Texture-Generic Deep Shape-From-Template

David Fuentes-Jimenez, Daniel Pizarro, David Casillas-Perez, Toby Collins, Adrien Bartoli

IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received April 26, 2021, accepted May 16, 2021, date of publication May 19, 2021, date of current version May 27, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3082011

Texture-Generic Deep Shape-From-Template

DAVID FUENTES-JIMENEZ[1], DANIEL PIZARRO[1], DAVID CASILLAS-PEREZ[2], TOBY COLLINS[3], AND ADRIEN BARTOLI[4]

[1]Department of Electronics, Universidad de Alcalá (UAH), E-28871 Alcalá de Henares, Spain
[2]Department of Signal Processing and Communications, Universidad Rey Juan Carlos (URJC), 28933 Madrid, Spain
[3]IRCAD, Place de l'Hôpital, 67000 Strasbourg, France
[4]EnCoV, 63000 Clermont-Ferrand, France

Corresponding author: David Fuentes-Jimenez (d.fuentes@edu.uah.es)

The unique source of funding and support for this work was the Project ARTEMISA under Grant TIN2016-80939-R provided by the Spanish Ministry of Economy, Industry and Competitiveness.

ABSTRACT Shape-from-Template (SfT) solves the registration and 3D reconstruction of a deformable 3D object, represented by the template, from a single image. Recently, methods based on deep learning have been able to solve SfT for the wide-baseline case in real-time, clearly surpassing classical methods. However, the main limitation of current methods is the need for fine tuning of the neural models to a specific geometry and appearance represented by the template texture map. We propose the first texture-generic deep learning SfT method which adapts to new texture maps at run-time, without the need for texture specific fine tuning. We achieve this by dividing the problem into a segmentation step and a registration and reconstruction step, both solved with deep learning. We include the template texture map as one of the neural inputs in both steps, training our models to adapt to different ones. We show that our method obtains comparable or better results to previous deep learning models, which are texture specific. It works in challenging imaging conditions, including complex deformations, occlusions, motion blur and poor textures. Our implementation runs in real-time, with a low-cost GPU and CPU.

INDEX TERMS Monocular, 3D model, image registration, 3D reconstruction, wide-baseline, dense, deformable reconstruction, shape-from-template.

I. INTRODUCTION

Image registration and image-based 3D reconstruction are fundamental problems extensively studied in Computer Vision. However, solving these problems with deformable objects remains challenges. In Shape-from-Template (SfT) [1]–[4], the objective is to reconstruct the 3D shape of a deformable object from a single image and a reference 3D model of the object, known as the template. The template is a fundamental component of SfT that provides prior knowledge via three models. The first model is the shape model, typically represented as a triangulated 3D mesh, which gives the shape of the object in a known position (usually called the reference shape). The second model is the deformation model that determines how the object may deform from the reference shape. This is used to combat the general ill-posedness of 3D reconstruction from a single image, and it restricts the space of possible solutions to ones that are physically viable. The third model is the appearance model that represents the texture of the object's surface. This is required to relate the image with the template's surface at the pixel level though the object's texture. By far, the most common way to implement the appearance model is with texture mapping. This approach assigns each point on the template's surface to a pixel colour sampled from a discrete colour map known as the texture map.

The shape model and texture map are usually constructed with an optical acquisition system. There are two main approaches: The first approach uses an RGBD camera, where the depth information is used to construct the shape model and the RGB information is used to construct the texture map. The second approach uses several images taken with an RGB camera of the object in the reference position, which are then used to build the shape model and texture map using a combination of Structure-from-Motion (to provide the relative poses of the camera images) and multi-view stereo (to densely reconstruct the shape models and to build the texture map). The second approach is usually preferred in practice because the same camera can be used for constructing the template and running SfT [2], [5].

The associate editor coordinating the review of this manuscript and approving it for publication was Zijian Zhang.

Using the template and an image of the deformed object, the general goal of SfT is to infer the 3D deformation of the template so that it matches the image. All three models in the template (shape, deformation and appearance) are essential to make SfT accurately solvable. Formally, two challenging and related problems are solved in SfT. The first challenge is to register the template to the image (*i.e.* to determine a dense spatial correspondence between the image and the template shape). The second challenge is to reconstruct the template's deformed shape (*i.e.* to determine the depth of each point on the template's surface with respect to the camera center).

SfT has numerous practical applications, and an important one is to facilitate Augmented Reality (AR) with deformable objects. Figure 1 shows the workflow diagram of a standard SfT + AR system. In this application, registration is required to correctly position virtual objects in the image to align with the deforming object's surface and reconstruction is required to correctly orient the virtual objects in 3D.

The texture map is necessary as the purpose of SfT is two-fold: to register the template to the image and to reconstruct it in 3D. Both tasks are strongly interdependent, and only the texture map allows a precise registration at the pixel level using the specific texture details of the object of interest. Without a texture map, it would be impossible to exploit the object's specific texture characteristics, fundamentally preventing accurate registration and reconstruction. In practice, requiring a texture map is not a limitation because it is obtained as part of template construction using an optical system as described above.

SfT needs a deformation model to constrain the solution. Multiple deformation models have been studied, such as isometric [4], [6], [7], conformal [2], [8], [9], equiareal [10] and elastic [11]–[13] deformations. The most popular deformation models to solve SfT are isometry and quasi-isometry, also known as the as-rigid-as-possible (ARAP) model, which is a widely used relaxation of isometry. These models prevent the object from stretching or shrinking, and they make SfT well-posed in the general case.

SfT is a challenging problem when facing the conditions of a real application. The challenges are related to: a) the type and complexity of the object in terms of geometry (volumic or thin shell), deformations (low or high dimensional), and texture (rich or poor), b) the imaging conditions (including illumination changes, occlusions and motion blur) and c) the baseline (short or wide). In video sequences, there is temporal continuity between the frames, which corresponds to the short-baseline case. On the contrary, the wide-baseline case implies that each input image has to be treated individually, without temporal connection to previous images, due to large camera movements and sudden deformations. In practice, the wide-baseline conditions are more realistic, since occlusions and fast camera movements that break the short-baseline assumptions are frequent.

SfT has been extensively studied over the last decade, and only very recently, methods based on Deep Neural Networks (DNN) have been proposed. We divide classical (*i.e.*, non-DNN) SfT methods into two main categories. The first category of methods decouple SfT into a registration followed by a reconstruction step. The registration step is based on robust feature-based matching methods that are not specific to SfT [14]–[16]. The reconstruction step is based on shape inference methods, usually a combination of a non-iterative step [2], [4] followed by iterative refinement [6]. These approaches cope with volumic and thin shell objects and work under wide-baseline conditions. They are limited because feature-based registration fails under challenging image conditions and poorly textured objects. The second category of methods perform registration and reconstruction at the same time [3], [17], [18]. Methods in this category use iterative optimisation and only work in short-baseline conditions, as they require to be initialized close to the solution. They fail when the short-baseline tracking conditions are violated due to occlusions or fast motion. As in the first category, these methods also fail under challenging imaging conditions and poorly textured objects.

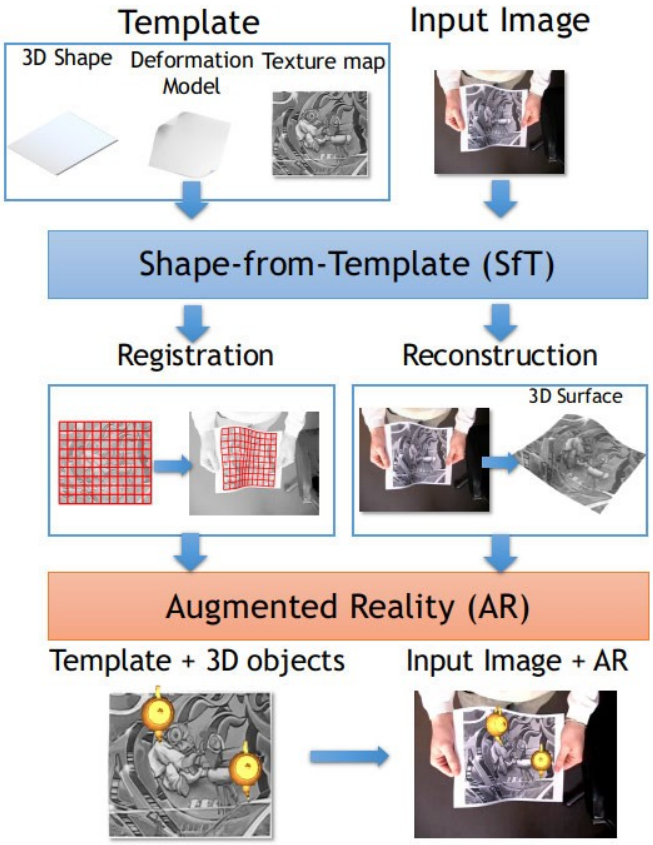

FIGURE 1. Principle of SfT and AR for deformable objects.

In recent years, the idea of using learning methods to solve SfT has been explored. These methods learn the mapping from the input image to the object 3D deformation [19]–[22]. They are potentially able to solve SfT in wide-baseline conditions and without the need to run optimisation at run-time. However, existing DNN-based SfT methods face four main limitations at the moment. First, some of them only

work with a specific template [22], requiring fine tuning for the specific template shape and texture map. Other works propose to solve SfT for a variety of texture maps [19]–[21], learning invariance to these. However, their results are far from satisfactory, still requiring fine tuning of the network to a specific template. This is because the templateless problem is highly ambiguous, making template-invariant methods unreliable. Second, in terms of template geometry, most existing methods tackle the thin-shell case [19]–[21], with the exception of the template-specific method [22]. Third, in terms of template complexity, most methods require a mesh with a reduced number of vertices (namely, 73×73 in [20], [21] and 10×10 in [19]), which is a strong limitation to work with complex templates or deformations. Fourth, they work for a specific camera configuration, not being able to adapt to dynamical camera parameters at run-time.

To sum up, a general DNN-based SfT solution is still missing, which would work for new templates without the need of fine tuning. The importance of pursuing such a solution resides in the complexity of generating training data for SfT, and the computational resources needed for fine tuning a network in a new dataset. It is thus a necessary step towards making DNN-based SfT methods widely applicable.

We make the following contributions to advance the state-of-the-art of DNN-based SfT. First, we propose the first DNN-based SfT method that takes the template texture map as a run-time input. This is used to condition the registration and reconstruction DNNs on the specific texture of the object of interest. Importantly, the texture map supplied at run-time can be completely novel in the sense that it was not (nor any similar texture map) used in the training data. Previous DNN-based methods [18−21] that exploit the specific texture map of an object require their DNNs to be retrained for the specific texture map (either from scratch or by fine-tuning), which is highly impractical for real-world applications such as those running on smartphones. Indeed, the limits of requiring DNN retraining have completely prevented DNN-based SfT methods from being used in real-world applications. Our method does not suffer this limitation of texture-specific training, which is a significant advance of state-of-the-art towards practical real-world use of DNN-based SfT.

Second, our proposed architecture is divided into two neural networks, the *segmentation network* for pixel-based detection of the template, and the *registration-reconstruction network* to recover the SfT solution. The segmentation network is crucial to solve SfT in our generic template approach. We have proposed a new semantic segmentation architecture that allows us to add the template texture map as one of the inputs, which clearly differs from the classical category-level semantic segmentation methods, where the semantic categories are learned and fixed during training. The output of the segmentation network is fed into the registration-reconstruction network.

Third, our DNN models for template segmentation and registration-reconstruction networks are fully-convolutional encoder-decoder networks, that use residual structures and specific layers, created by us to address SfT. They allow our method to be computationally efficient. In addition to our base models, we propose a lightweight architecture for the registration-reconstruction network that can be used to run our method in real-time in low-cost GPUs, CPUs and embedded systems. It is based on a new custom decoding layer that implements the inverse Block Discrete Cosine Transform (DCT), which allows us to greatly reduce the number of network parameters, while easily controlling the loss of information induced by the new decoding layers. Our lightweight architecture requires 21.59% of the total parameters and is 214.65% faster than the base architecture while being only 19.21% less accurate. This new optimized architecture based on the DCT can potentially be applied to other problems different from SfT, effectively reducing the size of encoder-decoder DNN regression and classification models. Fourth, our proposed method works independently of the camera parameters, recovering the correct depth and scale of the deformed object. This fact eliminates the need to use the same camera during the training and testing stages, which affects other methods [19]–[21]. Finally, we show that our proposed methods outperform, both qualitatively and quantitatively, the representative state-of-the-art methods in terms of accuracy, speed and number of parameters. Our results include challenging scenarios in wide-baseline conditions and effects like motion blur, occlusions, illumination changes and weak texture. All the data and code presented in this work will be released for public use.

The rest of this paper is organized as follows. Section II discusses previous work. Section III describes the SfT problem and the proposed methods with the DNN architectures. Section IV explains the training process, the loss function, and the setup carried out. Section V explains the setup, datasets and methods used for the experiments before showing the obtained results, both graphically and numerically, for various experimental settings. Finally, section VI gives our conclusions and future work.

II. PREVIOUS WORK

We divide the SfT state-of-the-art into two main categories, the classical SfT methods, which include the vast majority of existing work, and the DNN-based SfT methods. Before explaining these two categories, we introduce other vision problems that are related to SfT. After that we start with the classical methods, dividing them into two sub-categories, the decoupled and the integrated methods. The former solves registration and reconstruction as two independent problems and the latter solves registration and reconstruction jointly. Finally, we review the DNN-based methods and categorize them.

A. VISION PROBLEMS RELATED TO SfT

There are several vision problems that relate to SfT and have been recently attempted by DNN methods. These are *1)* optical flow computation [23], [24], *2)* scene flow computation [25], [26], *3)* monocular depth reconstruction

[27]–[30], *4)* human pose estimation [31], [32] and *5)* Shape-from-Shading [33], [34]. None of these methods or their combination compete with SfT-specific solutions or solve SfT under general conditions. For instance *1)* solves registration between frames in a short-baseline sequence without including depth, *2)* solves registration between frames in 3D, mainly from depth or stereo cameras and thus has not the same inputs as SfT, *3)* solves depth for a general scene from a single image and is limited to selected types of scenes (roads or indoor). Besides, it does not compute the registration with the template, which is fundamental in SfT, *4)* can be seen as a specialisation of SfT for a template that represents the human body. The deformation model in this case is low-dimensional, parametrized with a few joint angles, favouring the use of learning methods. In SfT, the number of degrees-of-freedom can be much larger than in the human skeleton, and it changes with the type of template, which makes it much more challenging than *4)*. Finally, *5)* uses radiometric models to recover 3D surfaces from shading cues detected in the images. It requires the scene to be lit by highly controlled light sources and, as in *3)*, does not include registration to the template.

B. CLASSICAL DECOUPLED SFT METHODS

The decoupled methods first obtain registration and then 3D reconstruction as two independent processes. The advantage of this approach is the reduction of complexity. The main drawback is that it does not include the constraints existing between registration and reconstruction. It usually solves wide-baseline registration using feature-based matching methods like SURF [35] and SIFT [36], which are then cleaned from mismatches using specific methods for deformable registration [14], [37]. Feature-based registration methods fail with repetitive or poorly textured objects and challenging image conditions, such as motion blur, strong viewpoint distortion or low resolution images. As a consequence, these methods are limited in challenging scenarios. The decoupled methods are usually categorised by the deformation model, isometry being the most common one. In terms of the solver strategy used in the reconstruction step, these methods typically follow one of the following approaches: *1)* the inextensibility model, a convex relaxation of isometry, and the maximum depth heuristic [1], [6], [38], [39], *2)* local differential geometry [2], [4], and *3)* minimisation of a non-convex cost function [6], [40]. Methods in *3)* are computationally expensive and require initialisation. They are mainly used as refinement methods for approaches in *1)* or *2)*, which are convex but less accurate in general. Other works study non-isometric deformations, such as non-linear elasticity [10], [41]–[43], linear elasticity [11], [12] or angle-preserving conformal models [2]. The main drawback of all these methods is that they require boundary conditions, usually in the form of 3D known points, to make the reconstruction problem well posed. Whether these non-isometric models can solve SfT uniquely without additional cues remains an open question.

C. CLASSICAL INTEGRATED SFT METHODS

The integrated methods jointly perform registration and reconstruction. The majority of them are restricted to short-baseline scenarios, which imply the use of video streams [3], [17], [44]. They minimize a non-convex cost function that jointly aligns the 3D solution with diverse image cues, such as feature point correspondences [3] or by using a pixel-level photo-consistency model [17], [44]. Integrated methods are effective and can recover complex deformations, some of them in real-time. However, they fail when the short-baseline conditions are violated, and require initialization with a wide-baseline method.

D. DNN-BASED SFT METHODS

DNN-based SfT methods have appeared in the last few years. We establish a difference between monocular reconstruction methods, that do not use a template but are specialised in deformable objects [30], [33], [45], and methods that genuinely solve SfT's registration and reconstruction [19]–[22]. We are interested in methods that belong to the second group. They are divided by type of output representation (dense or discrete) and whether they are specific to a template or work for a generic category of templates.

Regarding the type of output, the majority of methods represent the SfT solution with the 3D vertex coordinates of a regular mesh. The mesh vertex count varies between existing methods. [19] uses 10×10 meshes. Their approach is based on obtaining 2D belief maps for the image position of each of the mesh vertices, inspired by DNN-based human pose computation methods. This approach strictly limits the amount of vertices. Other approaches [20], [21] use three-channel 2D maps to model the coordinates of a regular mesh, reaching 73×73 vertex counts. We proposed a different approach [22], by recovering pixel-level depth and registration maps, allowing one to represent 3D objects and complex shapes of arbitrary topologies. In addition we implemented a post-processing step based on the ARAP model to recover the hidden parts of the surface, not explicitly reconstructed by the neural network. This paper is an extension of our previous work [22]. It uses fundamentally new constraints and scope but relies on the same parametrisation.

In terms of template specificity, all existing DNN-based methods are trained for a specific template shape, while they deal with the template texture map differently. We thus divide existing DNN-based methods in the following categories: *1) Single-texture methods* are trained for a specific template texture map. They exploit texture information to accurately solve SfT, and this information is built-in the DNN weights [22]. *2) Multi-texture methods* work similar to *1)*, but the DNN is trained for several texture maps. Although there is currently no existing method falling in *2)*, one could easily be created by combining object detection with several instances of a method from *1)*. *3) Texture-agnostic methods* are trained with many texture maps. Hence, they learn to solve SfT using general image cues that are independent of the texture map, such as occlusion boundaries and shading [19]–[21].

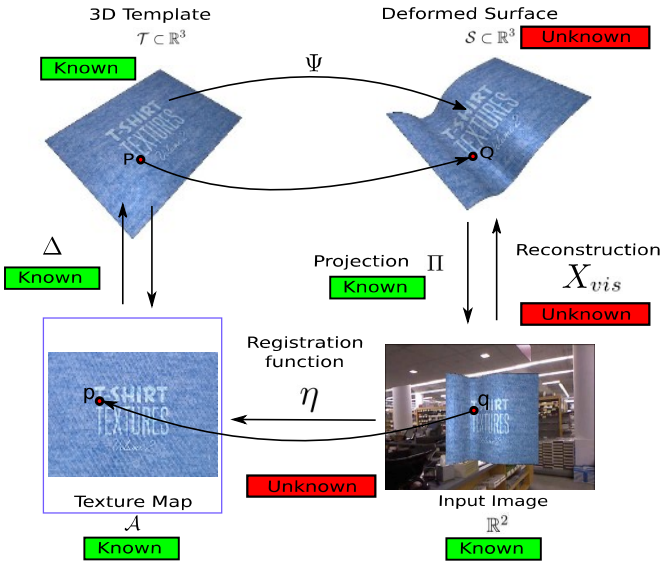

FIGURE 2. Geometric model of SfT.

Methods in *3)* are significantly less accurate than methods in *1)* and *2)*, producing coarse reconstruction, as shown in our experiments. Strictly speaking, methods in *3)* are not SfT methods, since they do not use the template texture map to solve SfT. Our work exploits the template texture map as in *1)* and *2)*, but using it as an input in the DNN. It can adapt to use new texture maps unseen during training and thus belongs to a new category of DNN methods, namely *4) Texture-generic methods*. We show in Section V that our method is only slightly less accurate than methods in *1)* and *2)*, while being much more applicable and flexible. Methods in *3)*, being independent of the texture map, have also high applicability as in *4)*. However, they fail to produce accurate reconstructions in general.

Summarizing, solving SfT with DNNs is a promising line of research, but the existing methods show important limitations. Our proposal can be classified as the first texture map-generic method, and so belongs to a new template-generic category of methods. Unlike existing methods, it exploits the texture map as an additional input to the DNN, allowing it to adapt to different texture maps at runtime. Our method uses all the available information, similarly to template-specific methods, thus guaranteeing that the SfT solution is well-posed. In terms of outputs, we use the same parametrisation as in our previous work [22], which copes with complex deformations and is not limited by the vertex count of the output mesh. Finally, we standardise the camera used during training by following [22], so that it is not necessary to use the same camera parameters during training and testing. These advances allow our method to deal with practical scenarios and contribute significantly to generic DNN-based SfT.

III. METHODOLOGY

A. SCENE GEOMETRY

Figure 2 shows the components of the scene geometry in the SfT problem. **Template.** It is composed of a known 3D surface $\mathcal{T} \subset \mathbb{R}^3$ and an appearance model represented by a texture map $\mathcal{A}_\mathcal{T} = (\mathcal{A}, A)$, where $\mathcal{A} \subset \mathbb{R}^2$ is a 2D subset and $A\colon \mathcal{A} \to (r, g, b)$ is a function that maps $\mathcal{A}$ to RGB values. In $\mathcal{A}$ normalized coordinates are used. The known $\Delta : \mathcal{A} \longmapsto \mathcal{T}$ parametrisation is a bijection that relates points of the texture map $\mathcal{A}$ to the surface $\mathcal{T}$.

1) DEFORMATION

The template $\mathcal{T}$ undergoes an unknown quasi-isometric deformation that results in the unknown surface $\mathcal{S} \subset \mathbb{R}^3$, represented from $\mathcal{T}$ by the unknown map $\Psi\colon \mathcal{T} \to \mathcal{S}$.

2) CAMERA PROJECTION

The input image is defined as a colour intensity function that contains 3 channels $I\colon \mathbb{R}^2 \to (r, g, b)$, in a discrete pixel grid. The camera model is represented with the perspective projection:

$$(x, y, z) \longmapsto \left(\frac{x}{z}, \frac{y}{z}\right) = (u, v). \tag{1}$$

We assume that the camera is intrinsically calibrated, so we know its aspect ratio, focal length and radial distortion, which in SfT methods is a reasonable and ordinary assumption. The coordinates (u, v) can be easily obtained through the image coordinates and are known as retinal coordinates.

3) VISIBLE SURFACE REGION AND REGISTRATION MAP

The visible surface $\mathcal{S}_{vis} \subset \mathcal{S}$ is represented by all the non-occluded areas of the camera image plane. Through the projection of this region in the image plane, we define a known bidimensional region $\mathcal{I} \subset \mathbb{R}^2$. $\mathcal{I}$ and $\mathcal{S}_{vis}$ are related by a perspective embedding function $X_{vis}\colon \mathcal{I} \to \mathcal{S}_{vis}$ with $X_{vis}(u, v) = \rho(u, v)\,(u, v, 1)$. In this embedding function, the function $\rho\colon \mathcal{I} \to \mathcal{S}_{vis}$ is the unknown depth function that allows one to obtain the depth of $\mathcal{S}_{vis}$ for each pixel in camera coordinates. The registration map $\eta\colon \mathcal{I} \to \mathcal{A}$ is an injective map that associates the points of $\mathcal{I}$ to their correspondences in $\mathcal{A}$.

B. DNN ARCHITECTURE

Figure 3 shows the general diagram of our SfT solution. It involves two steps. First, we use the *Segmentation Network*, that takes as inputs the image and template texture map and produces a pixel-wise binary segmentation map that classifies each pixel as background or object. Second, we obtain the SfT solution with the *Registration-Reconstruction Network*, that takes the image, the template texture map, and the binary mask from the previous step as inputs and obtains pixel-wise maps that represent the object's registration and depth with respect to the template. Third, we use an ARAP post-processing step to recover the occluded surface parts.

1) SEGMENTATION NETWORK

We propose a DNN model for the segmentation of the deformed template in the input image, named Deformed Template Segmentation Network (DTSNet) and defined with the