb{R}^2 \rightarrow (r, g, b)$ , which is discretised into a regular grid of pixels. We model the camera with perspective projection:

$$(x, y, z) \mapsto \begin{pmatrix} x \\ y \\ z \end{pmatrix} = (u, v). \quad (1)$$

We assume that the camera is intrinsically calibrated: radial distortion, focal length and aspect ratio are all known parameters. This is a very common assumption in SfT. Hence,  $(u, v)$  are retinal coordinates that, without loss of generality, can be readily obtained from the image coordinates.

**Visible surface region and registration map.** The surface region that is visible in the camera image (unobstructed by self or external occlusion) is unknown and denoted by  $\mathcal{S}_{vis} \subset \mathcal{S}$ . This region projects onto the image plane to define an unknown 2D region  $\mathcal{I} \subset \mathbb{R}^2$ . We relate  $\mathcal{S}_{vis}$  and  $\mathcal{I}$  with a perspective embedding function  $X_{vis}: \mathcal{I} \rightarrow \mathcal{S}_{vis}$  with  $X_{vis}(u, v) = \rho(u, v)(u, v, 1)$  and where the unknown depth function  $\rho: \mathcal{I} \rightarrow \mathcal{S}_{vis}$  gives the depth of  $\mathcal{S}_{vis}$  in camera coordinates at each pixel in  $\mathcal{I}$ . In the absence of self-occlusions,  $\mathcal{S}_{vis} = \mathcal{S}$ . 2D manifold templates always induce self-occlusions. The unknown registration map,  $\eta: \mathcal{I} \rightarrow \mathcal{A}$  is an injective map that relates each point of  $\mathcal{I}$  to its corresponding point in  $\mathcal{A}$ .

#### 3.2. Object-specific approach

Our proposed DNN SfT solution DeepSfT estimates  $\rho(u, v)$ ,  $\eta(u, v)$  and  $\mathcal{I}$  directly from the input image  $I$ . DeepSfT is

object-specific, as the template information is encoded from the training data into the network weights, as [16]. In other words, the trained network’s weights ‘memorise’ the object shape. This reduces the difficulty of the learning problem, requiring a considerably lower amount of training data, and allows us to propose a compact architecture that runs in real time. The downside of an object-specific approach such as the one proposed here is that it cannot be directly ported to other templates, but needs to be retrained with them. Although this factor limits its usability in certain real cases that do have this requirement, there are other real use cases such as the 3D reconstruction of organs or industrial parts in predictive mechanics, in which this system would work correctly without the need for any retraining, since the template does not change. DeepSfT is much more accurate than object-generic methods [17, 18, 33, 34], which are not mature enough to solve SfT in challenging conditions, as we show in the experiments section. Importantly, we also provide a semi-supervised method to train DeepSfT without the need of manual labelling, which is a main limitation of the state-of-the-art. It combines synthetic data generated with Blender, with real data captured with a low-cost commercial RGB-D sensor. Generating data for a new template is thus done easily and can be implemented as a highly automatized process.

### 3.3. DNN architecture

We encode DeepSfT outputs as DNN functions taking  $I$  as input, which is resized to a canonical resolution of  $270 \times 480$  px:

$$(\hat{\rho}, \hat{\eta}) = \mathcal{D}(I, \theta_W), \quad (2)$$

where  $\theta_W$  are the network weights. We encode  $I$  in the network outputs  $\hat{\rho}$  and  $\hat{\eta}$  as follows:

$$\hat{\rho}(u, v) \approx \begin{cases} \rho(u, v) & (u, v) \in \mathcal{I} \\ -1 & \text{otherwise} \end{cases} \quad (3)$$

$$\hat{\eta}(u, v) \approx \begin{cases} \eta(u, v) & (u, v) \in \mathcal{I} \\ (-1, -1) & \text{otherwise.} \end{cases}$$

Figure 3 shows the proposed network architecture. It uses a cascaded structure divided into three principal blocks shown in figure 4. The Main Block is denoted as  $\mathcal{D}_M$ :

$$(\tilde{\rho}, \tilde{\eta}) = \mathcal{D}_M(I, \theta_M), \quad (4)$$

where  $\tilde{\rho}$  and  $\tilde{\eta}$  are estimates of the depth and registration maps and  $\theta_M$  contains the Main Block network weights. The Depth Refinement Block  $\mathcal{D}_D$  inputs  $I$ ,  $\tilde{\rho}$  and  $\tilde{\eta}$  and outputs a refined depth map  $\hat{\rho}$ :

$$\hat{\rho} = \mathcal{D}_D(I, \tilde{\rho}, \tilde{\eta}, \theta_D), \quad (5)$$

where  $\theta_D$  are the Depth Refinement Block network weights. The Registration Refinement Block  $\mathcal{D}_R$  inputs  $I$ ,  $\tilde{\rho}$  and  $\tilde{\eta}$  and outputs a refined registration map  $\hat{\eta}$ :

$$\hat{\eta} = \mathcal{D}_R(I, \tilde{\rho}, \tilde{\eta}, \theta_R), \quad (6)$$

where  $\theta_R$  are the Registration Refinement Block network weights. The weights of the three blocks define the network’s total weights  $\theta_W = (\theta_M, \theta_D, \theta_R)$ .

The refinement blocks play an important role to adapt the network to real data, as described in §4. The three blocks use identity, convolutional and deconvolutional residual feed-forwarding structures based on ResNet50 [57]. They use encoder-decoder architectures, very similar to those used in semantic segmentation [58]. Each block is composed of two unbalanced parallel branches with convolutional layers that propagate information to deeper layers, preserving high spatial frequencies.

Table 2 shows the layered decomposition of the Main Block. It first receives  $I$  and performs a first spatial reduction using a 2D convolutional layer with ReLu activation and Max Pooling. Then, a sequence of three convolutional and identity blocks is used to encode texture and depth information as deep features (figure 4). Image information from  $I$  is reduced to a com-

| Layer num            | Type                         | Output size               | Kernels/Activation |
|----------------------|------------------------------|---------------------------|--------------------|
| 1                    | Input                        | (270,480,3)               | –                  |
| 2                    | Convolution 2D               | (135,240,64)              | (7,7)              |
| 3                    | Batch Normalisation          | (135,240,64)              | –                  |
| 4                    | Activation                   | (135,240,64)              | Relu               |
| 5                    | Max Pooling 2D               | (45,80,64)                | (3,3)              |
| 6                    | Encoding Convolutional Block | (45,80,[64, 64, 256])     | (3,3)              |
| 7-8                  | Encoding identity Block x 2  | (45,80,[64, 64, 256])     | (3,3)              |
| 9                    | Encoding Convolutional Block | (23,40,[128, 128, 512])   | (3,3)              |
| 10-12                | Encoding identity Block x 3  | (23,40,[128, 128, 512])   | (3,3)              |
| 13                   | Encoding Convolutional Block | (12,20,[256, 256, 1024])  | (3,3)              |
| 14-16                | Encoding identity Block x 3  | (12,20,[256, 256, 1024])  | (3,3)              |
| 17-20                | Encoding identity Block x 3  | (12,20,[1024, 1024, 256]) | (3,3)              |
| 21                   | Decoding Convolutional Block | (24,40,[512, 512, 128])   | (3,3)              |
| 22                   | Cropping 2D                  | (23,39,128)               | (1,1)              |
| 23-25                | Encoding identity Block x 3  | (23,39,[512, 512, 128])   | (3,3)              |
| 26                   | Decoding Convolutional Block | (46,78,[256, 256, 64])    | (3,3)              |
| 27                   | Zero Padding                 | (46,80,64)                | (0,1)              |
| 28-29                | Encoding identity Block x 2  | (46,80,[256, 256, 64])    | (3,3)              |
| 30                   | Upsampling                   | (138,240,64)              | (3,3)              |
| 31                   | Cropping 2D                  | (136,240,64)              | (2,0)              |
| 32                   | Convolution 2D               | (136,240,64)              | (7,7)              |
| 33                   | Batch Normalisation          | (135,240,64)              | –                  |
| 34                   | Activation                   | (136,240,64)              | Relu               |
| 35                   | Upsampling                   | (272,480,64)              | (3,3)              |
| 36                   | Cropping 2D                  | (270,480,64)              | (2,0)              |
| 37                   | Convolution 2D               | (272,480,3)               | (3,3)              |
| 38                   | Activation                   | (270,480,1)               | Linear             |
| Number of parameters | 81 664 765                   |                           |                    |

Table 2: Main Block architecture showing from left to right columns, the number of each layer used, the type of layer used, the output size of the tensor that inference the layer and the own parameters of each layer.

pressed feature vector in a representation space of dimension  $12 \times 20 \times 1024$ . Decoding is performed with decoding blocks. These require upsampling layers to increase the dimensions of the input features before passing through the convolution layers, as shown in figure 4. Finally, the last layers have convolutional and cropping layers that adapt the output of the decoding block to the size of the output maps ( $270 \times 480 \times 3$ ). The first output channel provides the depth estimate  $\tilde{\rho}$ , and the last two output channels provide the registration estimate  $\tilde{\eta}$ .

The Depth Refinement and Registration Refinement Blocks share the same structure, shown in table 3, which is a reduced version of the Main Block using only the first two encoder and decoder blocks. The Depth Refinement Block takes as input the concatenation of  $I$ ,  $\tilde{\rho}$ , and  $\tilde{\eta}$  (6 channels) and it outputs  $\hat{\rho}$ . The Registration Refinement Block takes as input the concatenation



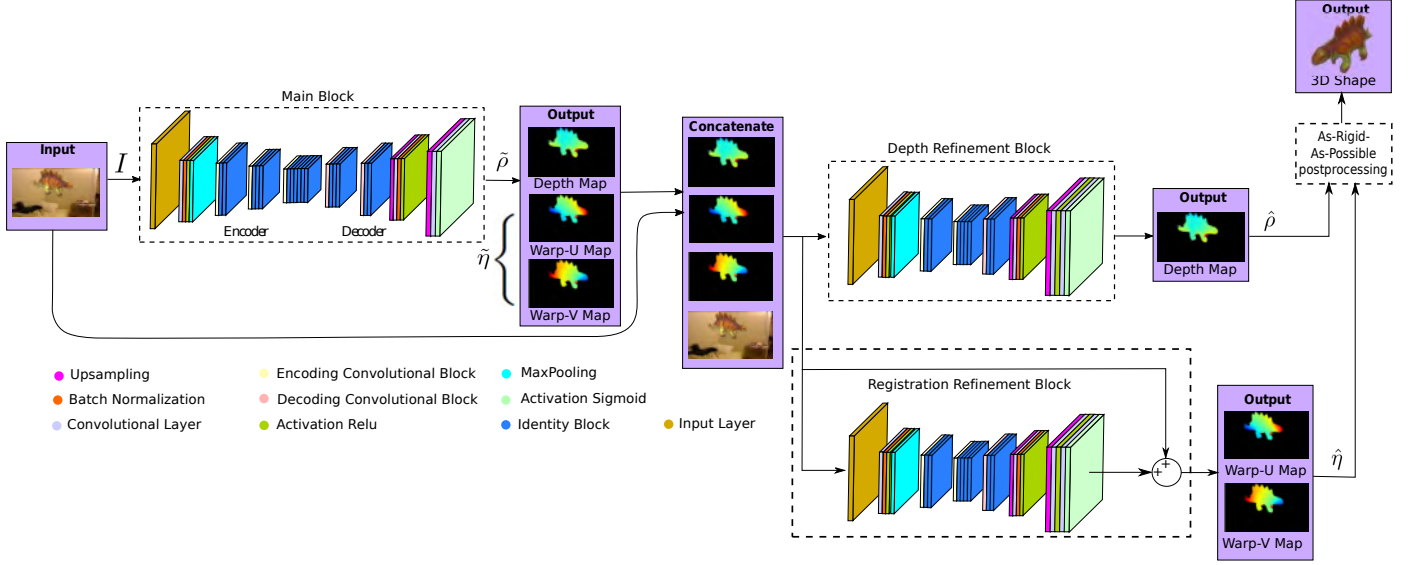


Figure 3: DeepSfT architecture. The proposed network architecture is composed of three principal blocks: the Main Block, the Depth Refinement Block and the Registration Refinement Block. Each block is an encoder-decoder designed for SfT. The Main Block receives an RGB input image  $I$  and outputs a first estimate of the registration and depth maps. The Depth and Registration Refinement Blocks improve the initial estimates, taking as input  $I$  and the Main Block outputs, and producing the final depth and registration maps.

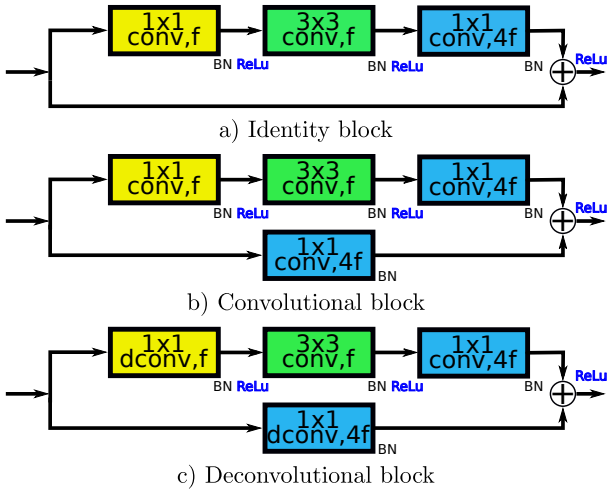


Figure 4: Identity, convolutional and deconvolutional residual blocks, similar to the ones used in residual architectures like [59].

of  $I$ ,  $\tilde{\rho}$ , and  $\tilde{\eta}$  (6 channels). Its output is added as an offset to  $\tilde{\eta}$  (last two channels) to produce  $\hat{\eta}$ .

### 3.4. Recovering occluded surface regions

Our DeepSfT network registers and reconstructs  $\mathcal{S}_{vis}$  by its outputs  $\hat{\rho}$  and  $\hat{\eta}$ . Due to self or external occlusions, always occurring with 2D manifold templates, the hidden surface part  $\mathcal{S}_h = \mathcal{S} \setminus \mathcal{S}_{vis}$  can be large and important. However, learning to infer  $\mathcal{S}_h$  from a single image is a very ill-posed problem due to ambiguities, and can be very difficult to train with real data. We propose a post-processing step to recover  $\mathcal{S}_h$  based on minimising the As-Rigid-As-Possible (ARAP) cost, widely used in graphics and mesh processing [60]. ARAP is also the most nat-

| Layer num            | Type                         | Output size             | Kernels/Activation |
|----------------------|------------------------------|-------------------------|--------------------|
| 1                    | Input                        | (270,480,6)             | -                  |
| 2                    | Convolution 2D               | (135,240,64)            | (7,7)              |
| 3                    | Batch Normalisation          | (135,240,64)            | -                  |
| 4                    | Activation                   | (135,240,64)            | Relu               |
| 5                    | Max Pooling 2D               | (45,80,64)              | (3,3)              |
| 6                    | Encoding Convolutional Block | (45,80,[64, 64, 256])   | (3,3)              |
| 7-8                  | Encoding identity Block x 2  | (45,80,[64, 64, 256])   | (3,3)              |
| 9                    | Encoding Convolutional Block | (23,40,[128, 128, 512]) | (3,3)              |
| 10-13                | Encoding identity Block x 4  | (23,40,[128, 128, 512]) | (3,3)              |
| 14                   | Decoding Convolutional Block | (46,80,[512, 512, 128]) | (3,3)              |
| 15-16                | Encoding identity Block x 2  | (46,80,[512, 512, 128]) | (3,3)              |
| 17                   | Upsampling                   | (92,160,128)            | (2,2)              |
| 18                   | Cropping 2D                  | (92,160,128)            | (2,0)              |
| 19                   | Convolution 2D               | (90,160,64)             | (3,3)              |
| 20                   | Batch Normalisation          | (90,160,64)             | -                  |
| 21                   | Activation                   | (90,160,64)             | Relu               |
| 22                   | Upsampling                   | (270, 480, 64)          | (3,3)              |
| 23                   | Convolution 2D               | (270, 480, 32)          | (3,3)              |
| 24                   | Activation                   | (270, 480, 32)          | Relu               |
| 25                   | Convolution 2D               | (272,480,1)             | (3,3)              |
| 26                   | Activation                   | (270,480,1)             | Linear             |
| Number of parameters |                              | 13 618 689              |                    |

Table 3: Depth and Registration Refinement Block architectures showing from left to right columns, the number of each layer used, the type of layer used, the output size of the tensor that inference the layer and the own parameters of each layer.

| Sequence | Samples | Train | Test |
|----------|---------|-------|------|
| DS1S     | 60000   | 47000 | 5000 |
| DS2S     | 60000   | 47000 | 5000 |
| DS3S     | 60000   | 47000 | 5000 |
| DS4S     | 60000   | 47000 | 5000 |
| DS1R     | 2116    | 1884  | 232  |
| DS2R     | 3100    | 2728  | 373  |
| DS3R     | 4800    | 3500  | 1300 |
| DS4R     | 4200    | 3650  | 550  |
| DS5R     | 193     | 143   | 50   |

Table 4: Train and test split for each image sequence. ‘S’ stands for synthetic generated with Blender and ‘R’ stands for real generated with Kinect V2.

ural prior for quasi-isometric templates [61, 62] and it does not require additional learning.

Unlike the DNN, which is independent of surface representation, the shape completion process requires the template to be represented as a triangular mesh. We use  $\mathcal{M}_S$  and  $\mathcal{M}_T$  to represent the deformed and rest template meshes respectively. These have 3D vertices  $\mathcal{V}_S = \{\mathbf{p}_1, \dots, \mathbf{p}_N\}$  and  $\mathcal{V}_T = \{\mathbf{q}_1, \dots, \mathbf{q}_N\}$  respectively. The objective of ARAP shape completion is to recover  $\mathcal{V}_S$  (and hence  $\mathcal{S}$ ) from  $\mathcal{S}_{vis}$  and  $\mathcal{V}_T$  by solving the following optimisation problem:

$$\mathcal{V}_S = \arg \min_{\mathcal{V}_S} E(\mathcal{V}_S), \quad (7)$$

where:

$$E = E_d(\mathcal{V}_S, \mathcal{S}_{vis}) + \lambda_a E_a(\mathcal{V}_S, \mathcal{V}_T) + \lambda_s E_s(\mathcal{V}_S, \mathcal{V}_T). \quad (8)$$

$E_d$  is the data term. It uses the Euclidean norm between the set of visible vertices in  $\mathcal{V}_S$  and their corresponding 3D coordinates in  $\mathcal{S}_{vis}$ , as produced by DeepSfT.  $E_a$  is the ARAP prior [62], that encourages the deformed mesh to be isometric with respect to the rest mesh. Finally,  $E_s$  is a smoothing term that penalizes large deviations in the local curvature of  $\mathcal{S}$  with respect to the template. The hyperparameters  $\lambda_a = 20$  and  $\lambda_s = 0.005$  control the influence of the ARAP and smoothing terms. We set them to a fixed value selected experimentally. We implement  $E_a$  and  $E_s$  following [62] and optimise  $E$  with Gauss-Newton, which typically converges in fewer than 10 iterations. This can be implemented easily on a GPU enabled device for real-time shape completion.

## 4. DNN Training

### 4.1. Training process overview

For a given template, we create a quasi-photorealistic synthetic dataset using rendering software. This process is described in detail in §5.1.1 and it is used to train DeepSfT with supervised learning. We also record a much smaller dataset with a real RGB-D camera capturing some representative deformations and poses of the object. We emphasize that the RGB-D

camera provides only depth labels and not registration labels, so it cannot be used for supervised learning of the registration.

Using both simulated and real data, we train DeepSfT in three steps. In the first step we use the synthetic data to train the Main, Depth Refinement and Registration Refinement Blocks end-to-end. In the second step we refine the Depth Refinement Block weights using real training data. In the third step we refine the Registration Refinement Block weights using real training data with unsupervised learning, by minimising a loss function that enforces the registered template to be photometrically consistent with the input images.

DeepSfT has been implemented in Keras/Tensorflow [63]. We have observed that Stochastic Gradient Descent (SGD) achieves better generalisation results when fine tuning the network with real data while Adaptive Moment Estimation (ADAM) [64] performs better when training from scratch. We thus use ADAM in the first step and SGD in the second and third steps. Mixing ADAM and SGD is common practice [65, 66].

### 4.2. Training step 1: initial global training

The Main, Depth Refinement and Registration Refinement Blocks are trained end-to-end with the following supervised loss function:

$$\begin{aligned} \mathcal{L}_1(\theta_W) = & \frac{1}{2} \sum_{i=1}^M \|\hat{\eta}_i - \eta_i\|_F^2 + \sum_{i=1}^M \|\hat{\rho}_i - \rho_i\|_F^2 + \\ & \frac{1}{2} \sum_{i=1}^M \|\tilde{\eta}_i - \eta_i\|_F^2 + \sum_{i=1}^M \|\tilde{\rho}_i - \rho_i\|_F^2, \end{aligned} \quad (9)$$

where  $\hat{\rho}_i$  and  $\hat{\eta}_i$  are the estimated depth and registration maps, and  $\tilde{\rho}_i$  and  $\tilde{\eta}_i$  are the outputs from the Main Block. The terms  $\rho_i$  and  $\eta_i$  are the label maps, and  $M$  is the number of synthetic images. The symbol  $\|\cdot\|_F$  is the Frobenius norm. We use ADAM optimisation with a learning rate of  $10^{-3}$  and parameters  $\beta_1 = \beta_2 = 0.9$ . Training is fixed to 40 epochs with a batch size of 7, taking approximately 12 hours in a single GPU workstation (Nvidia GTX1080). The weights are initialised with random uniform sampling [67].

### 4.3. Training step 2: Depth Refinement Block fine tuning

We fine-tune the Depth Refinement Block weights using real data while freezing the weights  $\theta_M$  of the Main Block. This step is crucial to adapt the network to handle the render gap and to cope with real illumination conditions, camera response and color balance. In this step a different loss function  $\mathcal{L}_2$  is used, which combines a supervised loss for the Depth Refinement Block and a spatial regulariser:

$$\mathcal{L}_2(\theta_D) = \sum_{i=1}^{M'} \|\hat{\rho}_i - \rho_i\|_F^2 + \lambda \sum_{i=1}^{M'} \|\nabla \hat{\rho}_i\|_1^2, \quad (10)$$

where  $M'$  is the number of real training images. We include total variation regularisation [68] to mitigate the effect of noise in the depth labels while preserving edges and details [69]. The hyperparameter  $\lambda$  is set to  $10^{-9}$  in all experiments and chosen

empirically. We train with SGD and a small and fixed learning rate of  $10^{-5}$ . We train for 10 epochs with a batch size of 7. Having both a low learning rate and a reduced number of epochs allows us to adapt our network to real data while avoiding overfitting.

#### 4.4. Training step 3: Registration Refinement Block fine tuning

In this step we use a property of SfT, which is that the input image can be synthesised from the registration solution, by warping the template texture map. We propose a self-supervised fine tuning algorithm for the Registration Refinement Block, based on minimising a photo-consistency loss that computes the error between the synthesised image and the input image. For each input image  $I_i$ , the corresponding synthesised image  $I'_i$  is computed as follows:

$$I'_i(u, v) = \begin{cases} A(\hat{\eta}_i(u, v)) & (u, v) \in \hat{I} \\ 0 & \text{otherwise,} \end{cases} \quad (11)$$

where  $\hat{I}(u, v) \triangleq \hat{\rho}(u, v) > -1$  is the object segmentation obtained from the Depth Refinement Block. The computation of equation (11) is first-order differentiable in the Registration Refinement Block network weights  $\theta_R$ , as described in Appendix A.

The unsupervised loss function  $\mathcal{L}_u$  forces the network to produce synthesised images that are photometrically similar to the input images. The loss involves the registration map  $\hat{\eta}_i$  computed by the Registration Refinement Block, each input image  $I_i$  and their corresponding synthesised image  $I'_i$ , and is defined as follows:

$$\mathcal{L}_u(\theta_R) = \sum_{i=1}^{M'} \sum_{(u,v) \in \mathcal{I}} \chi \left( (I'_i(u, v) - I_i(u, v))^2 \right) + \mu \sum_{i=1}^{M'} \sum_{(u,v) \in \mathcal{I}} \chi \left( (I_i^{\downarrow}(u, v) - I'_i(u, v))^2 \right) + \lambda \sum_{i=1}^{M'} \|\nabla \hat{\eta}_i\|_1^2. \quad (12)$$

where  $M'$  is the number of training images,  $\chi(x)$  is an M-estimator, and images  $I_i^{\downarrow}$  and  $I'_i$  are downsized versions of  $I_i$  and  $I'_i$  respectively by a factor 2. These are used to include losses at two spatial scales, which improves convergence similarly to image pyramids used in unsupervised optical flow [70]. The loss is controlled by a hyperparameter weight  $\mu$ , fixed to 0.5 in all our experiments. To handle illumination changes and shading effects that violate photo-consistency, we use the Cauchy M-estimator:

$$\chi(x) = \frac{c^2}{2} \log \left( 1 + (x/c)^2 \right), \quad (13)$$

with  $c = 4$  as default. We also include a total variation regularisation term in the loss that imposes smoothness in the registration output while preserving discontinuities. This term is usually included in optical flow methods [70] to improve convergence. The hyperparameter  $\lambda$  is set empirically to  $10^{-9}$  in all experiments. We optimise  $\mathcal{L}_u$  using SGD with momentum. We found that optimisers with an adaptive step, such as ADAM,

or large learning rates cause convergence problems when minimising  $\mathcal{L}_u$ . We use an initial learning rate of  $10^{-5}$  and a decay of  $10^{-9}$ . The Registration Refinement Block is trained for 10 epochs with a batch size of 6.

#### 4.5. Handling different camera intrinsics

We train DeepSfT with images generated by a camera with fixed intrinsics (called the training camera), which may potentially have different intrinsics to the test camera. Once the network is trained, we cannot immediately input images from the test camera into the network because its weights are trained specifically for the intrinsics of the training camera. A possible solution to this problem is to train DeepSfT to handle varying intrinsics. However, this is very challenging and a potential source of ambiguities between intrinsics and image deformations. We have been able to generalise DeepSfT to work with a different camera at test time without any need to retrain the network weights. This has not been achieved with other DNN-based SfT methods and it significantly broadens our applicability for real-world use. We propose to handle this by adapting the test camera’s effective intrinsics to match the training camera. Because the object’s depth within the training set varies (and so do the perspective effects), we can emulate testing on the training camera by applying a known affine transform to images from the test camera. The affine transform is the matrix  $A = K_{train} K_{test}^{-1}$ , where  $K_{train}$  and  $K_{test}$  are the intrinsic matrices of the training and test cameras respectively. The transformed test image is then clipped about its principal point and zero padded, if necessary, to obtain the canonical resolution of  $270 \times 480$  (the input image size of DeepSfT). It is important to highlight that the DeepSfT output maps correspond with the standardized intrinsics (i.e. the intrinsics used for training). For example, in the case of the depth map, the 3D surface is then estimated from this depth-map using the standardized intrinsics. If desired, the depth-map can be converted to an equivalent depth-map associated to the test intrinsics by the 2D affine transform  $A^{-1}$ .

## 5. Experimental Results

### 5.1. Datasets

#### 5.1.1. Templates

We have tested DeepSfT with 5 objects represented by 3 thin-shell and 2 2D manifold templates shown in table 5. We refer to these as DS1 to DS5. DS1 models an A4 paper sheet with very poor texture. DS2 models an A4 paper sheet with a richer texture and DS5 models an A4 paper sheet from a well-known public dataset [71]. DS3 is a 2D manifold model of a soft toy and DS4 is a 2D manifold model of an adult sneaker. DS1, DS2 and DS5 can be modelled with a rectangular template, however DS3 and DS4 cannot. They were built with triangular meshes using dense SfM (Agisoft Photoscan [72]). We emphasize that no previous DNN-based work has been able to solve SfT for 2D manifold templates like DS3 and DS4 in the wide-baseline setting.

### 5.1.2. Synthetic datasets

For each template a synthetic dataset was constructed by deforming the template with random quasi-isometric deformations and rendering the deformed template with fixed camera intrinsics and random viewpoints. We used *Blender* [55], which includes a physics-based simulation engine to simulate deformations with different degrees of stiffness using position-based dynamics. For DS1, DS2 and DS5 (rectangular templates) we simulated continuous videos with a high stiffness term and randomly located 3D anchor points. We applied tensile and compressive forces in randomised 3D directions. The simulation parameters are given in the supplementary material. For DS3 and DS4 (2D manifold templates) we used rig-based deformations with hand-crafted rigs. We generated independent deformations for each image using random joint angles.

For each deformation we rendered an image with a random camera pose (random rotation around the camera’s optical axes with angle variations in the interval  $[-\frac{\pi}{4}, \frac{\pi}{4}]$  radians and random translations in the intervals  $t_x \in [-150, 150]$  mm,  $t_y \in [-150, 150]$  mm and  $t_z \in [100, 600]$  mm). A distant light model was used with illumination angles parameterised by spherical coordinates that was drawn randomly in the interval  $[-\frac{\pi}{18}, \frac{\pi}{18}]$  radians around the camera’s optical axis. The diffuse surface reflectance component was modelled as Lambertian and the specular component was modelled with Blender’s Cook-Torrence model. We generated brightness variations by a random gain in the range  $[0.9, 1.1]$ . We randomly changed the image background with images from [73]. To simulate occlusions, we randomly introduced a maximum of 4 synthetically generated circles of constant random color in each image with variable diameter in the range  $[1, 10]$  px at random locations. In total, each dataset consists of 60000 RGB images with labelled depth and registration maps. These were standardised to a canonical resolution of  $270 \times 480$  px.

### 5.1.3. Real datasets

Real datasets of each object were recorded with Microsoft Kinect v2 with deformations caused by hand manipulation, as shown in table 5. Videos for DS1, DS2, DS3 and DS4 were recorded by us and the video for DS5 was provided in the public dataset (192 frames). The recorded depth maps were aligned with the RGB images using the extrinsic parameters and down-sized to  $270 \times 480$  px. Note that these RGB-D videos do not provide labelled registration data.

### 5.1.4. Training/testing data splits

We evaluate DeepSfT in terms of reconstruction and registration errors with synthetic and real test data. Synthetic test data were generated using the same process as the synthetic training data (§5.1.2), using random configurations not present in the training data. Real test data were generated using the same process as the real training data, using new videos, consisting of new viewpoints and object manipulations not present in the training data. We also generated test data using two new real cameras: an Intel Realsense D435[74] (an RGB-D camera for quantitative reconstruction evaluation) and a Gopro Hero

V3[75] (an RGB camera for qualitative evaluation). Table 6 shows their respective camera intrinsics.

Table 4 shows the train and test split for all real datasets. When testing DeepSfT with synthetic data, results from the Main Block are evaluated. When testing with real data, results from the Depth Refinement Block and Registration Refinement Block are evaluated.






### 5.2. Compared methods and evaluation metrics

We compare DeepSfT with two classical state-of-the-art SfT methods. The first is an isometric SfT method [9] with public code, referred to as CH17. We provide this method with two types of registration: CH17+GTR uses Ground-truth Registration (indicating its best possible performance independent of the registration method) and CH17+DOF uses a state-of-the-art Dense Optical Flow registration method [22]. In the latter case we generate registration only for image sequences using frame-to-frame tracking. We also add to these two methods a final refinement step based on minimising a statistically optimal non-convex cost function with Levenberg-Marquardt [7]. We refer to the refined solutions as CH17R+GTR and CH17R+DOF. The second classical SfT method we test is [76] with public code, referred to as NGO15. In addition to this, we do not compare with [60] because its performance in terms of surface reconstruction error registration and surface is similar to [9] with which we compare our proposal. [60] is a classic 2D manifold SfT method that estimates volume deformation and has some strong limitations, like not achieving real-time, being feature-based and requiring densely textured surfaces with distinct texture features.

We compare DeepSfT with three DNN-based methods. The first is a naïve application of the popular ResNet architecture [57] to solve SfT, referred to as R50F. The reason to include the ResNet model was to compare our fully convolutional encoder-decoder architecture against a combination of an encoder and a fully connected model. This comparison demonstrates that the proposed architecture outperforms the classic encoder-fully connected architectures such as ResNet. We adapt ResNet by removing the final two layers and introduce a dense layer with 200 neurons and a final dense layer with a 3-channel output (for depth and registration maps) of the same size as the input image. We trained R50F with exactly the same training data as DeepSfT and with real-data fine tuning. Fine-tuning was implemented by optimising the depth loss, using the same optimiser and learning rate as we used for DeepSfT. The second DNN method is [17], which we refer to as HDM-net. This is tested only with rectangular templates (DS1 and DS2) because it only handles textureless or weakly textured rectangular templates. We carefully re-implemented [17], requiring an adaptation of the image input size and the mesh size so that it matched the size of the template meshes. The third DNN method is [18] using the authors’ code, which we refer to as IsMo-GAN, that is also applied only to DS1 and DS2 as it requires a rectangular template.

We evaluate reconstruction error using the Root Mean Square Error (RMSE) in millimeters. We also use RMSE to evaluate the registration accuracy in pixels. The evaluation of regis-

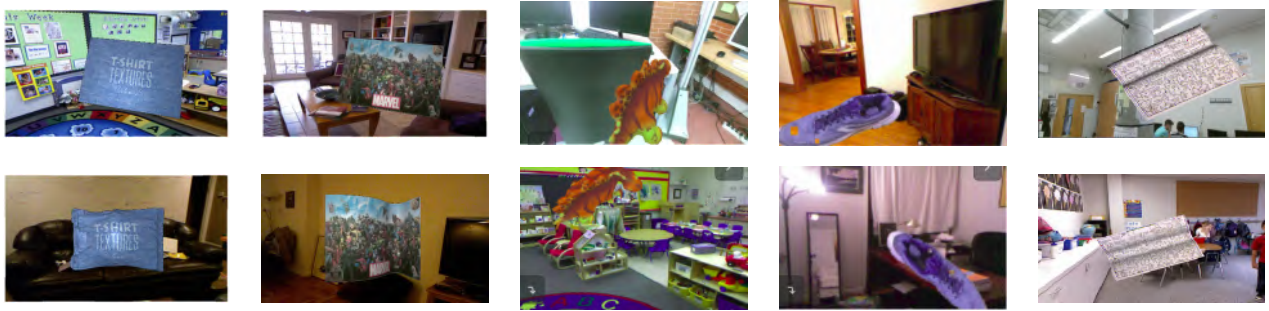
Template 3D shapes

| DS1   | DS2   | DS3   | DS4  | DS5   |
|---|---|---|--|---|
|  |  |  |  |  |
| Mesh Faces=1521   | Mesh Faces=1521   | Mesh Faces=36256  | Mesh Faces=5212  | Mesh Faces=1521   |

Template texture maps



Synthetic images



Real images

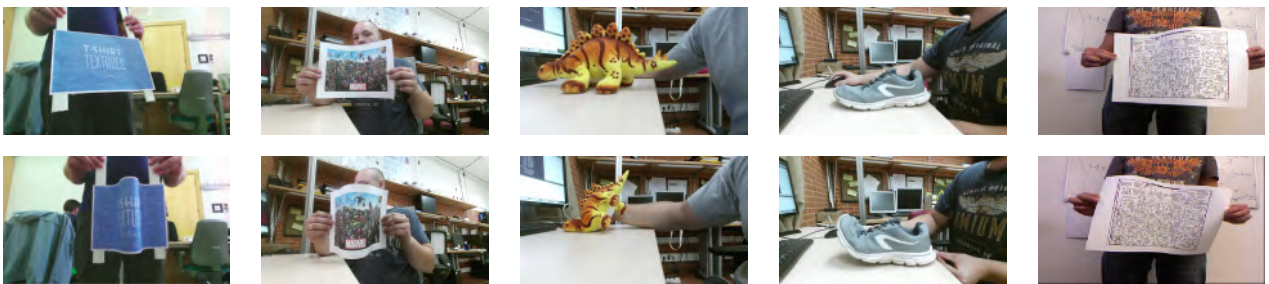


Table 5: Visualization of templates and input images. Rows 1 and 2 show the five templates DS1, DS2, DS3, DS4 and DS5. Rows 3 and 4 show example renders with simulated deformations. Rows 5 and 6 show real deformations of the physical objects.

| Camera               | Resolution  | $f_u$  | $f_v$  | $c_u$ | $c_v$ |
|----------------------|-------------|--------|--------|-------|-------|
| Kinect V2            | 1920 × 1080 | 1057.8 | 1064.0 | 947.6 | 530.4 |
| Intel Realsense D435 | 1270 × 720  | 915.5  | 915.5  | 645.5 | 366.3 |
| Gopro Hero V3        | 1920 × 1080 | 1686.8 | 1694.2 | 952.8 | 563.5 |

Table 6: Camera intrinsics of the different real cameras used in our experiments. We use Kinect V2 for training and all three cameras for testing.

tration accuracy is notoriously difficult with real data because there is no way to obtain reliable ground-truth. We propose to use as a proxy for the ground-truth the output from a state-of-the-art dense trajectory optical flow method DOF [22]. We only make this quantitative evaluation for videos, for which DOF can reliably compute registration. We manually selected sequences where DOF produces stable tracks. The use of DOF or any other optical flow method as a registration baseline can introduce bias. However, obtaining registration results with a wide-baseline method such as DeepSfT that are comparable with DOF is considered a very strong result for a wide-baseline method.

### 5.3. Evaluation with rectangular templates

We show in tables 7, 11 and 12 quantitative and qualitative results obtained with rectangular templates and synthetic test datasets, denoted by DS1S and DS2S, and real test datasets, denoted by DS1R, DS2R and DS5R. In terms of reconstruction error, DeepSfT is considerably better than the other methods, both in synthetic test data, where the RMSE remains below 2 mm, and for real test data, where the RMSE is below 10 mm. Kinect V2 has an uncertainty of about 10 mm at a distance of one meter, which partially explains the higher error for real data. The second and third best methods are IsMo-GAN and R50F respectively, also DNN-based. However, their errors are far worse compared to DeepSfT. CH17 obtains reasonable results when it is provided with ground-truth registration (CH17-GTR and CH17R-GTR). However, the performance is considerably worse when real registration is provided by DOF (CH17-DOF and CH17R-DOF). NGO15 obtains the worst result on DS1 and the second worst result on DS2. This was expected because we evaluate this algorithm in a wide-baseline setting and, as mentioned by the authors, this method was designed to work only for small deformations (small-baseline).

In terms of registration error, DeepSfT also has the best results both for synthetic test data, where ground-truth registration is available, and real test data, where DOF is used as the ground-truth proxy. In all cases DeepSfT has a mean registration RMSE of approximately 2 px. The performance of R50F is competitive with DOF, with registration RMSE of approximately 5 px.

### 5.4. Evaluation with 2D manifold templates

The quantitative and qualitative results of the experiments for the 2D manifold templates DS3 and DS4 are provided in tables 9, 11 and 12, with both synthetic test data, denoted by DS3S and DS4S, and real test data, denoted by DS4R and DS4R. Recall that the test datasets consist of unorganised images, unlike

DS1, DS2 and DS5, and it is thus impossible to estimate registration reliably with DOF. Therefore we only compute registration error with synthetic data (DS3S and DS4S). CH17+GTR and CH17R+GTR are tested only on DS3S and DS4S, because these are the only datasets they can handle.

The results show a similar trend as with the rectangular template datasets: DeepSfT outperforms the other methods in terms of reconstruction error, with an RMSE of the order of millimeters, and in registration with an RMSE close to 2 px. The second best method is R50F, although its results are significantly worse than DeepSfT is. The results of CH17 and its variants are very poor. This may be because CH17 is not well adapted for 2D manifold objects with stronger non-isometric deformation.

We show in table 13 qualitative reconstruction results obtained with DS1R, DS3R and DS4R with real images. We observe that shapes recovered with DeepSfT are similar to ground-truth obtained with the RGB-D camera and have no 'outliers' in their boundaries, in contrast to the RGB-D camera ground-truth. We observe that the error is larger near self-occlusion boundaries.

### 5.5. Evaluation of ARAP shape completion

We show in table 13 example results before and after ARAP shape completion using DS1, DS3 and DS4 arranged in three rows. The table shows from left to right a representative input image, ground-truth provided by Kinect V2, registration and reconstruction outputs from DeepSfT as point clouds, outputs as coloured point clouds, and lastly the 3D shape completion results. The reconstruction errors are evaluated across the visible surface regions before and after shape completion and denoted by DNN RMSE and ARAP RMSE respectively. We can see that these errors are very similar, which implies that the benefit of shape completion is only to recover the occluded regions. It does not improve significantly the reconstruction of the visible regions compared to the DNN output. Quantitatively the completed 3D shapes look compelling and representative of the true object deformations.

### 5.6. Evaluation of test camera generalisation

Using the technique described in §4.5, we test performance with three different real test cameras (Microsoft Kinect V2: same as for training, Intel Realsense D435 and Gopro Hero V3). Table 14 gives reconstruction errors with the Kinect and Realsense cameras. For the Gopro Hero V3 (an RGB camera) we show qualitative results. Quantitatively, the reconstruction errors with the Kinect and Realsense cameras are quite similar. This is an important point and clearly demonstrates the ability of DeepSfT to generalise well to images taken with a different test camera. Furthermore, DeepSfT copes with images from another camera even if the focal lengths are significantly different, as indicated qualitatively with the GoPro camera. We emphasize that this is the first time SfT has been solved with different train/test cameras with a DNN. This has a big practical benefit, because we are not limited to using the same camera at test time.

| Sequence | Samples | Registration RMSE (px) |      |             | Reconstruction RMSE (mm) |          |           |           |       |         |          |       |             |
|----------|---------|------------------------|------|-------------|--------------------------|----------|-----------|-----------|-------|---------|----------|-------|-------------|
|          |         | DOF                    | R50F | DeepSfT     | CH17+GTR                 | CH17+DOF | CH17R+GTR | CH17R+DOF | NGO15 | HDM-net | IsMo-GAN | R50F  | DeepSfT     |
| DS1S     | 5000    | 4.63                   | 6.69 | <b>1.87</b> | 6.89                     | 15.60    | 8.27      | 15.41     | 18.77 | 10.80   | 7.32     | 7.99  | <b>1.68</b> |
| DS2S     | 5000    | 5.91                   | 6.13 | <b>1.34</b> | 6.89                     | 28.26    | 8.27      | 28.04     | 21.32 | 9.92    | 6.94     | 7.75  | <b>1.63</b> |
| DS1R     | 232     | -                      | 5.02 | <b>2.32</b> | -                        | 38.12    | -         | 34.24     | -     | -       | -        | 17.53 | <b>9.51</b> |
| DS2R     | 373     | -                      | 4.13 | <b>1.53</b> | -                        | 27.31    | -         | 25.24     | -     | -       | -        | 14.45 | <b>7.37</b> |
| DSSR     | 50      | -                      | 6.33 | <b>2.74</b> | -                        | 22.57    | -         | 19.42     | 32.3  | -       | -        | 16.30 | <b>6.97</b> |

Table 7: Quantitative evaluation on synthetic and real test data with rectangular templates (DS1S, DS2S, DS1R, DS2R and DSSR).

### 5.7. Evaluation of light and occlusion resistance

We show that DeepSfT is resistant to light changes and significant occlusions in table 15. The first two rows of the table show representative examples of scenes with external and self occlusions. DeepSfT is able to cope with them, accurately detecting the occlusion boundaries. The third and fourth rows show examples of scenes with illumination that produce significant shading variations. DeepSfT shows good resistance to these variations.

### 5.8. Failure modes

There are some instances where DeepSfT fails, shown in the final two rows of table 15. There are general failure modes of SfT (very strong occlusions and illumination changes), for which all methods will fail at some point. There are also failure modes specific to learning-based approaches (excessive deformations that are not represented in the training set). However, recall that wide-baseline methods like DeepSfT can recover easily from failures with video inputs because they process each image independently, unlike short-baseline methods. Therefore failure for some frames in a video does not prevent successful reconstruction and registration in the later frames.

### 5.9. Ablation studies

#### 5.9.1. The benefit of Total Variation regularisation

We included total variation smoothness during fine tuning of the Depth Refinement Block and the Registration Refinement Block. In the Depth Refinement Block, the main objective of this term is to alleviate the effect of noise and outliers in depth data used as ground-truth in equation (10). In the Registration Refinement Block, it is used to improve convergence of the self-supervised algorithm, based on minimising the photometric error in equation (12). We investigate the effect of this term in both depth and registration accuracy, when testing with real data. We show in table 10 the quantitative results obtained with all the templates DS1, DS2, DS3, DS4 and DS5. As can be seen, the Total Variation regularisation improves the reconstruction and registration errors in all the cases, especially for the DS2 template.

#### 5.9.2. The benefit of depth refinement

We evaluate the influence of the Depth Refinement Block and its results in terms of depth RMSE. We show these errors in table 16 where the RMSE obtained using only the Main Block of DeepSfT is compared to the RMSE obtained by the Depth Refinement Block. Recall that the Main Block has been trained exclusively using synthetic data whereas the Depth Refinement

Block has been fine-tuned with real data. It can be clearly seen that the Depth Refinement Block RMSE is much lower compared to the Main Block RMSE. Recall that the Main Block provides a first approximation of the depth map but, due to the render gap between synthetic and real data, this approximation is not highly accurate. The Refinement Block refines this approximation. This agrees with the widely held view that refining a network with real data can significantly reduce the render gap, and improve generalisation [77].

It is important to highlight that the results indicate an error increase according to the template complexity. For the rectangular templates, like DS1 and DS2, there is less of an error gap with and without the Depth Refinement Block. This is likely because these objects are the less difficult to represent and easier for the network to generalise, because their intrinsic deformation space is smaller compared with the 2D manifold objects that deform in more complex ways. The RMSE gap for the 2D manifold templates is large, and the benefit of the Depth Refinement Block is very evident in these cases.

#### 5.9.3. The benefit of registration refinement

We evaluate the impact of the Registration Refinement Block in terms of registration accuracy. Given that we lack registration ground-truth with real data we use photometric error as a proxy, computed as follows. We compute the Mean Square Error between the rendered images  $I'_i$  and the input image  $I_i$  in the visible region  $\mathcal{I}_i$ :

$$\mathcal{E}_{pr} = \left( \frac{1}{|\mathcal{I}_i|} \sum_{(u,v) \in \mathcal{I}_i} (I'_i(u,v) - I_i(u,v))^2 \right)^{\frac{1}{2}}. \quad (14)$$

We show in table 17 the photometric error and qualitative results when using the output of the Main Block, and when using the Registration Refinement Block. In terms of photometric error, the Registration Refinement Block output has less error than the Main Block output, which we recall was trained using only synthetic data. The templates with more texture features like DS5, DS2, and DS3 show qualitatively more improvement than DS1 and DS4, which have less texture. tables 17 and 10 show quantitative and qualitative results before and after registration refinement.

We also give a qualitative visualization of the registration error, computed by blending the input image  $I$  and the rendered image  $I'$ , computed from the DeepSfT registration:

$$I_{avg} = \frac{I'_i + I_i}{2}. \quad (15)$$

A sharper  $I_{avg}$  indicates a better registration. We show this visualization in figure 5, where the greater the photometric error,

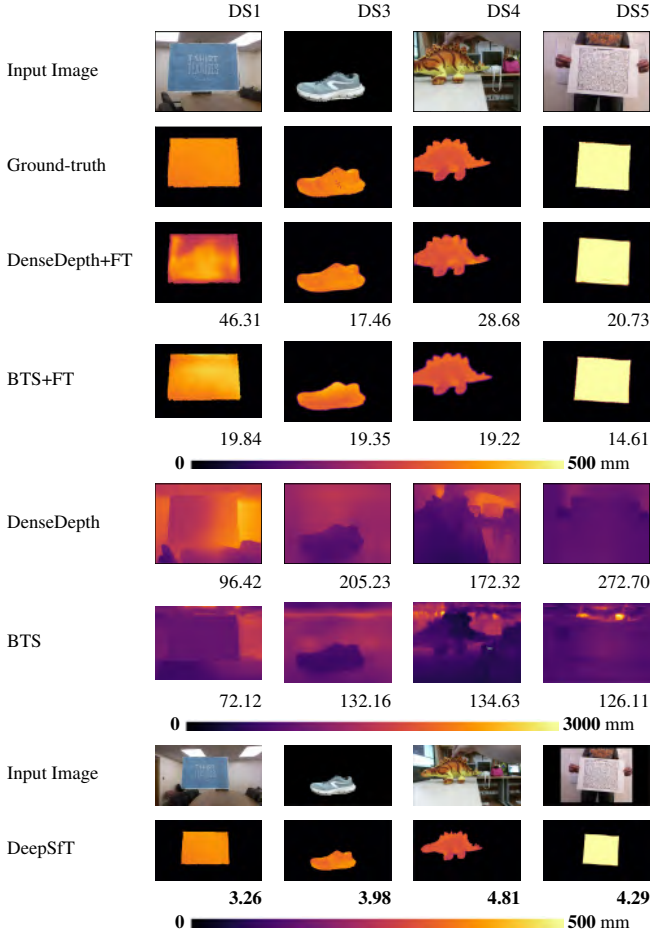


Table 8: Representative results and comparison of DeepSfT with other monocular depth reconstruction methods. The estimated depth maps and corresponding RMSE error in mm are shown for each method with one example input image from 4 templates (arranged in 4 columns).

the worse the accuracy of the registration, and the more blurred the average image. In table 17 we show the registration error visualization zoomed in the region of interest to provide a better visualization. We can clearly see a strong registration improvement with DS5, DS2 and DS4, with a smaller improvement for DS1 and no clear improvement with DS3.

### 5.10. Timing experiments

Table 18 shows the average frame rate of the compared methods, benchmarked on a conventional Linux desktop PC with a single NVIDIA GTX-1080 GPU. The DNN methods are considerably faster than the other methods, with frame rates close to real time for DeepSfT. Solutions based on CH17 are far from real-time.

### 5.11. Monocular depth estimation comparison

We have compared object-generic monocular reconstruction method with DeepSfT. We use DenseDepth [33] and BTS [34], two state-of-the-art DNN monocular reconstruction methods. The former is based on DenseNet [78]. We evaluated their ability to recover the object’s depth with two experiments. 1) We

|          |         | Registration RMSE (px) |             | Reconstruction RMSE (mm) |           |       |             |
|----------|---------|------------------------|-------------|--------------------------|-----------|-------|-------------|
| Sequence | Samples | R50F                   | DeepSfT     | CH17+GTR                 | CH17R+GTR | R50F  | DeepSfT     |
| DS3S     | 5000    | 7.14                   | <b>1.05</b> | 45.21                    | 43.67     | 6.34  | <b>1.16</b> |
| DS4S     | 5000    | 8.93                   | <b>3.60</b> | 73.80                    | 70.70     | 12.62 | <b>1.57</b> |
| DS3R     | 1300    | -                      | -           | -                        | -         | 12.43 | <b>8.12</b> |
| DS4R     | 550     | -                      | -           | -                        | -         | 27.31 | <b>6.86</b> |

Table 9: Quantitative evaluation on synthetic and real data with 2D manifold templates (DS3S, DS4S, DS3R and DS4R).

|          |         | Reconstruction RMSE (mm) |             | Registration (photometric error $\mathcal{E}_{pr}$ ) |              |
|----------|---------|--------------------------|-------------|--|--------------|
| Sequence | Samples | DeepSfT                  | DeepSfT+TV  | DeepSfT  | DeepSfT+PR   |
| DS1R     | 5000    | 9.51                     | <b>4.12</b> | 0.266  | <b>0.211</b> |
| DS2R     | 5000    | 7.37                     | <b>3.39</b> | 0.094  | <b>0.015</b> |
| DS3R     | 1300    | 8.12                     | <b>7.40</b> | 0.141  | <b>0.109</b> |
| DS4R     | 550     | 6.86                     | <b>5.80</b> | 0.196  | <b>0.184</b> |
| DS5R     | 50      | 6.97                     | <b>6.89</b> | 0.388  | <b>0.203</b> |

Table 10: Quantitative evaluation on real test data and 2D manifold templates (DS1R, DS2R, DS3R, DS4R and DS5R).

test DenseDepth and BTS depth accuracy on the real datasets when they are pre-trained with the NYUDepth dataset [73], which contains RGB-D images from indoor scenes with different types of common objects. To make a fair comparison we have adapted our images to match the intrinsics of the NYUDepth dataset. We compute depth error only in the visible region of each image. We see that DenseDepth and BTS depth RMSE are several orders of magnitude higher compared to DeepSfT. 2) We fine tune DenseDepth and BTS with all training examples from DS1R, DS3R, DS4R and DS5R datasets at the same time and separately with each one of the templates. This restricts DenseDepth and BTS to detect four different objects in the first case, and to each one of the objects in the second case. The results are shown in tables 19 and 8 where DenseDepth and BTS fine tuned versions are named as DenseDepth+FT4 and BTS+FT4 and DenseDepth+FT1 and BTS+FT1 respectively. In this case the errors have considerably reduced specially in the one object fine tune case, but they are still larger than the error achieved by DeepSfT. With this experiment we show that instance-level monocular reconstruction solutions such as DeepSfT are able to achieve much more accurate reconstruction results compared to object-generic methods

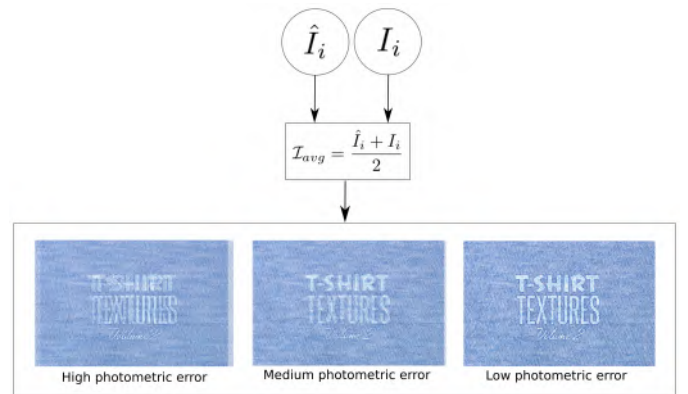


Figure 5: Visualization of registration accuracy using image blending and presenting important reference cases.



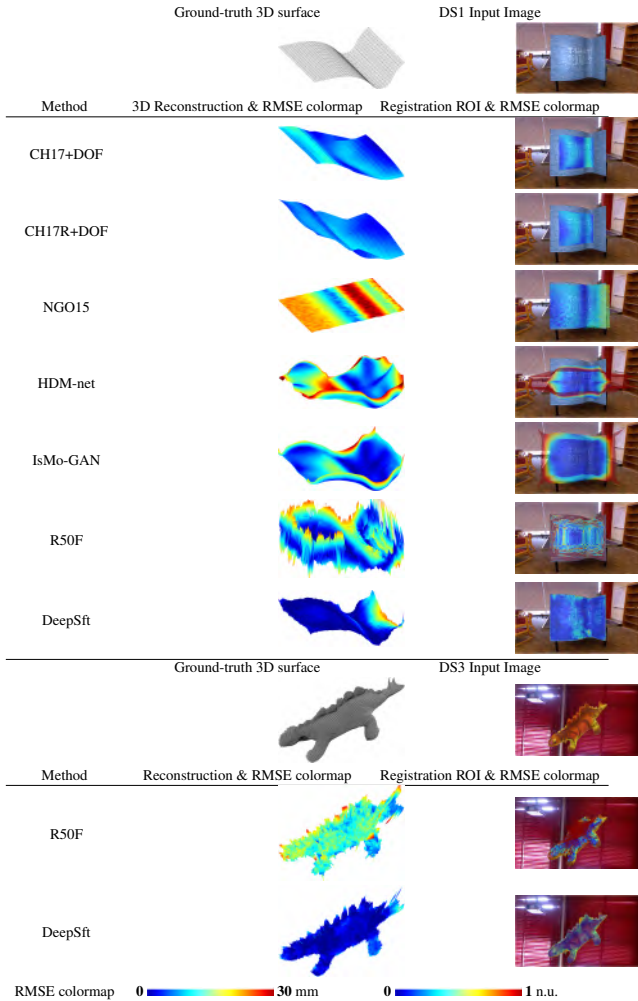


Table 11: Visual comparison of results computed from DeepSfT and other classical and DNN SfT methods with two test objects. The reconstructions are colored according to RMSE with heatmaps (middle column). The registration results are visualised with an overlay of the predicted template shape projected onto the input image. Registration errors are visualised with heatmaps (right column). n.u. stands for normalised texture map units.

such as [33, 34]. DenseDepth and BTS obtain an approximately correct average shape, the latter being best. However, they are not able to achieve comparable results to DeepSfT, even when training them with only a reduced set of objects.

## 6. Conclusions

We have presented DeepSfT, the first dense, real-time solution for wide-baseline SfT with general templates. This has been an open computer vision problem for over a decade. No previous DNN-based method is able to accurately solve SfT for weakly-textured, non-flat object templates, such as the dinosaur or shoe examples. DeepSfT will enable many real-world applications that require dense registration and 3D reconstruction of deformable objects, in particular augmented reality with deforming objects. In future work we aim to generalise DeepSfT to multiple templates, using them as explicit inputs to our network, or by using object detectors to select an object-specific

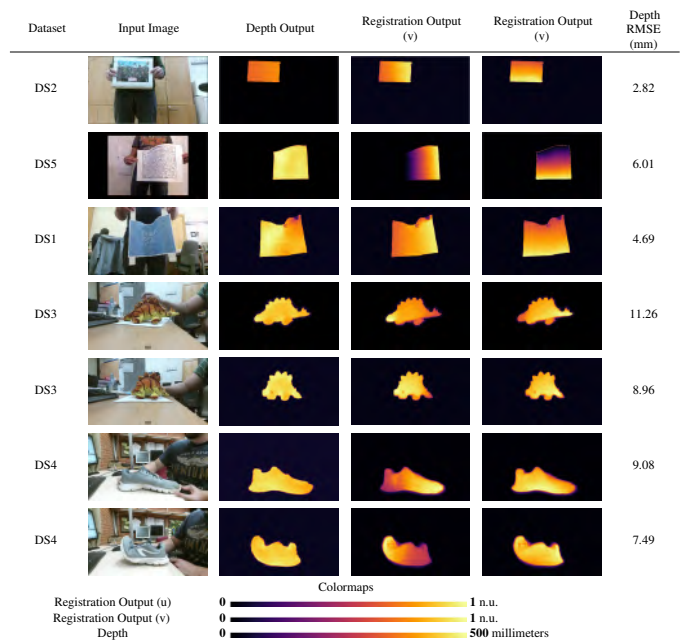


Table 12: Example outputs for the five objects used to test DeepSfT. n.u. stands for normalised units in the template texture map.

SfT network. We also will investigate how to train DeepSfT with self-supervised learning approaches, which may require incorporating other priors, such as temporal and spatial smoothness, and other deformation models. Another future way to investigate is the use of test cameras, whose intrinsics are unknown [79]. Our DeepSfT architecture and results may also contribute to develop future DNN NRSfM solutions. In particular, the use of dense maps as the network output followed by post-processing steps for mesh completion significantly reduces the complexity of the learning process, as opposed to inferring the entire surface or volume. Semi-supervised learning approaches, similar to the one implemented in DeepSfT, can also boost the goal.

## Appendix A. Warping with Bilinear Interpolation

We describe the process to create the image  $I'(u, v)$  from the registration  $\hat{\eta}(u, v) = (\hat{\eta}_U(u, v), \hat{\eta}_V(u, v))$  and the texture map  $A(U, V)$ . We recall that  $U, V$  are normalised coordinates drawn from the unit square. We define the texture map image  $\tilde{A}(\tilde{U}, \tilde{V})$  of size  $H \times W$  with  $(\tilde{U}, \tilde{V}) \in [1, W] \times [1, H]$  being pixel coordinates, obtained by de-normalising  $(U, V)$  with  $W$  and  $H$ . Image coordinates  $(u, v) \in [1, w] \times [1, h]$  are already in pixels and  $I'(u, v)$  is of size  $h \times w$  pixels. In addition, we assume that both  $\tilde{A}$  and  $I'$  are single channel images. The generalisation to 3-channel images is straightforward. We have that  $\hat{\eta}(u, v)$  is a differentiable function of the Photometric Refinement Block weights  $\theta_{\mathcal{R}}$  and  $\frac{\partial \hat{\eta}_U(u, v)}{\partial \theta_{\mathcal{R}}}$  and  $\frac{\partial \hat{\eta}_V(u, v)}{\partial \theta_{\mathcal{R}}}$  are the first derivatives of the registration with respect to  $\theta_{\mathcal{R}}$ . We recall that  $\mathcal{I}$  is a subset of coordinates  $(u, v)$  where the object is visible, and thus  $I'(u, v)$  is set to a constant value outside  $\mathcal{I}$ . By using bilinear interpolation we obtain  $I'$  as follows:

| Dataset | Input Image | Ground-truth | DNN reconstruction<br>output (blue) vs GT<br>(red) | Textured DNN<br>reconstruction out-<br>put | 3D Shape comple-<br>tion                        |
|---------|-------------|--------------|--|--|---|
| DS1     |             |              |  |  | DNN RMSE (mm)<br><br>ARAP RMSE (mm)<br>3.41     |
| DS3     |             |              |  |  | DNN RMSE (mm)<br>9.51<br>ARAP RMSE (mm)<br>9.84 |
| DS4     |             |              |  |  | DNN RMSE (mm)<br>7.42<br>ARAP RMSE (mm)<br>7.47 |

Table 13: Examples of DeepSfT results before and after ARAP 3D shape completion as described in §3.4 using real data. From left to right column we show, the dataset of the template used, the input image to use, the groundtruth of the surface reconstructed, the comparison of the reconstruction and the groundtruth overlaid, the textured reconstruction, and the result of ARAP shape completion.

$$I'(u, v) = \begin{cases} \sum_{i=1}^4 w_i \bar{A}(\bar{U}_i, \bar{V}_i) & (u, v) \in \mathcal{I} \\ 0 & \text{otherwise} \end{cases} \quad (\text{A.1})$$

where:

$$\begin{aligned} \bar{U}_1 = \bar{U}_2 = \zeta_U(\lfloor \hat{\eta}_U(u, v) \rfloor) \quad \bar{U}_3 = \bar{U}_4 = \zeta_U(\lceil \hat{\eta}_U(u, v) \rceil) \\ \bar{V}_1 = \bar{V}_3 = \zeta_V(\lfloor \hat{\eta}_V(u, v) \rfloor) \quad \bar{V}_2 = \bar{V}_4 = \zeta_V(\lceil \hat{\eta}_V(u, v) \rceil) \end{aligned} \quad (\text{A.2})$$

and

$$\begin{aligned} w_1 &= (1 + \bar{U}_1 - \hat{\eta}_U(u, v))(1 + \bar{V}_1 - \hat{\eta}_V(u, v)) \\ w_2 &= (1 + \bar{U}_1 - \hat{\eta}_U(u, v))(1 - \bar{V}_2 + \hat{\eta}_V(u, v)) \\ w_3 &= (1 - \bar{U}_3 + \hat{\eta}_U(u, v))(1 + \bar{V}_3 - \hat{\eta}_V(u, v)) \\ w_4 &= (1 - \bar{U}_4 + \hat{\eta}_U(u, v))(1 - \bar{V}_4 + \hat{\eta}_V(u, v)) \end{aligned} \quad (\text{A.3})$$

with  $\lfloor \cdot \rfloor$  and  $\lceil \cdot \rceil$  being the ‘floor’ and ‘ceil’ operators respectively. We assume that their derivatives with respect to  $\theta_{\mathcal{R}}$  are zero. Also,  $\zeta_U(x) = \max(\min(x, W), 1)$  and  $\zeta_V(x) = \max(\min(x, H), 1)$  are functions ensuring that coordinates remain inside the domain of  $\bar{A}$ . Therefore, all terms in equa-

tion (A.3) are bilinear in  $\eta(u, v)$  with:

$$\frac{\partial I'(u, v)}{\partial \theta_{\mathcal{R}}} = \begin{cases} \sum_{i=1}^4 \frac{\partial w_i}{\partial \theta_{\mathcal{R}}} \bar{A}(\bar{U}_i, \bar{V}_i) & (u, v) \in \mathcal{I} \\ 0 & \text{otherwise.} \end{cases} \quad (\text{A.4})$$

## References

- [1] R. Hartley, A. Zisserman, Multiple view geometry in computer vision, Cambridge university press, 2003.
- [2] C. Bregler, A. Hertzmann, H. Biermann, Recovering non-rigid 3D shape from image streams, in: IEEE Conference on Computer Vision and Pattern Recognition, IEEE, 2000.
- [3] L. Torresani, A. Hertzmann and C. Bregler., Nonrigid structure-from-motion: Estimating shape and motion with hierarchical priors., IEEE Transactions on Pattern Analysis and Machine Intelligence 30(5) (2008) 878–892.
- [4] Y. Dai, H. Li, and M. He., A simple prior-free method for non-rigid structure-from-motion factorization., in: IEEE Conference on Computer Vision and Pattern Recognition, 2012.
- [5] A. Chhatkuli, D. Pizarro, T. Collins, A. Bartoli, Inextensible non-rigid structure-from-motion by second-order cone programming, IEEE Transactions on Pattern Analysis and Machine Intelligence (2017) 1–1.
- [6] M. Salzmann, F. Moreno-Noguer, V. Lepetit, P. Fua, Closed-form solution to non-rigid 3d surface registration, European Conference on Computer Vision (2008) 581–594.

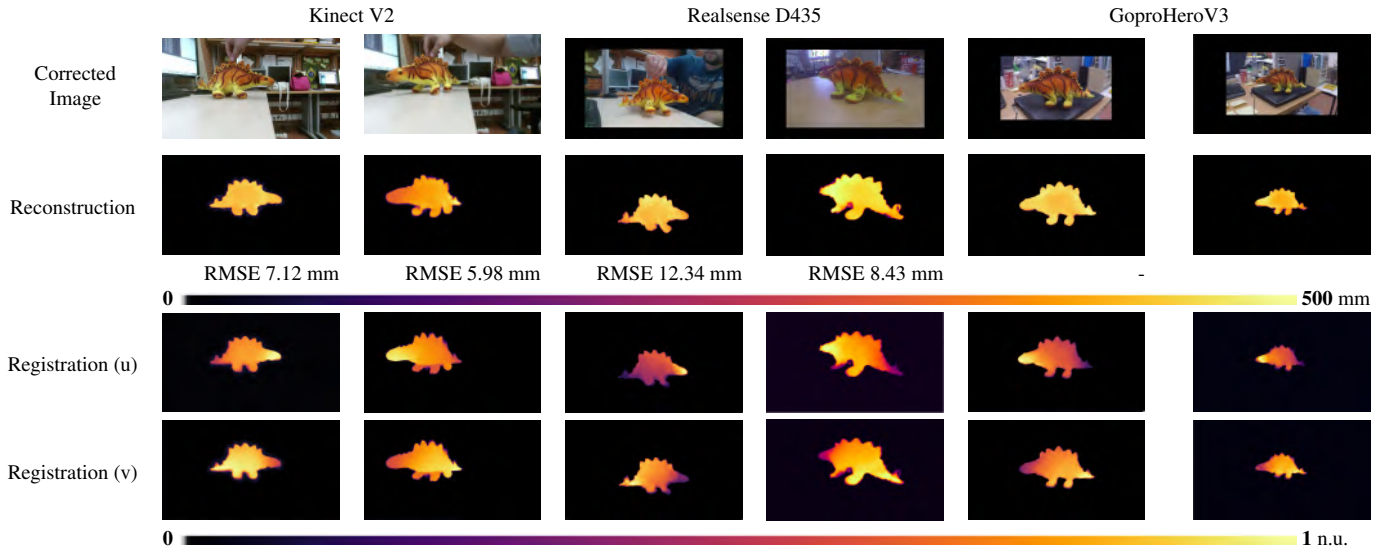


Table 14: Experimental results with different camera models. n.u. stands for normalised units in the template texture map. We show two sample cases for each camera used and we recall that give that GoproHero V3 is only an RGB camera, we can't calculate the RMSE in these cases.

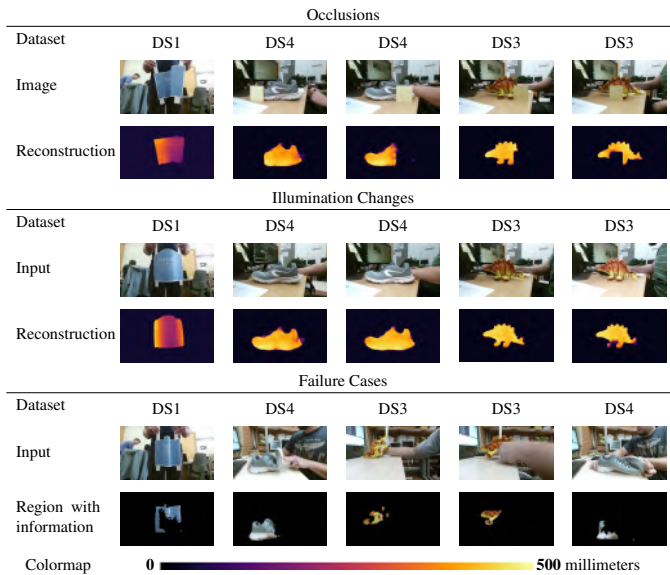


Table 15: Representative occlusion resistance, light change resistance and failure cases.

- [7] A. Bartoli, Y. Gérard, F. Chadebecq, T. Collins, D. Pizarro, Shape-from-template, *IEEE Transactions on Pattern Analysis and Machine Intelligence* 37 (10) (2015) 2099–2118.
- [8] D. T. Ngo, J. Östlund, P. Fua, Template-based monocular 3d shape recovery using laplacian meshes, *IEEE Transactions on Pattern Analysis and Machine Intelligence* 38 (1) (2016) 172–187.
- [9] A. Chhatkuli, D. Pizarro, A. Bartoli, T. Collins, A stable analytical framework for isometric shape-from-template by surface integration, *IEEE Transactions on Pattern Analysis and Machine Intelligence* 39 (5) (2017) 833–850.
- [10] D. Casillas-Perez, D. Pizarro, D. Fuentes-Jimenez, M. Mazo, A. Bartoli, The isowarp: The template-based visual geometry of isometric surfaces, *International Journal of Computer Vision* (May 2021). doi:10.1007/s11263-021-01472-w. URL <https://doi.org/10.1007/s11263-021-01472-w>
- [11] D. Pizarro, A. Bartoli, Feature-based deformable surface detection with self-occlusion reasoning, *International Journal of Computer Vision* 97 (1)

| Sequence | DeepSfT (Main Block only) RMSE | DeepSfT RMSE |
|----------|--------------------------------|--------------|
| DS3R     | 70.55                          | 8.12         |
| DS4R     | 85.58                          | 6.86         |
| DS2R     | 14.43                          | 1.53         |
| DS1R     | 17.20                          | 2.32         |

Table 16: Results and comparison from left to right, of DeepSfT with and without Depth Refinement Block. All the errors are expressed in mm.

- (2012) 54–70.
- [12] V. Gay-Bellile, A. Bartoli, P. Sayd, Direct estimation of nonrigid registrations with image-based self-occlusion reasoning, *IEEE Transactions on Pattern Analysis and Machine Intelligence* 32 (1) (2010) 87–104. doi:10.1109/TPAMI.2008.265.
- [13] T. Collins, P. Mesejo, A. Bartoli, An analysis of errors in graph-based keypoint matching and proposed solutions, in: *European Conference on Computer Vision*, Springer, 2014, pp. 138–153.
- [14] T. Collins, A. Bartoli, N. Bourdel, 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*, Springer, 2016, pp. 404–412.
- [15] A. Agudo, F. Moreno-Noguer, B. Calvo, J. M. M. Montiel, Sequential non-rigid structure from motion using physical priors, *IEEE Transactions on Pattern Analysis and Machine Intelligence* 38 (5) (2016) 979–994.
- [16] A. Pumarola, A. Agudo, L. Porzi, A. Sanfeliu, V. Lepetit, F. Moreno-Noguer, Geometry-aware network for non-rigid shape prediction from a single view, in: *IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 4681–4690.
- [17] V. Golyanik, S. Shimada, K. Varanasi, D. Stricker, Hdm-net: Monocular non-rigid 3d reconstruction with learned deformation model, *CoRR* abs/1803.10193 (2018). arXiv:1803.10193. URL <http://arxiv.org/abs/1803.10193>
- [18] S. Shimada, V. Golyanik, C. Theobalt, D. Stricker, IsMo-GAN: Adversarial learning for monocular non-rigid 3d reconstruction, in: *IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, 2019.
- [19] Q. Liu-Yin, R. Yu, L. Agapito, A. Fitzgibbon, C. Russell, Better together: Joint reasoning for non-rigid 3d reconstruction with specularities and shading, *British Machine Vision Conference (BMVC)* (2016) 42.1–42.12.
- [20] E. Ilg, N. Mayer, T. Saikia, M. Keuper, A. Dosovitskiy, T. Brox, FlowNet 2.0: Evolution of optical flow estimation with deep networks, in: *IEEE*

| Dataset            | Input Image | Average Image without registration refinement | Average Image with registration refinement | Zoomed average Image without registration refinement | Zoomed average Image with registration refinement |
|--------------------|-------------|---|--|--|---|
| DS5                |             |   |  |  |   |
| $\mathcal{E}_{pr}$ |             |   | 0.380                                      |  | 0.345   |
| DS1                |             |   |  |  |   |
| $\mathcal{E}_{pr}$ |             |   | 0.309                                      |  | 0.250   |
| DS3                |             |   |  |  |   |
| $\mathcal{E}_{pr}$ |             |   | 0.133                                      |  | 0.101   |
| DS2                |             |   |  |  |   |
| $\mathcal{E}_{pr}$ |             |   | 0.090                                      |  | 0.015   |
| DS4                |             |   |  |  |   |
| $\mathcal{E}_{pr}$ |             |   | 0.192                                      |  | 0.172   |

Table 17: Examples from DS1, DS2, DS3, DS4 and DS5 templates showing photometric error with and without registration refinement. The first column shows the input images and the third, fourth, fifth and sixth columns show the Main Block results, the Registration Refinement Block results and the corresponding photometric errors from equation (14).

|            | DeepSfT | R50F  | CH17 | CH17R | DOF  | NGO15 | HDM-net | IsMo-GAN |
|------------|---------|-------|------|-------|------|-------|---------|----------|
| Time (fps) | 20.40   | 37.00 | 0.75 | 0.19  | 8.84 | 0.03  | 25.12   | 10.47    |

Table 18: Average framerate of the evaluated methods.

- conference on computer vision and pattern recognition, 2017, pp. 2462–2470.
- [21] A. Dosovitskiy, P. Fischery, E. Ilg, P. Hausser, C. Hazirbas, V. Golkov, P. van der Smagt, D. Cremers, T. Brox, FlowNet: Learning optical flow with convolutional networks, in: IEEE International Conference on Computer Vision, IEEE Computer Society, 2015, pp. 2758–2766.
- [22] N. Sundaram, T. Brox, K. Keutzer, Dense point trajectories by gpu-accelerated large displacement optical flow, European Conference on Computer Vision (2010).
- [23] Z. Lv, K. Kim, A. Troccoli, D. Sun, J. M. Rehg, J. Kautz, Learning rigidity in dynamic scenes with a moving camera for 3d motion field estimation, in: European Conference on Computer Vision (ECCV), 2018, pp. 468–484.
- [24] M. Hornacek, A. Fitzgibbon, C. Rother, SpheroFlow: 6 dof scene flow from rgb-d pairs, in: IEEE Conference on Computer Vision and Pattern

- Recognition, 2014, pp. 3526–3533.
- [25] R. Schuster, O. Wasenmuller, G. Kuschik, C. Bailer, D. Stricker, Scene-flowfields: Dense interpolation of sparse scene flow correspondences, in: IEEE Winter Conference on Applications of Computer Vision (WACV), IEEE, 2018, pp. 1056–1065.
- [26] A. Wedel, T. Brox, T. Vaudrey, C. Rabe, U. Franke, D. Cremers, Stereoscopic scene flow computation for 3d motion understanding, International Journal of Computer Vision 95 (1) (2011) 29–51.
- [27] J. Hur, S. Roth, Self-supervised monocular scene flow estimation, in: IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2020, pp. 7396–7405.
- [28] Y. Zou, Z. Luo, J.-B. Huang, Df-net: Unsupervised joint learning of depth and flow using cross-task consistency, in: European conference on computer vision (ECCV), 2018, pp. 36–53.
- [29] F. Brickwedde, S. Abraham, R. Mester, Mono-sf: Multi-view geometry meets single-view depth for monocular scene flow estimation of dynamic traffic scenes (2019). arXiv:1908.06316.
- [30] D. Eigen, R. Fergus, Predicting depth, surface normals and semantic labels with a common multi-scale convolutional architecture, in: IEEE International Conference on Computer Vision, 2015, pp. 2650–2658.
- [31] R. Garg, V. K. B.G., G. Carneiro, I. Reid, Unsupervised cnn for single

| Sequence | DenseDepth RMSE | DenseDepth+FT4 RMSE | DenseDepth+FT1 RMSE | BTS RMSE | BTS+FT4 RMSE | BTS+FT1 RMSE | DeepSft RMSE |
|----------|-----------------|---------------------|---------------------|----------|--------------|--------------|--------------|
| DS3      | 184.52          | 32.53               | 17.64               | 122.06   | 23.45        | 12.67        | <b>5.52</b>  |
| DS4      | 221.84          | 18.96               | 13.92               | 96.80    | 17.83        | 9.44         | <b>4.76</b>  |
| DS5      | 395.53          | 22.47               | 9.76                | 84.45    | 12.34        | 7.43         | <b>6.47</b>  |
| DS1      | 93.87           | 51.25               | 10.33               | 89.22    | 14.76        | 8.12         | <b>2.33</b>  |

Table 19: Reconstruction, results comparison of [33, 34], with and without fine tuning and DeepSft. All the errors are in mm.

- view depth estimation: Geometry to the rescue, in: European Conference on Computer Vision, Springer International Publishing, 2016, pp. 740–756.
- [32] F. Liu, C. Shen, G. Lin, I. D. Reid, Learning depth from single monocular images using deep convolutional neural fields., *IEEE Transactions on Pattern Analysis and Machine Intelligence* 38 (10) (2016) 2024–2039.
- [33] I. Alhashim, P. Wonka, High quality monocular depth estimation via transfer learning, *arXiv e-prints abs/1812.11941* (2018). [arXiv:1812.11941](https://arxiv.org/abs/1812.11941).  
URL <https://arxiv.org/abs/1812.11941>
- [34] J. H. Lee, M. Han, D. W. Ko, I. H. Suh, From big to small: Multi-scale local planar guidance for monocular depth estimation, *CoRR abs/1907.10326* (2019).
- [35] J. Martinez, R. Hossain, J. Romero, J. J. Little, A simple yet effective baseline for 3d human pose estimation, in: *IEEE International Conference on Computer Vision*, Vol. 1, 2017, p. 5.
- [36] R. Alp Güler, N. Neverova, I. Kokkinos, Densepose: Dense human pose estimation in the wild, in: *IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 7297–7306.
- [37] A. Kendall, M. Grimes, R. Cipolla, PoseNet: A convolutional network for real-time 6-dof camera relocalization, in: *IEEE International Conference on Computer Vision (ICCV)*, 2015, pp. 2938–2946. doi:10.1109/ICCV.2015.336.
- [38] J. Bednarik, P. Fua, M. Salzmann, Learning to reconstruct texture-less deformable surfaces from a single view, in: *International Conference on 3D Vision (3DV)*, 2018, pp. 606–615. doi:10.1109/3DV.2018.00075.
- [39] A. Tsoli, A. Argyros, Patch-based reconstruction of a textureless deformable 3d surface from a single rgb image, in: *IEEE International Conference on Computer Vision (ICCV)*, 2019, pp. 4034–4043. doi:10.1109/ICCVW.2019.00498.
- [40] D. Fuentes-Jimenez, D. Casillas-Perez, A. Bartoli, T. Collins, D. P. Pérez, Deep shape from template dataset(dsftd) (2021). doi:10.34740/KAGGLE/DSV/1955934.  
URL <https://www.kaggle.com/dsv/1955934>
- [41] T. Collins, A. Bartoli, Using isometry to classify correct/incorrect 3D-2D correspondences, in: *European Conference on Computer Vision*, 2014.
- [42] M. Salzmann, P. Fua, Reconstructing sharply folding surfaces: A convex formulation, in: *IEEE Conference on Computer Vision and Pattern Recognition*, IEEE, 2009, pp. 1054–1061.
- [43] M. Perriollat, R. Hartley, A. Bartoli, Monocular template-based reconstruction of inextensible surfaces, *International Journal of Computer Vision* 95 (2) (2011) 124–137.
- [44] F. Brunet, A. Bartoli, R. I. Hartley, Monocular template-based 3d surface reconstruction: Convex inextensible and nonconvex isometric methods, *Computer Vision and Image Understanding* 125 (2014) 138–154.
- [45] D. G. Lowe, Distinctive image features from scale-invariant keypoints, *International Journal of Computer Vision* 60 (2004) 91–110.
- [46] J. Pilet, V. Lepetit, P. Fua, Fast non-rigid surface detection, registration and realistic augmentation, *International Journal of Computer Vision* 76 (2) (2008) 109–122.
- [47] E. Özgür, A. Bartoli, Particle-sft: A provably-convergent, fast shape-from-template algorithm, *International Journal of Computer Vision* 123 (2) (2017) 184–205.
- [48] A. Malti, R. Hartley, A. Bartoli, J.-H. Kim, Monocular template-based 3d reconstruction of extensible surfaces with local linear elasticity, in: *IEEE Conference on Computer Vision and Pattern Recognition*, 2013, pp. 1522–1529.
- [49] A. Malti, A. Bartoli, R. Hartley, A linear least-squares solution to elastic shape-from-template, in: *IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 1629–1637.
- [50] N. Haouchine, S. Cotin, Template-based monocular 3D recovery of elastic shapes using lagrangian multipliers, in: *IEEE Conference on Computer Vision and Pattern Recognition*, IEEE, 2017. doi:10.1109/cvpr.2017.381.
- [51] N. Haouchine, J. Dequidt, M.-O. Berger, S. Cotin, Single view augmentation of 3D elastic objects, in: *International Symposium on Mixed and Augmented Reality*, IEEE, 2014, pp. 229–236.
- [52] A. Agudo, F. Moreno-Noguer, Simultaneous pose and non-rigid shape with particle dynamics, in: *IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 2179–2187.
- [53] D. Casillas-Perez, D. Pizarro, D. Fuentes-Jimenez, M. Mazo, A. Bartoli, Equiareal shape-from-template, *J. Math. Imaging Vis.* 61 (5) (2019) 607–626.
- [54] N. Hafez, V. Dietrich, S. Roehrl, Analysis of different methods to close the reality gap for instance segmentation in a flexible assembly cell, in: S. Zeghloul, M. A. Laribi, J. S. Sandoval Arevalo (Eds.), *Advances in Service and Industrial Robotics*, Springer International Publishing, Cham, 2020, pp. 505–515.
- [55] Blender Online Community, Blender - a 3D modelling and rendering package, Blender Foundation, Blender Institute, Amsterdam.  
URL <http://www.blender.org>
- [56] J. D. Foley, A. van Dam, S. K. Feiner, J. F. Hughes, *Computer Graphics: Principles and Practice* (2nd Ed.), Addison-Wesley Longman Publishing Co., Inc., USA, 1990.
- [57] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, in: *IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.
- [58] V. Badrinarayanan, A. Kendall, R. Cipolla, Segnet: A deep convolutional encoder-decoder architecture for image segmentation., *CoRR abs/1511.00561* (2015).
- [59] K. H. X. Z. S. R. J. Sun, Deep residual learning for image recognition, *Arxiv arXiv:1512.03385* (December 2015).
- [60] O. Sorkine, M. Alexa, As-rigid-as-possible surface modeling, in: *Fifth Eurographics symposium on Geometry processing*, 2007, pp. 109–116.
- [61] S. Parashar, D. Pizarro, A. Bartoli, T. Collins, As-rigid-as-possible volumetric shape-from-template, in: *IEEE International Conference on Computer Vision*, 2015.
- [62] T. Collins, A. Bartoli, Realtime shape-from-template: System and applications., in: *International Symposium on Mixed and Augmented Reality, ISMAR*, 2015, pp. 116–119.
- [63] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, et al., *TensorFlow: Large-scale machine learning on heterogeneous systems*, software available from tensorflow.org (2015).  
URL <https://www.tensorflow.org/>
- [64] J. B. Diederik P. Kingma, Adam: A method for stochastic optimization, *Arxiv arXiv:1412.6980* (6) (December 2014).
- [65] N. S. Keskar, R. Socher, Improving generalization performance by switching from adam to SGD, *CoRR abs/1712.07628* (2017). [arXiv:1712.07628](https://arxiv.org/abs/1712.07628).  
URL <http://arxiv.org/abs/1712.07628>
- [66] Z. Zhang, L. Ma, Z. Li, C. Wu, Normalized direction-preserving adam, *CoRR abs/1709.04546* (2017).
- [67] Y. B. Xavier Glorot, Antoine Bordes, Understanding the difficulty of training deep feedforward neural networks, *Machine Learning Research* (2010).
- [68] L. I. Rudin, S. Osher, E. Fatemi, Nonlinear total variation based noise removal algorithms, *Physica D: Nonlinear Phenomena* 60 (1) (1992) 259–268. doi:https://doi.org/10.1016/0167-2789(92)90242-F.  
URL <http://www.sciencedirect.com/science/article/pii/016727899290242F>
- [69] D. Strong, T. Chan, Edge-preserving and scale-dependent properties of total variation regularization, *Inverse Problems* 19 (6) (2003) S165–S187. doi:10.1088/0266-5611/19/6/059.

- [70] Z. Ren, J. Yan, B. Ni, B. Liu, X. Yang, H. Zha, Unsupervised deep learning for optical flow estimation, in: AAAI Conference on Artificial Intelligence, 2017.
- [71] Computer Vision Laboratory, Deformable Surface Reconstruction-Database, Computer Vision Laboratory, Ecole Polytechnique Fédérale de Lausanne – EPFL.  
URL <https://cvlab.epfl.ch/data/data-dsr-index-php/>
- [72] Agisoft Photoscan, <https://www.agisoft.com>.
- [73] P. K. Nathan Silberman, Derek Hoiem, R. Fergus, Indoor segmentation and support inference from rgb-d images, in: European Conference on Computer Vision, 2012.
- [74] Intel, Intel realsense d435 stereo depth camera, <http://realsense.intel.com>.
- [75] GoPro, Gopro hero silver v3 rgb camera, <https://es.gopro.com/update/hero3>.
- [76] D. T. Ngo, S. Park, A. Jorstad, A. Crivellaro, C. D. Yoo, P. Fua, Dense image registration and deformable surface reconstruction in presence of occlusions and minimal texture, in: IEEE International Conference on Computer Vision (ICCV), 2015, pp. 2273–2281.
- [77] H. Zhang, Y. Tian, K. Wang, H. He, F. Wang, Synthetic-to-real domain adaptation for object instance segmentation, in: International Joint Conference on Neural Networks (IJCNN), 2019, pp. 1–7.
- [78] G. Huang, Z. Liu, K. Q. Weinberger, Densely connected convolutional networks, CoRR abs/1608.06993 (2016). arXiv:1608.06993.  
URL <http://arxiv.org/abs/1608.06993>
- [79] A. Bartoli, T. Collins, Template-based isometric deformable 3d reconstruction with sampling-based focal length self-calibration, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2013, pp. 1514–1521.