Can We Jointly Register and Reconstruct Creased Surfaces by Shape-from-Template Accurately?

Mathias Gallardo(✉), Toby Collins, and Adrien Bartoli

ISIT, UMR 6284 CNRS/Université d'Auvergne, Clermont-Ferrand, France
mathiasgallardo@gmail.com

Abstract. Shape-from-Template (SfT) aims to reconstruct a deformable object from a single image using a texture-mapped 3D model of the object in a reference position. Most existing SfT methods require well-textured surfaces that deform smoothly, which is a significant limitation. Due to the sparsity of correspondence constraint and strong regularizations, they usually fail to reconstruct strong changes of surface curvature such as surface creases. We investigate new ways to solve SfT for creased surfaces. Our main idea is to implicitly model creases with a dense mesh-based surface representation with an associated robust bending energy term, which deactivates curvature smoothing automatically where needed. Crucially, the crease locations are not required *a priori* since they emerge as the lowest-energy state during optimization. We show with real data that by combining this model with correspondence and surface boundary constraints we can successfully reconstruct creases while also preserving smooth regions.

Keywords: 3D reconstruction · Shape-from-Template · Isometry · Boundaries · Bending energy · M-estimator · Creases · Sharpness

1 Introduction

The 3D reconstruction of deformable objects from images or videos is one of the biggest open challenges in computer vision. The main difficulty arises from the variability and complexity of deformations. Two main paradigms have emerged: Non-Rigid Structure-from-Motion (NRSfM), which uses multiple images, and Shape-from-Template (SfT), which uses a single image and a *template*. The template is composed of a texture map and a model of the object's 3D shape in a reference pose. Recent works in SfT made interesting theoretical and applicative contributions in the entertainment industry with augmented reality [2–4] and real-time animation deformation transfer [5,6]. Perspectives on using SfT in medical applications were also given [7,8]. The main advantage of SfT is that it

Electronic supplementary material The online version of this chapter (doi:10.1007/978-3-319-46493-0_7) contains supplementary material, which is available to authorized users.

B. Leibe et al. (Eds.): ECCV 2016, Part IV, LNCS 9908, pp. 105–120, 2016.
DOI: 10.1007/978-3-319-46493-0_7

Fig. 1. An example of creased surface reconstruction by our method compared to a state-of-the-art method [1]. Unlike our method, [1] fails to reconstruct the creases.

needs only one image in order to achieve 3D reconstruction, which may be computed analytically [9]. SfT also registers the input image to the template, and so is well adapted to augmented reality. SfT is constrained by image data and deformation priors. The former includes feature correspondences [9–12] or (less commonly) direct pixel-wise matches [13–15]. Currently feature correspondences are the most common because they allow the problem to be solved globally. The latter includes conformity (angle preserving) [9], linear elasticity [7,8,16] and isometry (distance preserving) [9–11,17].

Most existing SfT methods break down when the surface creases, and this is for two reasons. Firstly, feature correspondences are not usually dense enough to tell us where creases occur. Secondly, most existing SfT methods use smoothed parameterizations for the surface and/or deformation to regularize the problem, which prefer smooth rather than creased solutions. The closest work to ours is [1], however this does not strictly model creases because the reconstructed creases are a by-product of the way it relaxes the isometry prior. We find that in practice it does not accurately reconstruct creases in many cases, as Fig. 1 shows. A fundamental problem with reconstructing deformable creased 3D surfaces from 2D image data is that we do not know *a priori* the crease locations. This makes it very difficult to employ existing parametric crease models used in other applications, such as b-splines, since we do not know *a priori* where to modify the spline to permit high changes in curvature. Instead our solution is to *implicitly model creases through an adaptive bending energy prior acting on a high-resolution non-parametric surface mesh*. Crucially, this does not require knowing anything *a priori* about the crease locations, since they emerge as the lowest-energy state during optimization.

While studying this problem we have found that correspondence constraints are often not sufficient data constraints. To remedy this, we complement them with a boundary constraint, which encourages the boundary of the surface to project to strong intensity edges in the image. This is a powerful constraint and should be used wherever possible. One main challenge is that we must ensure the boundary is attracted to correct image edges, which is not trivial. To deal with this we use statistical color models to help disambiguate non-boundary edges (*e.g.* from background clutter or texture). In the broader context of SfT, this is the first time that statistical color models has been exploited to solve the problem.

The paper is organized as follows. In Sect. 2, we give further background details and discuss state-of-the-art methods. In Sect. 3, we present our implicit crease energy model and its associated cost function. In Sect. 4, we present the full optimization framework. In Sect. 5, we validate our method with real data using ground truth generated by a high-precision structured-light scanner.

2 Background

We divide the background section into three parts. In the first part we discuss existing deformation models and priors in SfT. In the second part we discuss common data constraints used in SfT. In the third part we discuss how creases and surface discontinuities are modeled in other 3D reconstruction problems, then discuss a previous attempt to handle creases in [1].

2.1 Deformation Models and Priors in SfT

There are two main ways that deformations have been modeled in SfT. The first uses thin-shell models [9,11,12,17,18], where only the object's surface is modeled. The second way is volumetric models [19], where the object's surface and interior volume are modeled. Thin-shell models are the most common and give good approximations for thin or hollow surfaces made of *e.g.* paper, cloth and plastic. Most existing thin-shell models used in SfT use an approximate physical model of the object's material. The models have varied in complexity, from simple algebraic models such as smooth b-splines [18] or thin-plate splines [9,10]. Most recent methods use triangulated mesh models, which are conceptually simple and can handle general topologies [1,6,12,14,15].

Most of these methods use some forms of dimensionality reduction to reduce the problem's search space. This regularizes the problem and reduces the cost of optimization, and is based on the fundamental assumption that the surface can only deform smoothly. A significant problem with this is that when we want to reconstruct a creased surface, the deformations are non-smooth, so such dimensionality reduction will prevent the surface from being accurately reconstructed. Spline models such as b-splines and thin-plate splines reduce dimension by definition, because they model deformation with a finite set of control points. Thin-plate splines enforce global smooth deformations, and are not suitable to model surface creases. It is possible to model high-frequency and/or discontinuous deformations with b-splines by changing the spline's order and introducing repeated control points [20,21]. However, to correctly distribute the control points, one needs to know where the surface crease is, which in SfT is not known *a priori*. Other ways to reduce dimensionality have included using the eigen bases formed from the smooth modes of variation of the surface's stiffness matrix [12]. However a large number of bases are needed to model high frequency deformation such as creases, which dramatically increases the cost of optimization.

We propose to not enforce globally smooth deformations and to not apply such dimensionality reduction. Instead we use a so-called non-parametric approach where the surface is modelled by a dense triangulated mesh. We have found that creased surfaces such as folded paper can be recovered using mesh resolutions of $\mathcal{O}(10^4)$ vertices. We are able to work with such high resolution meshes because the constraints we apply on the mesh are very sparse (each constraint only applies to a small number of vertices), and this allows us to solve the resulting system iteratively with sparse linear solvers.

2.2 Data Constraints in SfT

To constrain the object's deformation data constraints must be extracted from the input image. There are three broad classes. The first and most common are *correspondence constraints* [9,11,18,22]. These match keypoints from the template's texture map and the input image using *e.g.* SURF [23], and tell us where points on the object's surface project to in the input image. The main advantages are that they do not require an initial estimate of the deformation and are reasonably fast to evaluate. However they only provide sparse constraints, and have only been used for smoothly deforming objects in the past. The second class of constraints are called *direct constraints*, and are computed directly from pixel values. They work by measuring the photometric agreement between the image and the deformed template and provide denser motion constraints than features. However direct constraints are highly non-convex and require a good initial estimate. They can also have difficulty handling strong photometric changes and when the object self-occludes or is occluded by other objects. The third type of constraints are contour constraints which are used to make the object's occluding contours align to edges in the image [24,25]. Similarly to direct constraints, these are highly non-convex constraints and require a good initial estimate. The main challenge with using contour constraints is they are difficult to apply robustly, particularly with strong background clutter.

2.3 Modelling Creases in Other Problem Domains and Previous Attempts in SfT

The problem of fitting unsmooth surfaces, including creases, has been extensively addressed in the curve [26] and surface [27–29] fitting literature. These generally address the problem of fitting 2D curves or 3D surfaces to 2D or 3D point sets respectively. Two approaches exist: one can densify the mesh [26–28] or adjust directly a model to the data [29,30]. [27] proposes the idea of tagging control points of a mesh and using subdivision surfaces to model discontinuities like creases and corners. This 3D concept is adapted to the 2D case by [26]. Another use of adaptive mesh is proposed by [28]. It starts by fitting a model to a downsampled set of points that excludes outliers. Then, to provide a better fitting, it selects new data points which have the smallest prediction residuals. The second category reconstructs 3D surfaces from 3D point cloud by having the user to select a set of global forms [30] or local shape examples [29].

The problem of reconstructing discontinuous 3D surfaces from 3D data is strongly data-driven, which is different to the SfT problem. In the SfT problem we do not have such 3D data. Instead we only have 2D projection data present in the input image, which is much weaker information. Indeed the whole reason why we require a 3D template is to form a well-posed problem by using the template's physical deformation constraints. To do this we must simultaneously register the template and reconstruct its deformed 3D shape (including creases). It would therefore particularly difficult to apply parametric models such as b-splines, because one would have to simultaneously register, reconstruct and restructure the b-spline's control points.

Practically all existing SfT methods use an ℓ_2 norm to regularize surface bending, but such norm cannot model creases because it incorrectly penalizes non-smooth solutions. The problem of creases or "sharp folds" has been looked at before in [1], via a convex formulation which maximizes the depth of each vertex and relaxes the inextensibility constraint. The inextensibility constraint preserves the geodesic distance between two vertices, but this distance may decrease when folds appear. [1] proposed to relax this constraint: the geodesic distance is replaced by the euclidean distance. Vertices may thus come closer to each other without making the surface shrink or extend. Two reasons make for this method difficult the reconstruction of 3D creases: correspondences are not sufficiently informative and it does not use smoother.

2.4 Contributions

Our main contribution is to use a high-density surface mesh model that implicitly model creases through the use of a robust smoothing regularizer known as an M-estimator. This deactivates excessive smoothing automatically during the optimization process. Crucially, crease location is not required *a priori* since it emerge as the lowest-energy state during optimization. Other problems such as optical flow [31,32] use M-estimators for handling non-smooth solutions, and this gave much inspiration. However it was unclear whether M-estimators offered a good solution to handle the SfT problem, where 3D shape has to be reconstructed from 2D data. The second main contribution is to introduce robust boundary constraint that aligns the surface boundaries to the edges in the image. Importantly, we use color information to help determine where the true surface edges are.

3 Problem Formulation

3.1 Template Definition

The template consists of a *texture map* and a non-parametric *embedding* function φ that maps the texture map to 3D camera coordinates. The texture map, denoted by $\mathfrak{I}_{\mathcal{T}} : \mathbb{R}^2 \rightarrow \{1, ..., 255\}^3$ models the color at each point on the template's surface. This can be constructed by texture-mapping the template from

one or more photographs. In the simple case when the template's surface can be seen entirely in a single calibrated image, texture-mapping is particularly simple, and can be done by inverting the image projection function, as shown in Fig. 2 (bottom left). We refer to the calibrated image as the *reference image*, and we assume the template is registered to the reference image. Here $\pi_{\mathcal{T}}$ denotes the projection function of the reference image's camera. We define the texture map's domain with $\Omega_{\mathcal{T}} \subset \mathbb{R}^2$. We define $\Omega_{B_{\mathcal{T}}} \triangleq \Omega_{\mathcal{T}}$ as the boundary points of the texture map.

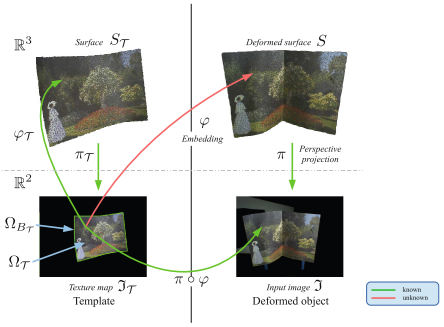

Fig. 2. Geometric setup of SfT with embedding functions.

We model the embedding function φ with a discrete dense triangular surface mesh of N vertices $\mathbf{x} = \{\mathbf{x}_1, ..., \mathbf{x}_N\} \in \mathbb{R}^{3\times N}$ and F faces $\mathscr{F} = \{f_1, ..., f_F\} \in [1, N]^{3\times F}$. We denote $E \in [1, N]^{2\times N_E}$ as the set of mesh edges where N_E is the number of edges. We embed a point $\mathbf{u} \in \mathcal{I}_{\mathcal{T}}$ using piecewise linear interpolation of the triangle vertices. Using its barycentric coordinates:

$$\varphi(\mathbf{u}; \mathbf{x}) = b_1 \mathbf{x}_{f_i(1)} + b_2 \mathbf{x}_{f_i(2)} + b_3 \mathbf{x}_{f_i(3)} \in \mathbb{R}^3, \tag{1}$$

where b_1, b_2 and $b_3 = 1 - b_1 - b_2$ are the barycentric coordinates of the point $\mathbf{x}$ on f_i. The embedding function $\varphi_{\mathcal{T}}$ gives the surface 3D reference 3D shape. The texture map and the reference image are known, then its corresponding embedding function $\varphi_{\mathcal{T}}$ is known. The embedding function φ is unknown, and the SfT problem is to determine its respective vertex positions $\mathbf{x} \subset \mathbb{R}^{3\times N}$ in camera coordinates.

3.2 Global Cost Function

We solve the problem by combining *data constraints* (correspondence and boundary constraints) and *deformation priors* (isometry and bending constraints). The cost function is as follows:

$$C(\mathbf{x}) = C_{crsp}(\mathbf{x}) + \lambda_{iso} C_{iso}(\mathbf{x}) + \lambda_{bound} C_{bound}(\mathbf{x}) + \lambda_{bend} C_{bend}(\mathbf{x}). \tag{2}$$

Correspondence Constraint. We compute initial correspondences between the texture map image and the input image with an existing method. In all presented experiments we use SURF features that are matched with the graph-based method from [33][1]. Let M be the number of point correspondences in $\Omega_{\mathcal{T}}$, denoted $\mathbf{u}_j$, $j \in \{1, ..., M\}$. Let $\mathbf{q}_j \in \mathbb{R}^2$ be the pixel normalized coordinates of the j^{th} correspondence in the input image $\mathfrak{I}$. Note that feature-matching method such as [33] are never guaranteed to be outlier free. We deal with this using a robust cost function as defined as follows:

$$C_{crsp}(\mathbf{x}) = \sum_{j=1}^{M} \rho\left((\pi \circ \varphi)(\mathbf{u}_j; \mathbf{x}) - \mathbf{q}_j\right), \tag{3}$$

where ρ is an M-estimator. This encourages the embedding function φ to project each point $\mathbf{u}_j$ onto the image at the correspondence position $\mathbf{q}_j$, but in a way that can tolerate outliers through an M-estimator ρ. We have investigated various M-estimators and found that $(\ell_1 - \ell_2)$ works well, with $\rho(\mathbf{y}) = 2(\sqrt{1 + \|\mathbf{y}\|_2^2/2} - 1)$.

Isometry Constraint. The isometry constraint is used to penalize surface extension and compression, and is required in general to make the SfT problem well posed. Following [1,34], we define the isometry constraint as:

$$C_{iso}(\mathbf{x}) = \sum_{(i,j)\in E} \left(l_{ij}^2 - \|\mathbf{x}_i - \mathbf{x}_j\|_2^2\right)^2, \tag{4}$$

where l_{ij} is the Euclidean distance between neighboring vertices (i, j) on the template's reference position.

Boundary Constraint. The aim of this constraint is to align $\Omega_{B_{\mathcal{T}}}$ to wherever it is visible to the input image. We do this by defining a *boundariness* map I_B where likely surface boundaries locations behave like potential wells. I_B is computed using image edge information and is used to define the boundary constraint as follows:

$$C_{bound}(\mathbf{x}) = \int_{\Omega_{B_{\mathcal{T}}}} \rho\left(I^B\left((\pi \circ \varphi)(\mathbf{u}_j; \mathbf{x})\right)\right) d\Omega, \tag{5}$$

where ρ is an M-estimator that we use to reduce the influence of false boundaries points on the energy function.

We base I^B on the fact that the surface's boundaries tend to coincide with strong image edges. We define I^G a blurred grayscale version of the input image $\mathfrak{I}$, which is computed using a Gaussian filter (h, σ). A naive way to compute the boundariness map would then be as follows:

$$I^B = \exp\left(-\frac{|\nabla I^G|}{\sigma^B}\right), \tag{6}$$

[1] The code is available at http://isit.u-clermont1.fr/~ab/Research/index.html.

where ∇I^G is the gradient of I^G, σ^B is a bandwidth term which governs the potential well's width. We illustrate this in Fig. 3, where an input image is shown in Fig. 3(a) and its corresponding boundariness map according to Eq. (6) is shown in Fig. 3(b). The true boundaries are represented with low potentials, but so are many false boundaries corresponding to background clutter and texture edges. This is a serious problem because they may attract the solution to a wrong local minimum.

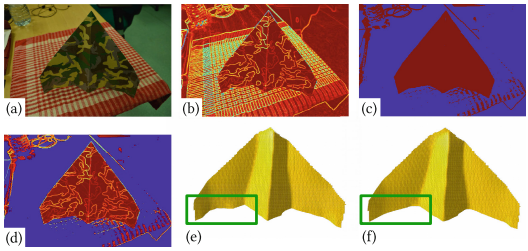

Fig. 3. (a) Input image, (b) naive boundariness map, (c) color model segmentation, (d) the enhanced boundariness map, (e) our 3D reconstruction *without* the enhanced boundariness map, (f) our 3D reconstruction *with* boundariness map using foreground color segmentation. (Color figure online)

We propose to exploit color information to significantly reduce false boundary edges. This works by applying a color-based foreground detector, trained on the target surface to each input image pixel, and setting $I^B = 1$ for any pixel which has a detection score below a threshold T_d. We train the detector using the foreground of input image (in our experiments we use an RGB Gaussian Mixture Model of 4 components) and use a default threshold of $T_d = 50$. In Fig. 3(b) and (d) we show the difference between the naive boundariness map and the boundariness map using the color-based statistical filter. Here we see that many false boundary edges in the background have been removed. Similarly to the correspondence constraint, we currently use the $(\ell_1 - \ell_2)$ M-estimator.

Bending Constraint. Our bending energy term robustly penalizes non-smooth embeddings, and is defined as follows:

$$C_{bend}(\mathbf{x}) = \int_{\Omega_\mathcal{T}} \rho\left(\frac{\partial^2 \varphi(\mathbf{u};\mathbf{x})}{\partial \mathbf{u}^2}\right) d\Omega, \tag{7}$$

where ρ is an M-estimator used to reduce the energy at creases.

4 Optimization

4.1 Overview

Equation (2) is a large-scale, sparse nonlinear optimization problem. The system is sparse because each constraint only depends on a small number of unknowns.

Regarding size, there are typically $\mathcal{O}(10^4)$ unknowns (the example in Fig. 1 has 11,557 unknowns and approximately 300,000 constraints). We propose to solve it with numerical gradient-based minimization, starting from an initial estimate $\mathbf{x}_0$. We determine $\mathbf{x}_0$ using an existing SfT method (specifically we use [10]), which does not use boundary constraints and assumes the surface is smooth. Given $\mathbf{x}_0$ our general strategy is to iteratively solve with Gauss-Newton iterations, by re-linearizing the constraints about the current estimate, then solving the corresponding linear system. For our problem sizes this can be done efficiently using sparse Cholesky decomposition, and we ensure convergence using backtracking line-search.

4.2 Improving Convergence

One caveat is that the boundary constraints are highly non-convex and can cause convergence on the wrong local minimum. The color-based filtering described in Sect. 3.2 partially deals with this, however we also introduce two more strategies. The first is to cascade the constraints, by first optimizing the solution without using boundary constraint until convergence. Once done, the detector detailed in Sect. 3.2 is trained using the region of the input image where the current estimate surface projects. Then, boundary constraints are introduced and the solution is refined. The second strategy is to use an image pyramid, which gives coarse-to-fine versions of the boundariness map and increases the convergence basin. This is a standard practice used in related problems such as optic flow. We currently use a two-level pyramid: the kernel sizes and standard-deviations are respectively $h_1 = (10, 10)$ and $\sigma_1 = 5$ and $h_2 = (5, 5)$ and $\sigma_2 = 2.5$ for a default image size of 1288×964 pixels. At the finest level, we do not apply the color-based filtering to the boundariness map. This is because assuming correct convergence, at the start of the finest level the boundaries should align reasonably closely to their true locations, and we therefore have less risk of false boundary edges steering the solution away to a wrong local minimum. The benefit is to use all edge information at the finest level, including edges where there is little color separation between the surface and its background.

5 Experimental Results

We compare the accuracy of our method with four others [1,9,10,12], which we denote respectively **ReD12**, **ReJ14**, **MDH09** and **LM16**. **MDH09** refers to the convex formulation of [1].

5.1 Ground Truth Acquisition

Some previous datasets with ground truth 3D exist [35], however these are low resolution, noisy and do not contain creased surfaces. To accurately evaluate our method new datasets with ground truth were required. We constructed three new data sets of three different objects with a highly-accurate commercial structured

light system [36]. This consists of a HD data projector and an industrial machine vision camera [37], and captures depth maps to sub-millimeter accuracy. The advantage of this setup is that the depth maps are constructed in the camera's coordinate frame, so there is no need to register them to the camera's image. RGB images were captured from the camera at a resolution of 1288×964 pixels. It takes approximately 10 s to capture an image and its associated depth map. Our dataset consists of three creased objects scanned at approximately 20 cm from the camera: a *creased paper* (6 input images), a *folded aeroplane* (9 input images) and a *cardboard box* (8 input images). The 3D reference surfaces were obtained by flattening the objects. We also evaluated the accuracy of our method on an existing smooth dataset [35], to assess how our approach coped when creased reconstruction was not required.

5.2 Implementation Details and Evaluation Metrics

For all experiments we constructed the embedding meshes by laying a triangulated 100×100 vertex grid on the reference image which was then cropped to $\Omega_{\mathcal{T}}$. We found that this resolution was sufficient to accurately reconstruct creases. Correspondences were computed using the public code from [33], which gave approximately 300 correspondences per image. We discretized $\Omega_{B_{\mathcal{T}}}$ to 1000 uniformly spaced points. For the state-of-the-art methods, there is no way to automatically optimize their free parameters. Therefore we tried our best to do this by hand, to obtain the best average error on all datasets. This was done by a search starting from the default values, and modifying each free parameter in turn to improve the average error. The values we used are found in the supplementary material. For our method, all experiments were ran using the same parameters, which were manually set.

On our new data sets we used two evaluation metrics. The first was the 3D position error (%), which was given by the average relative depth error over the region in the input image belonging to the surface. To do this, the template was fitted to the ground truth depth map, then distances between this deformed template and the estimated one were computed. The second metric was the normal error (in degrees), which was given by the average error in surface normal. To investigate the improvement at creased regions, we averaged results with two schemes. The first was to use the whole image region that corresponded to the surface. The second was using local neighborhoods around each crease, with a neighborhood distance of approximately 0.5 cm on the surface.

5.3 Results

The first part covers our investigation to see whether the choice of bending energy M-estimator had a significant impact on results (using our new data sets). The second part compares our results with the state-of-the-art methods using our new datasets. The third part compares our results with state-of-the-art methods using the dataset from [35].

Comparing Different Bending Energy M-estimators. Various M-estimators have been proposed in the literature (a good review can be found in [38]). Each have slightly different qualities and it is very hard to know *a priori* which work best for a given problem. M-estimators can either be parameterless, such as ℓ_1 or $(\ell_1 - \ell_2)$ or have at least one free parameter (usually it is only one), such as Huber and Cauchy. They may also be redescending or non-redescending. In this section we compared the performance of $(\ell_1 - \ell_2)$ and Huber, which are two non-redescending M-estimators. The purpose was two-fold. Firstly, to see if Huber's free parameter was sensitive and required careful tuning and secondly to see whether there was a significant performance difference between the two.

We run our method with 17 different bending energy M-estimator settings. The first 16 were with Huber using 16 different parameter values in the range $k = 10^2 ... 10^{-6}$. The 17^{th} was with $(\ell_1 - \ell_2)$. We evaluated performance using all input images in all three datasets. For each M-estimator setting, we run our method with 9 different bending energy weights from λ_{bend}, from 10^{-6} to 10^{-2} (which was a sufficient range), and then took the weight which produced the lowest average error. The corresponding results for all 17 M-estimator settings are shown in Fig. 4. We observe that with $(\ell_1 - \ell_2)$ we obtain very similar results to the best result obtained with Huber. The best k changes according to the error metric, but we can say reasonably that the best k varies between $k_7 = 0.05$ and $k_9 = 0.005$. This suggests that for our problem, given the optimal bending energy weight there is no clear difference between using $(\ell_1 - \ell_2)$ or Huber, and the choice for Huber's free parameter is important but not extremely sensitive in the range $k_7 = 0.05$ to $k_9 = 0.005$.

As a final experiment, we investigated the range of optimal bending energy weights for a given M-estimator. The purpose was to see how easy it is in practice to tune the bending energy weight for a given M-estimator. This was done by measuring the best bending energy weight for each test image independently, then measuring the corresponding spread. The results are shown in Fig. 4. Figure 4 shows that in general the spread of the best bending energy weight is similar for $(\ell_1 - \ell_2)$ and Huber with its parameter in the range k_7 to k_9. This implies that the difficulty of choosing a good weight for the bending energy is the same for the two M-estimators. From these experiments, we can conclude that there is very little difference in practice between using $(\ell_1 - \ell_2)$ or Huber with its parameter in the range $0.005 \leq k \leq 0.05$.

Results on Creased Datasets. For our method we used the $(\ell_1 - \ell_2)$ M-estimator for the bending energy with a corresponding weight of $\lambda_{bend} = 10^{-5}$. In Fig. 6 we show the result of the compared methods using a representative input image from each of the three datasets. In Fig. 5 we give summary statistics for each method across all images. These visual observations support the statistical results in Fig. 5. We notice that our method provides the best accuracy at the neighbors of the creases and a best global reconstruction. We remark that the large smooth regions of the surfaces are also reconstructed well in general. In Fig. 6, 5^{th} row, 3^{rd} column we show an example of a failure mode, where the

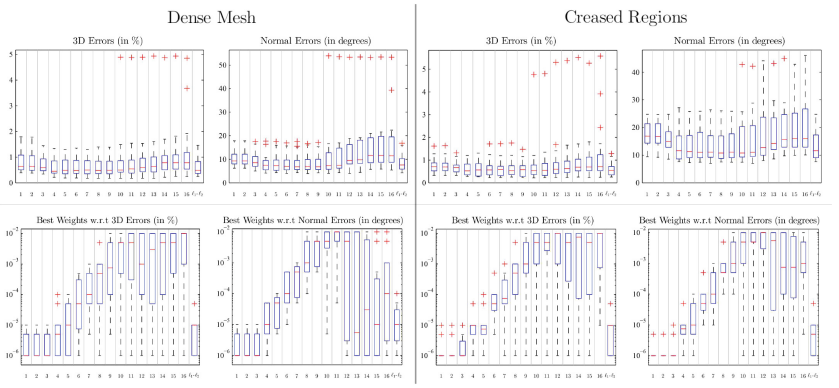

Fig. 4. Results of bending energy M-estimator analysis. We consider all input images of all the objects of our test datasets Sect. 5.3. **First row**: Errors obtained by running our method with 17 different M-estimators settings. In the first 16 settings we use Huber with 16 different hyper-parameter values. In the last setting, we use $(\ell_1 - \ell_2)$. **Second row**: the distribution of optimal weights λ_{bend} for each M-estimator setting.

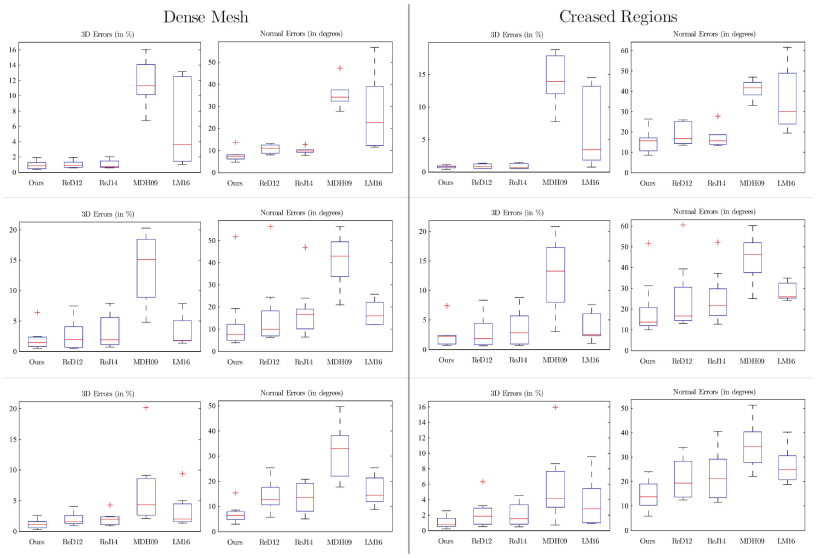

Fig. 5. Quantitative results comparison on the three new datasets. **First row**: *creased paper* dataset. **Second row**: *folded aeroplane* dataset. **Third row**: *cardboard box* dataset.

reconstruction at the bar code does not appear correct. The reason for this is because the boundary points were incorrectly fitted to the bottom of the bar code, which was compensated by the surface bending away from the camera in order to respect the isometric constraint.

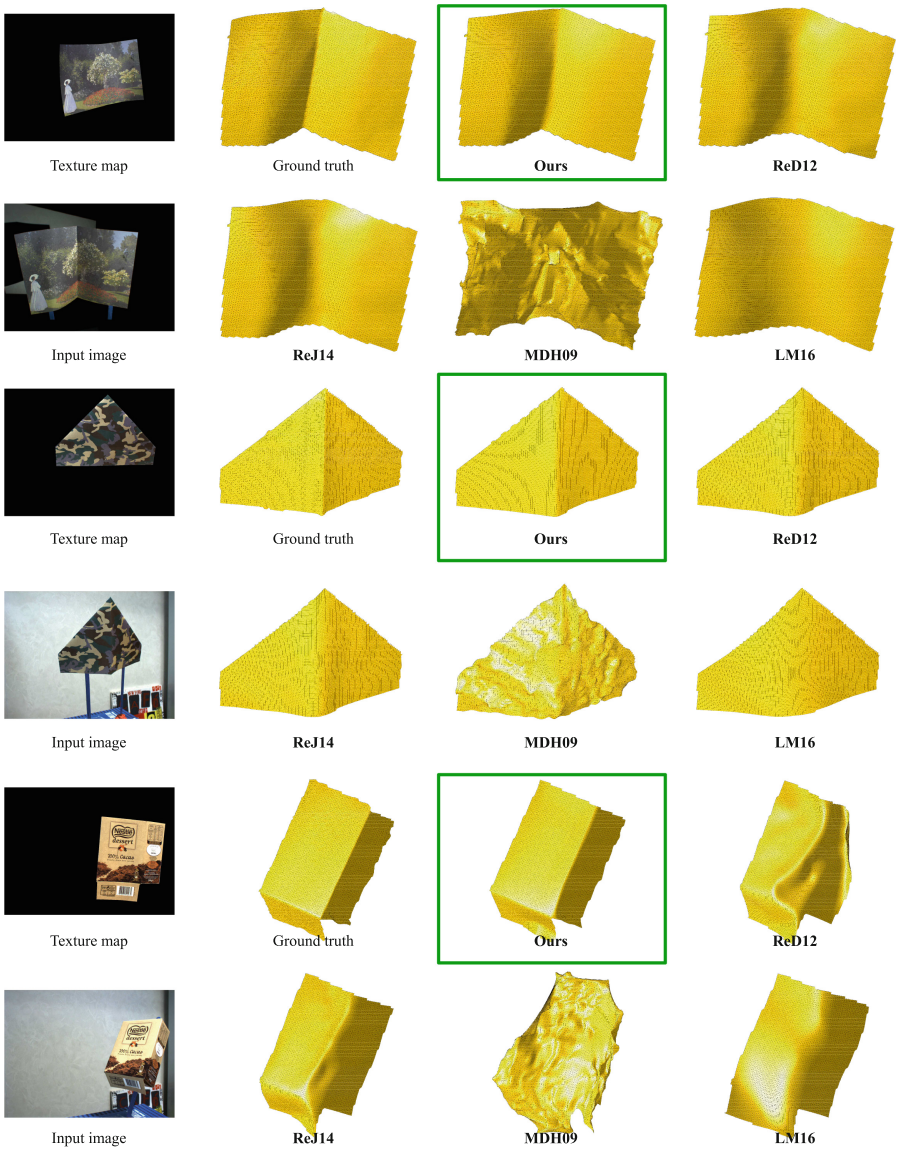

Fig. 6. Qualitative results comparison on the three new datasets. Rows $n°1$ and $n°2$: input image $n°1$ of the *creased paper* dataset. Rows $n°3$ and $n°4$: input image $n°1$ of the *folded aeroplane* dataset. Rows $n°5$ and $n°6$: input image $n°6$ of the *cardboard box* dataset.

Results on an Existing Smooth Dataset. We also tested whether our method also works well for simpler problems with smooth surfaces where ℓ_2 regularization is sufficient. We have found this to be the case with existing benchmark datasets. On the commonly-used public EPFL kinect paper dataset (193 frames) and using the same parameters, we evaluated accuracy using 40 images uniformly sampled, which produced a mean 3D error of 5.63 mm. This puts it among the best performing method, presented in [10], which uses an ℓ_2 regularization and which gives a mean 3D error of 5.74 mm.

6 Conclusion

We have developed a modeling and optimization framework for reconstructing smooth and creased 3D surfaces from a single image and a deformable 3D template. We implicitly model creases using a dense mesh-based surface representation with an associated robust bending energy term whose influence is governed by an M-estimator. We have shown that there is little difference in practice between two common M-estimators ($(\ell_1 - \ell_2)$ and Huber with a correctly set hyper-parameter), and our results indicate significantly better performance compared to previous state-of-the-art methods. An important aspect of our approach is to combine motion constraints with boundary constraints, which can significantly improve results at the surface's boundaries. One difficulty with using them is potential confusion with non-boundary image edges. We have addressed this using statistical color models, which are particularly effective when the surface's color is significantly different to the background. In future work we will extend the approach to templates with arbitrary topologies and study dynamic crease modeling.

Acknowledgments. This research has received funding from Almerys Corporation and the EUs FP7 through the ERC research grant 307483 FLEXABLE.

References

1. Salzmann, M., Fua, P.: Reconstructing sharply folding surfaces: a convex formulation. In: International Conference on Computer Vision and Pattern Recognition, pp. 1054–1061 (2009)
2. Pilet, J., Lepetit, V., Fua, P.: Fast non-rigid surface detection, registration and realistic augmentation. Int. J. Comput. Vis. **76**(2), 109–122 (2007)
3. Herling, J., Broll, W.: High-quality real-time video inpainting with PixMix. IEEE Trans. Vis. Comput. Graph. **20**(6), 866–879 (2014)
4. Magnenat, S., Ngo, D.T., Zund, F., Ryffel, M., Noris, G., Rothlin, G., Marra, A., Nitti, M., Fua, P., Gross, M., Sumner, R.: Live texturing of augmented reality characters from colored drawings. IEEE Trans. Vis. Comput. Graph. **21**(11), 1201–1210 (2015)
5. Sumner, R.W., Popović, J.: Deformation transfer for triangle meshes. ACM Trans. Graph. **23**(3), 399–405 (2004)

6. Collins, T., Bartoli, A.: Realtime Shape-from-Template: system and applications. In: International Symposium on Mixed and Augmented Reality (2015)
7. Malti, A., Hartley, R., Bartoli, A., Kim, J.: Monocular template-based 3D reconstruction of extensible surfaces with local linear elasticity. In: International Conference on Computer Vision and Pattern Recognition (2013)
8. Haouchine, N., Dequidt, J., Berger, M.O., Cotin, S.: Single view augmentation of 3D elastic objects. In: International Symposium on Mixed and Augmented Reality, pp. 229–236 (2014)
9. Bartoli, A., Gérard, Y., Chadebecq, F., Collins, T., Pizarro, D.: Shape-from-Template. IEEE Trans. Pattern Anal. Mach. Intell. **37**(10), 2099–2118 (2015)
10. Chhatkuli, A., Pizarro, D., Bartoli, A.: Stable template-based isometric 3D reconstruction in all imaging conditions by linear least-squares. In: International Conference on Computer Vision and Pattern Recognition (2014)
11. Salzmann, M., Fua, P.: Linear local models for monocular reconstruction of deformable surfaces. IEEE Trans. Pattern Anal. Mach. Intell. **33**(5), 931–944 (2011)
12. Ngo, D.T., Östlund, J., Fua, P.: Template-based monocular 3D shape recovery using Laplacian meshes. IEEE Trans. Pattern Anal. Mach. Intell. **38**(1), 172–187 (2016)
13. Malti, A., Bartoli, A., Collins, T.: A pixel-based approach to template-based monocular 3D reconstruction of deformable surfaces. In: Proceedings of the IEEE International Workshop on Dynamic Shape Capture and Analysis at ICCV, pp. 1650–1657, November 2011
14. Ngo, T.D., Park, S., Jorstad, A.A., Crivellaro, A., Yoo, C., Fua, P.: Dense image registration and deformable surface reconstruction in presence of occlusions and minimal texture. In: International Conference on Computer Vision (2015)
15. Yu, R., Russell, C., Campbell, N.D.F., Agapito, L.: Direct, dense, and deformable: template-based non-rigid 3D reconstruction from RGB video. In: ICCV (2015)
16. Malti, A., Bartoli, A., Hartley, R.I.: A linear least-squares solution to elastic Shape-from-Template. In: International Conference on Computer Vision and Pattern Recognition (2015)
17. Collins, T., Bartoli, A.: Using isometry to classify correct/incorrect 3D-2D correspondences. In: Fleet, D., Pajdla, T., Schiele, B., Tuytelaars, T. (eds.) ECCV 2014. LNCS, vol. 8692, pp. 325–340. Springer, Heidelberg (2014). doi:10.1007/978-3-319-10593-2_22
18. Brunet, F., Hartley, R., Bartoli, A.: Monocular Template-Based 3D surface reconstruction: convex inextensible and nonconvex isometric methods. Computer Vis. Image Underst. **125**, 138–154 (2014)
19. Parashar, S., Pizarro, D., Bartoli, A., Collins, T.: As-rigid-as-possible volumetric Shape-from-Template. In: IEEE International Conference on Computer Vision (2015)
20. Gregorski, B.F., Hamann, B., Joy, K.I.: Reconstruction of B-spline surfaces from scattered data points. In: Computer Graphics International, 2000. Proceedings, pp. 163–170 (2000)
21. He, Y., Qin, H.: Surface reconstruction with triangular B-splines. In: Geometric Modeling and Processing, 2004. Proceedings, pp. 279–287 (2004)
22. Ostlund, J., Varol, A., Ngo, T., Fua., P.: Laplacian meshes for monocular 3D shape recovery. In: European Conference on Computer Vision (2012)
23. Bay, H., Ess, A., Tuytelaars, T., Van Gool, L.: Speeded-Up Robust Features (SURF). Comput. Vis. Image Underst. **110**(3), 346–359 (2008)

24. Ilić, S., Salzmann, M., Fua, P.: Implicit meshes for effective Silhouette handling. Int. J. Comput. Vis. **72**(2), 159–178 (2006)
25. Vicente, S., Agapito, L.: Balloon shapes: reconstructing and deforming objects with volume from images. In: 2013 International Conference on 3D Vision - 3DV 2013, June 2013
26. Kaess, M., Dellaert, F.: Reconstruction of objects with jagged edges through Rao-Blackwellized fitting of piecewise smooth subdivision curves. In: Proceedings of the First IEEE International Workshop on Higher-Level Knowledge in 3D Modeling and Motion Analysis, pp. 39–47 (2003)
27. Hoppe, H., DeRose, T., Duchamp, T., Halstead, M., Jin, H., McDonald, J., Schweitzer, J., Stuetzle, W.: Piecewise smooth surface reconstruction. In: Proceedings of the 21st Annual Conference on Computer Graphics and Interactive Techniques, pp. 295–302 (1994)
28. Fleishman, S., Cohen-Or, D., Silva, C.T.: Robust moving least-squares fitting with sharp features. ACM Trans. Graph. **24**(3), 544–552 (2005)
29. Gal, R., Shamir, A., Hassner, T., Pauly, M., Cohen-Or, D.: Surface reconstruction using local shape priors. In: Proceedings of the Fifth Eurographics Symposium on Geometry Processing. Eurographics Association, pp. 253–262 (2007)
30. Pauly, M., Mitra, N.J., Giesen, J., Gross, M., Guibas, L.J.: Example-based 3D scan completion. In: Proceedings of the Third Eurographics Symposium on Geometry Processing (2005)
31. Black, M.J., Anandan, P.: A Framework for the robust estimation of optical flow. In: Proceedings of Fourth International Conference on Computer Vision, pp. 231–236, May 1993
32. Zach, C., Pock, T., Bischof, H.: A duality based approach for realtime TV-L1 optical flow. In: Hamprecht, F.A., Schnörr, C., Jähne, B. (eds.) DAGM 2007. LNCS, vol. 4713, pp. 214–223. Springer, Heidelberg (2007). doi:10.1007/978-3-540-74936-3_22
33. Collins, T., Mesejo, P., Bartoli, A.: An analysis of errors in graph-based keypoint matching and proposed solutions. In: European Conference on Computer Vision (2014)
34. Perriollat, M., Hartley, R., Bartoli, A.: Monocular template-based reconstruction of inextensible surfaces. Int. J. Comput. Vis. **95**(2), 124–137 (2011)
35. Salzmann, M., Hartley, R., Fua, P.: Convex optimization for deformable surface 3D Tracking. In: International Conference on Computer Vision (2007)
36. David 3D Scanner (2014). http://www.david-3d.com/en/products/david4
37. Grey, P.: Flea2G 1.3 MP Color Firewire 1394b (Sony ICX445). https://www.ptgrey.com/
38. Zhang, Z.: Parameter estimation techniques: a tutorial with application to conic fitting. Image Vis. Comput, **15**(1), 59–76 (1997)