HAL
open science

Texturing and inpainting a complete tubular 3D object reconstructed from partial views

Julien Fayer, Bastien Durix, Simone Gasparini, Géraldine Morin

To cite this version:

Julien Fayer, Bastien Durix, Simone Gasparini, Géraldine Morin. Texturing and inpainting a complete tubular 3D object reconstructed from partial views. Computers and Graphics, 2018, 74, pp.126 - 136. 10.1016/j.cag.2018.05.012 . hal-01804180

HAL Id: hal-01804180

https://hal.science/hal-01804180v1

Submitted on 31 May 2018

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Texturing and inpainting a complete tubular 3D object reconstructed from partial views

Julien Fayer[a,b], Bastien Durix[a], Simone Gasparini[a,*], Géraldine Morin[a]

[a] *University of Toulouse, Toulouse INP – IRIT; 2 rue Camichel, 31000 – Toulouse (France)*
[b] *SAS InnerSense, 10 Avenue de l'Europe, 31520 Ramonville-Saint-Agne, France*

Abstract

We present a novel approach to texture 3D tubular objects reconstructed from partial views. Starting from few images of the object, we rely on a 3D reconstruction approach that provides a representation of the object based on a composition of several parametric surfaces, more specifically *canal surfaces*. Such representation enables a complete reconstruction of the object even for the parts that are hidden or not visible by the input images. The proposed texturing method maps the input images on the parametric surface of the object and complete parts of the surface not visible in any image through an inpainting process. In particular, we first propose a method to select, for each 3D canal surface, the most suitable images and fuse them together for texturing the surface. This process is regulated by a confidence criterion that selects images based on their position and orientation w.r.t. the surface. We also introduce a global method to fuse the images taking into account their exposure difference. Finally, we propose two methods to complete or inpaint the texture in the hidden parts of the surface according to the type of the texture.

Keywords: Texturing, Parametric Surfaces, 3D Reconstruction

1. Introduction

In the last decade many methods and approaches have been proposed to generate a 3D model of an object from a set of images. Most approaches are based on Structure-from-Motion and Multi-View Stereo (MVS) [1], which enables the reconstruction of the object from an unordered set of images [2]. These methods perform well if the object is sufficiently textured, so that anchors (interest points) can be found for creating correspondences among the images. The geometric model generated by these classic reconstruction methods is generally a 3D point cloud, which is then triangulated to generate a triangle mesh. The final stage, *texturing*, aims at providing a consistent texture for the mesh from the multiple source images, and, in particular, at insuring a consistent texturing across neighbor triangles of the mesh [3]. Note that, in order to obtain a good quality model the whole object needs to be covered by sufficiently many images taken under different points of view.

In this work, we deal with the reconstruction of a specific family of objects that can be represented by a set of canal surfaces (*branches*) [4]. In particular, we build upon the geometric reconstruction method for tubular objects recently proposed by Durix *et al*. [5, 6] (see Fig. 1): from a limited number (usually, from two to five) of calibrated images (Fig. 1a) they generate a geometric model of the object that is composed by a set of parametric canal surfaces (Fig. 1b), *i.e*. a piecewise canal surface model. One of the advantages of this reconstruction method is that a full reconstruction of the object can be obtained with few images, not necessarily covering the whole space around the object. Moreover, it does not require good quality images nor elaborate calibration, and it is able to reconstruct objects, even if they have a uniform texture (*c.f*. Fig. 1). We propose to extend and complete their pipeline by **texturing the reconstructed geometric model**. The major problem to address is the completion of the texture for the parts of the object that are not covered by the input images or that are hidden because of occlusions. We propose a texturing method that maps the input images on the parametric surface of the object and complete parts of the surface not visible in any input image through an inpainting process. Similarly to the classic texturing technique, we propose a novel method to select, for each 3D canal surface, the most suitable images and fuse them together for texturing the surface. This process is regulated by a confidence criterion that selects images based on their position and orientation w.r.t. the surface (Fig. 1c). We then propose two methods to complete the texture in the hidden parts of the surface according to the type of the texture (Fig. 1d). Our method is based on a global optimization process that fuse the images taking into account their exposure difference, and correcting misalignment. Once integrated in the original pipeline, a full textured 3D model can be generated from a few input images, possibly not covering the whole object (Fig. 1e).

The advantages of the proposed approach are the generation of a full textured 3D model from a few input images, possibly not covering the whole object. The major contributions of this work are (*i*) the texturing of each branch of the model from the input images choosing the most suitable image, (*ii*) the fusion of the texture from different images with exposure and alignment correction, and (*iii*) two methods to complete the texture for the parts of the branches that are not seen by any camera (see Fig. 2b).

The paper is organized as follows. Section 2 overviews the

*Corresponding author: simone.gasparini@irit.fr

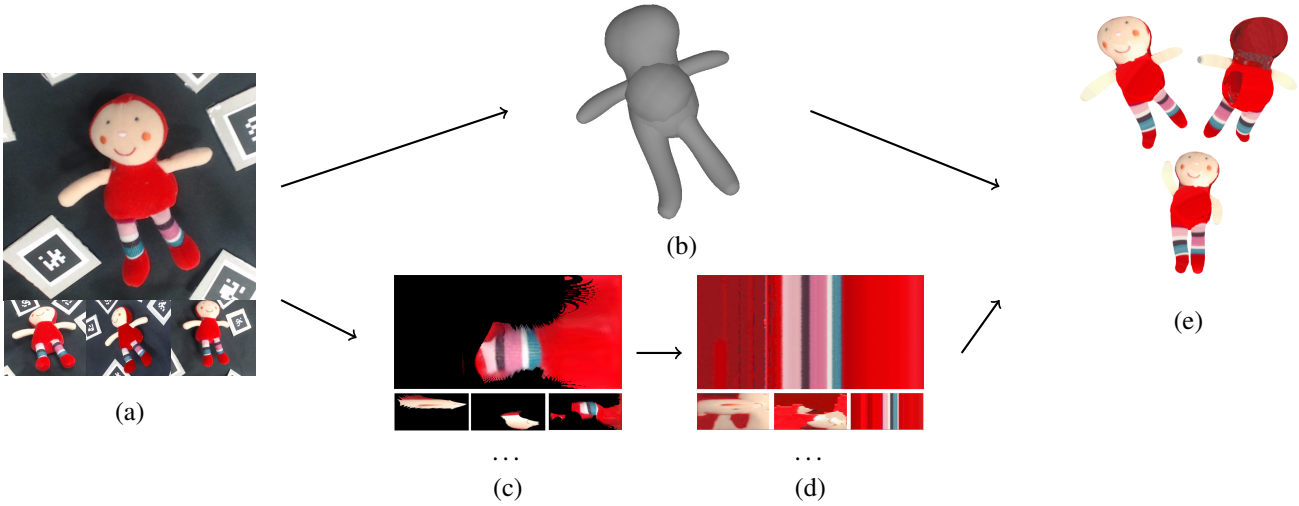


Figure 1: Illustration of the textured reconstruction pipeline. (a) Calibrated acquisitions of the object to reconstruct. (b) The object is reconstructed with its skeleton. (c) Apparent textures are extracted from the images for each branch of the model. (d) As those textures are partial (for example, we can not see the back of the plush here), they are completed. (e) The completed textures are applied to the object, that can be easily animated.

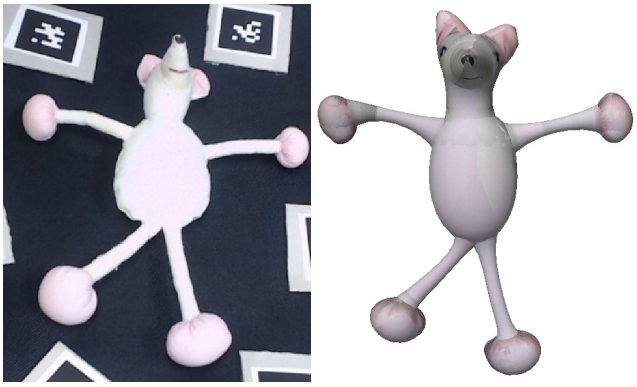


Figure 2: A plush with a very uniform texture reconstructed and textured. Here, only the front texture (a) is visible on the images, and the rear texture (b) is estimated by the described method. Note that despite the lack of interest points on the initial model, the whole object is reconstructed.

state of the art of texturing reconstructed 3D models; Section 3 presents the main steps of the proposed pipeline and Section 4 details the proposed method for adjusting the exposure of the texture and completing the missing parts. Section 5 presents some preliminary results and a discussion of the limitations, while Section 6 concludes the paper with future directions and improvements of the proposed method.

2. Related works

In this section we review the state of the art for texturing 3D models generated by different approaches, and then we introduce the most relevant approaches for inpainting.

2.1. *Texturing 3D models generated by MVS*

As mentioned in the Section 1, the 3D reconstruction approach that gives more promising results is the Multi-View Stereo (MVS) [1]. Once the 3D mesh model has been generated, the last step of the pipeline is the texturing of the mesh, *i.e.* assigning a color to each face of the mesh. Often, as pre-processing step, the model is decimated before texturing in order to get larger triangles: while the overall geometry can be maintained with a sufficient accuracy, the larger patches can make the texturing process more efficient and effective [7]. The main problem to address when assigning a texture to a face is the selection of more suitable source image. Thus, texturing can be seen as the problem of selecting the best view(s) for each face, taking into account different parameters, such as the distance of the image w.r.t. the face, the angle under which the face is seen by the image and, more generally, the quality of the image (blur effects, lighting *etc.*). Moreover, in order to get a photo-realistic 3D model, the texture map applied to the model should be independent of the light conditions under which the original images were taken. The texture map should be rendered properly when the model is shown with a different lighting. This requires to normalize and register the original images, for example by optimizing their color consistency [8] or maximizing the mutual information between the projected images [9].

Texturing methods can be roughly divided into two main approaches. *Single-view* approaches select, for each face, independently the best view [10]. This solution is only optimal locally, as it inevitably generates visible artifacts and discontinuity seams among neighbour faces having different associated images with possibly different exposure or lighting. Blending the images at face borders may mitigate the problem, otherwise more advanced techniques use labeling [11] or energy minimization that penalizes discontinuities [3]. *Multi-view* approaches, instead, blend together a subset of the source

images in order to get a more uniform and consistent image. This global approach may suffer of loss of quality and of details when blending together images taken at different distances from the faces. This requires to adopt a weighted blending that favors images that are closer to the model [12]. Another issue is related to the imperfect estimation of the geometry and the alignment of the cameras (*e.g*. camera calibration), which, again, may generate artifacts in the blended image. To overcome this, [13] proposed a patch based synthesis in which a synthetic view is generated from two or more images, taking into account misalignments while preserving the photometric consistency.

2.2. Texturing 3D models generated by other reconstruction methods

Other reconstruction methods use the silhouette of the object [14], but they require a precise calibration (requiring most of the time a dedicated capturing environment) and a larger number of images (at least 20). The generated model, the visual hull, is piecewise linear, and does not provide any parametrization. Texturing from a reference image is thus difficult, as no knowledge of the surface within the silhouette corresponding to the view point is given.

On the other hand, man-made objects are often complex shapes that can be decomposed into simpler shapes and rotationally-symmetric surfaces, *e.g*. tubular, which can be modeled and reconstructed more easily with parametric surfaces.

Our approach is similar in spirit to Chen *et al*. [15], which generates a 3D model from a single image of the object with the interaction of the user. Chen *et al*. segment a complex shape of the object into smaller and simpler parts guided by a series of "sweeps" gestures: these allow the user to define two dimensions of a 2D profile and "sweeping" it along the curved axis of the object. They recover the texture from the image by back-projection, for the occluded parts two approaches are proposed. Under the assumption that the object is symmetric, the visible texture is mirrored and mapped to the occluded parts. Textureless regions may still exist for symmetrical points both out of the sight of the camera. In this case inpainting is used to complete the texture [16].

Rather than relying on an accurate geometry, we instead rely on the reconstruction of the geometry and the topology of the object based on skeletons [5, 6]. Texturing models obtained with a parametric reconstruction faces other challenges. Since the model approximates the geometry of the real object, camera and geometry misalignments can be more severe than the classic pipelines, and they must be taken into account when blending and registering the images. While MVS can only reconstruct what is visible by the cameras, parametric reconstruction can reconstruct occluded parts of the object for which the photometric information is thus unavailable. In that case, texturing needs to fill the "holes" of those parts. Inpainting [17] can be used to "hallucinate" the regions without texture by propagating the texture of the neighbor regions.

2.3. Inpainting

There are two main approaches for inpainting techniques [17]. *Diffusion-based methods* [18] are used in image restoration to fill or correct small regions of the images for which a mask is provided by the user. These methods are generally based on Partial Differential Equations (PDEs) and a diffusion model that iteratively propagate the information from the outside of the mask along the isophotes, *i.e*. the level lines perpendicular to the gradient vectors of the pixels on the contour. These methods perform well when filling small and smooth regions but are not adapted if a structure or texture needs to be propagated. Moreover, these methods, being iterative, have a high computational cost.

Patch-based methods [19, 20] are instead used to fill larger portions of the image by copying either single pixels (*sparsity-based* [21]), entire patches or a mixture of those from other parts of the image (*exemplar-based* [19]). For each pixel p of the mask, they search the most similar patch in the image to the one centered in p, and they copy it. The search for this similar patch is the most important but also the most expensive step of the algorithm. Many variants and optimizations have been proposed over the last decade. One of the most effective approaches is PatchMatch [22], which efficiently finds for every patch the approximate nearest-neighbour in the image using a randomized cooperative hill climbing strategy. In [23], the search is restricted to the most likely offsets, reducing the complexity and also enhancing the propagation of the geometric structures of the image. The other critical step in patch-based methods is the selection of p and the order of filling. *Onion-peel* order fills the missing data starting from the pixels on the border and proceeding layer after layer towards the region's center. This sometimes leads to unexpected results at the center of the region and, in general, structures are not propagated inside the region. *Structure-aware* methods, instead, give priority to pixels lying on borders of objects, thus favoring the preservation of structures. On the other hand, a known issue of the PatchMatch approach is that it does not properly handle regular textures, *i.e*., textures embedding regular patterns or structures. Other methods have been proposed to handle regular textures by performing a *statistical analysis* of the texture that allows to find the map of the dominant directions (or *offsets*) [24, 25]: the inpainting problem is then cast as a global minimization of an energy function written in terms of an offset map that enforces the structure and texture consistency. In [26], the minimization problem is solved via *graph cuts* in order to reduce the computational complexity.

3. Surface parametrization and texturing

In the following sections we present our pipeline for texturing a 3D model reconstructed from a set of n calibrated images $(I_i)_{i=1\ldots n}$.

3.1. Reconstruction of the geometry

We start from a reconstruction based on skeletons [6], which generates a set of canal surfaces representing the captured 3D

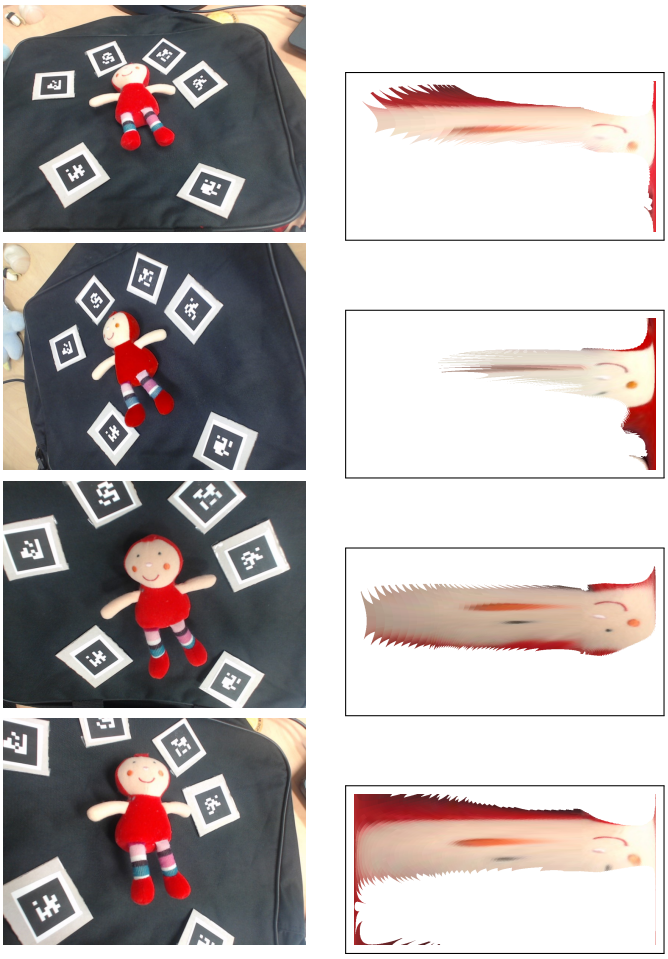


Figure 3: For a given branch, in this case one of the branches composing the face of the puppet, we can project the each image I_i (first column) of the branch into a (partial) image I_i^b (second column). The final texture image I_b for the branch can be obtained by fusing together these images as described in Section 3.4.

object [27]. This reconstruction is based on an estimation of the projection of the skeleton of the 3D object on each image; using such skeleton correspondences, the 3D skeleton is estimated. First, a shape is captured on several images and is segmented on each image with the semi-interactive algorithm GrabCut [28]. Then, perspective skeletons are computed on each image, and the user associates the extremities of each skeleton. Finally, a 3D skeleton of the object is estimated. Each branch is reconstructed separately, as a canal surface. The skeleton is used here as a set of parametric surfaces, such as canal surfaces [4], that approximates the complex shape of the real object. A canal surface is the envelope of a family of spheres, which centers and radii vary along a continuous curve. Intuitively, the 3D objects reconstructed have a curve medial axis. Canal surfaces have a natural regular parametrization of their surfaces [29], thus enabling texture mapping.

3.2. A parametric domain for the texture of a branch

For each branch b of the model, we aim at creating a reference image texture I_b. Each branch of the model is represented by a parametric canal surface $S(t,\theta)$, where t is moving along the skeleton curve C, and θ is turning around the skeleton point $C(t)$ on the surface. If t varies in an interval A so that $\{C(t), t \in A\}$ describes the entire skeleton curve, w.l.o.g. we can take $A = [0,1]$. Thus, $A \times [0,2\pi]$ is the parametric domain of the canal surface. Our goal is to reconstruct a complete image I_b on the domain $A \times [0,2\pi]$[1].

Then, for each image I_i used for the geometric reconstruction of the branch, we can project (since we assume the camera is calibrated) the surface of the branch onto the image (see Fig. 3). Each pixel of I_i covered by the branch projection corresponds to at least one 3D point $S(t,\theta)$. Note that, as the branches are generated by triangulation of a 2D skeleton from different images, the back-projection does not exactly fit the image. However, since the mask of the object on each input image is known from the skeleton-based reconstruction, we only use pixels from inside the mask to estimate the texture of each branch. This mask filtering avoids considering background pixels.

3.3. Handling occlusions

As the model is composed of different canal surfaces, one of the challenges is to correctly identify the texture belonging to a branch b, for example in case of (self-) occlusions among the different parts of the object. To that end, we rely on a z-buffer: we project the points back on the reference image I_i, to identify the point $S(t,\theta)$ closest to the viewpoint, that is, determining the parameters (t,θ) of the point $S(t,\theta)$ visible on I_i. So, each point of $S(t,\theta)$ visible on I_i gets its color from image I_i. Then, we generate a label image, similar to a z-buffer, such that each pixel has the label of the canal surface closest to the viewpoint. However, as each reconstructed object is a combination of several canal surfaces, some canal surfaces are partially inside others. Thus, these hidden surfaces do not get a color from any image, which is a loss of useful information for the completion of the texture. Painting all the hidden surfaces, independently of the depth is wrong too. To assign a color each surface coherently, we paint each point behind the visible point at distance less than a chosen threshold ϵ with the color of the visible point.

3.4. Texturing a branch from multiple images

As discussed in Section 2, for each point of the surface there may be several images from which the texture can be selected. In our approach, given the set V of images I_i in which a point of the surface is visible, we apply the texture of the best image in the set that fits the surface S. For that, we use the confidence criterion defined in [30] and originally used for inpainting.

This criterion gives for each pixel from the image a score named *trust* based on two parameters: the distance from the viewpoint to the tangent plane in the surface where the pixel is projected and the angle of inclination between the normal to the surface and the camera axis.

[1] Note that in the actual implementation we consider a sampled, discrete domain for $A \times [0,2\pi]$.

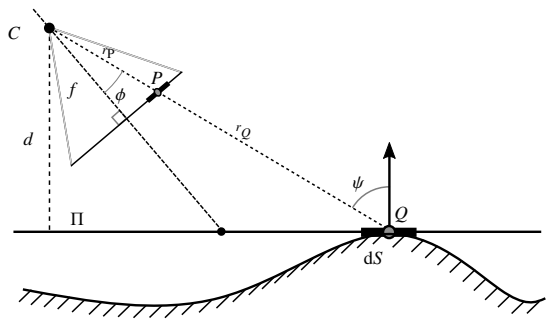


Figure 4: Confidence criterion applied to a viewpoint C and to a surface dS with tangent plane Π at point Q.

We define for the center P_i of each pixel p of the image $I_i \in V$, the confidence function $trust(P_i)$ that depends on the distance from the viewpoint of the image I_i, and the normal to S at parameters (t, θ):

$$\text{trust}(P_i) = \left(\frac{f_i}{d_i}\right)^2 \left(\frac{\cos\psi}{\cos\phi}\right)^3 \tag{1}$$

where f_i is the focal length of the camera, d_i its distance from the tangent plane Π in $S(t, \theta)$, ϕ the angle between the camera axis and the normal to $S(t, \theta)$ and ψ the angle between the projection ray of the point P_i and the normal of the plane (*c.f.* Fig. 4).

The value of trust decreases as the distance d_i increases and also as the angle ψ increases. Higher values of trust are pixels of good quality, whereas lower values denote points that are likely to represent regions of the plane far from the camera and/or seen under a very skew angle.

Based on this criterion, for each viewpoint i, we create a confidence map C_i for each pixel of the reference image domain $A \times [0, 2\pi] \subset \mathbb{N}^2$:

$$C_i : \begin{cases} A \times [0, 2\pi] & \longrightarrow \mathbb{R} \\ p & \longmapsto C_i(p) = \text{trust}(P_i) \end{cases} \tag{2}$$

with $C_i(p) = 0$ if p is not a projection of a point of the plane Π. Fig. 5 shows an example of two confidence maps corresponding to two distinct viewpoints, with the second image (c) having better trust values than the first image (a).

Then, we construct the reference image I_b of the branch b by selecting the pixels having the highest value of confidence:

$$I_b(p) = I_i(p), \quad i = \arg\max_i C_i(p).$$

The resulting image merges together all the textures from different viewpoints. However, as shown in Fig. 5, there may be regions of the image for which no texture can be retrieved as they are not seen by any camera. Moreover, artifacts may exist due to the different expositions of the images and their misalignment.

4. Improving texture

In the previous section, we showed how we map existing textures into the geometric model, how to determine the correct branch, and how to choose the best image according to the geometry. As stated in Section 3 some issues still affect the quality of our texturing. First, textures from different images may be misaligned when mapped in the reference image due to geometric approximation. On one hand, our 3D model implicitly only approximates the geometry of the real object by a set of non degenerate canal surfaces. On the other hand, a 3D branch

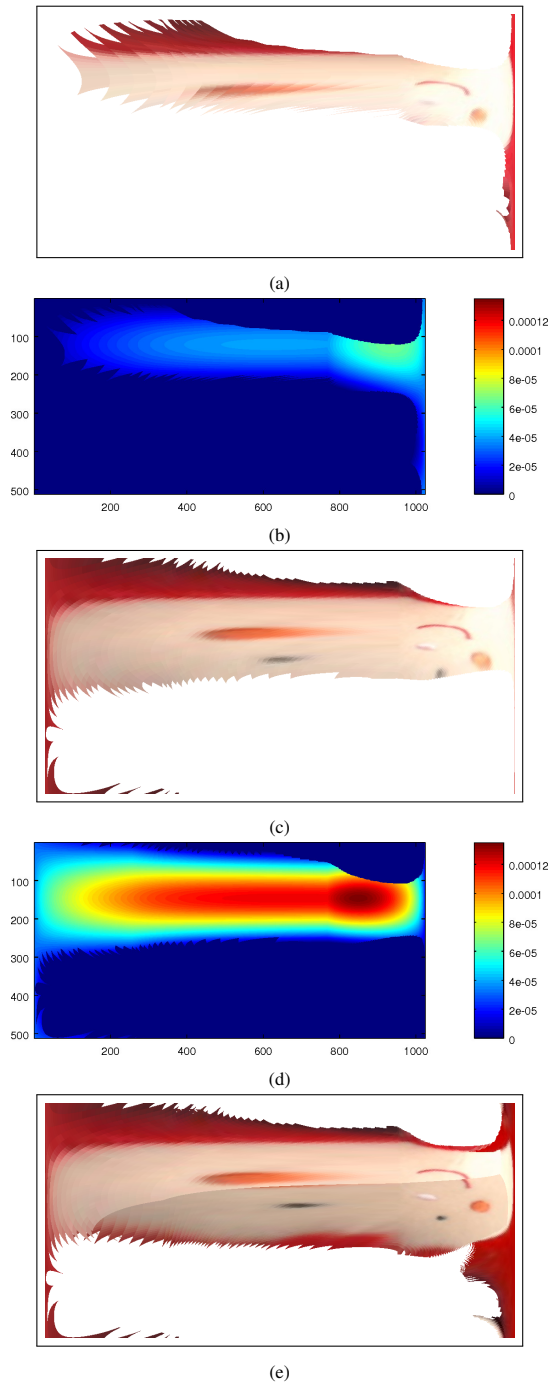


Figure 5: Confidence maps of the head branch b for two different viewpoints (out of four, not shown): (a,c) the reference images I_i^b generated independently for each image I_i; (b,d) the corresponding confidence maps where higher confidence values are depicted with the Jet colormap; (e) the final reference image I_b that fuses together all four original views.

is computed by a least square triangulation process [6]. For these two reasons the reconstructed geometry is approximate and thus offset the texture when back projected in parameter space. Classical reconstruction uses a least square triangulation to reconstruct a 3D point from its projection in several images [31], here, the same triangulation process is applied for reconstructing a sphere, that is, a 3D point and a radius. Textures from different images may have lighting discontinuities, due to different exposure or different light conditions. Finally, some parts of the model may not be visible on any images and the relevant texture need to be generated. The next sections introduce some adjustment on the textures that we put in place to overcome these issues.

4.1. Correcting exposure

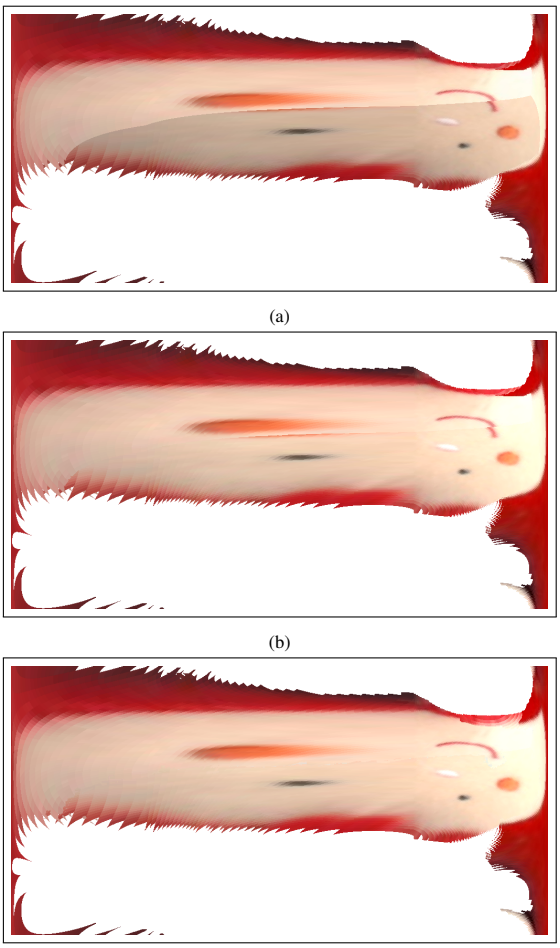
We take into account the different exposure of the images by applying sequentially a local and a global adjustment. The local adjustment is applied separately for each branch of the model. Then a global optimization of the exposure is computed for all the images.

Local exposure correction. When building the reference image for a branch, different images may have different exposures, thus leading to color and brightness discontinuities in the final image (see Fig. 7). To cope with the difference in exposure, we apply a radiometric calibration of exposure derived from [32]. Given a branch b and its set V of images I_i in which b is visible, we generate a set of images I_i^b, parametrized in the same domain $A \times [0, 2\pi]$ as I_b. Note that, in general (up to alignment errors), each pixel $I_i^b(p)$ is the same pixel for each i, *i.e.* it represents the same point of the surface, possibly with a different color value. If for some images of V the value of $I_i^b(p)$ is not defined because, *e.g.*, the point is not visible or occluded, we set its value to an arbitrary value of 0. We need now to fuse together these images in order to obtain I_b while adjusting and correcting the exposure. We formulate this problem as the following non-linear optimization problem:

$$\min_{\alpha_i, r(p)} \sum_{i,p} \left(I_i^b(p) - \alpha_i \, r(p) \right)^2 , \tag{3}$$

where α_i is the exposure coefficient for I_i and $r(p)$ is the pixel radiance value. Fig. 7 shows a visual comparison between the original and normalized exposure.

Global exposure correction. Once that for each branch b independently, exposures in images $(I_i^b) \in V$ have been corrected, we adjust the exposure of the entire 3D object. Pixels of a same image I_i can be projected onto different branches, that is, different domains $I_i^{b_1}$ and $I_i^{b_2}$ inherit different exposure correction parameters. Thus, exposure may vary on adjacent parts of the 3D model. To avoid this situation, we adjust the different exposure corrections so that, on a single image I_i the variance is minimized. Let us define for n images and p branches the $n \times p$ matrix $A = \left(\alpha_{i,j}\right)$, where $\alpha_{i,j}$ is the exposure correction of the i-th image in the j-th mesh. Then, for each image I_i we estimate a coefficient β_i that minimizes the variance of the $\alpha_{i,j}$ over all the branches j. More formally, we consider a diagonal matrix $X = \mathrm{diag}\left(\beta_1, \beta_2, \ldots, \beta_p\right)$ that, when multiplied by A, minimizes



(a)

(b)

(c)

Figure 6: The correction of the exposure and the registration on a global reference image: (a) the reference image of a branch without exposure correction showing a clear discontinuity between the texture of two images; (b) the reference image with the exposure correction leading to a smoother transition between the images; (c) the image of a branch with registration corrected: smile and cheek are now continuous.

the variance; we then want to find the set of values $\hat{X}$ such that:

$$\hat{X} = \arg\min_{X} \left(\sum_i \mathrm{Var}((AX)_i) \right), \tag{4}$$

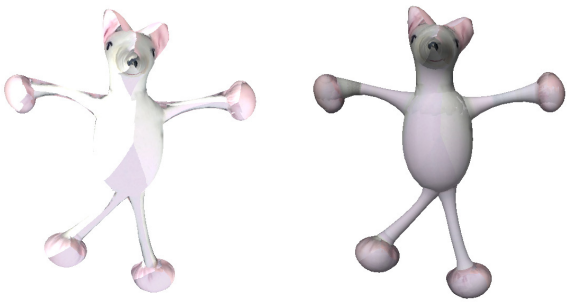
where $(AX)_i$ is the i-th row of AX. We then update the texture of each mesh j with the new exposure coefficient $\alpha_{j,new} = \hat{\beta}_j \, \alpha_j$. Fig. 8 shows the difference between a render without global correction (left) an with the global correction (right).

4.2. Correcting texture misalignment

Due to difference in the model or calibration errors, geometric misalignment of the branches may occur and cause discontinuities in the final texture image I_b, in regions where the texture comes from different views with close values of trust (as illustrated on Fig. 6b). To handle that, we consider the registration as an energy minimization by graph cut [33]. For a pixel p of the image texture I_b, we assign a label l corresponding to

(a) (b)

Figure 7: The artifact caused by the difference of exposures in images: (a) the lower part of the face of this character is brighter than the upper part; (b) after correction of the exposure, the local correction leads to a coherent color between the two parts of the face, the global correction fixes the difference of color between the two arms.



(a) (b)

Figure 8: Generated texture without (a) and with (b) global correction of the exposure. The artifacts caused by the differences on the exposure on each image are handled by the global correction.

the view from which p has been taken (as explained in Section 3.3). Then the following energy is minimized:

$$E = \sum_{p \in I_b} E_d(p, l) + \lambda \sum_{(p_1, p_2) \in \mathcal{N}} E_r(p_1, p_2, l_1, l_2), \tag{5}$$

where E_d is the data term, E_r is the regularization term and λ is a parameter that regulate the importance of the two energy functions. In the data term, we penalize the labels with a poor trust for the pixel p. Thus, we set E_d as:

$$E_d(p, l) = f(C_l(p)), \tag{6}$$

where f is a decreasing function for C_l. The regularization term should penalize the choice of neighbor pixel belonging to different views, in order to ensure the coherence of the pixels in the same region. We considered a similar function as the one defined in [24]:

$$E_r(p_1, p_2, l_1, l_2) = \left\| I_{l_1}(p_1) - I_{l_2}(p_1) \right\| + \left\| I_{l_1}(p_2) - I_{l_2}(p_2) \right\|. \tag{7}$$

We use the computed image explained in Section 3.4 as initialization of the labeling of each pixel p of I_b. Fig. 6c shows the texture of the branch after the energy minimization, discontinuities along the mouth have been fixed.

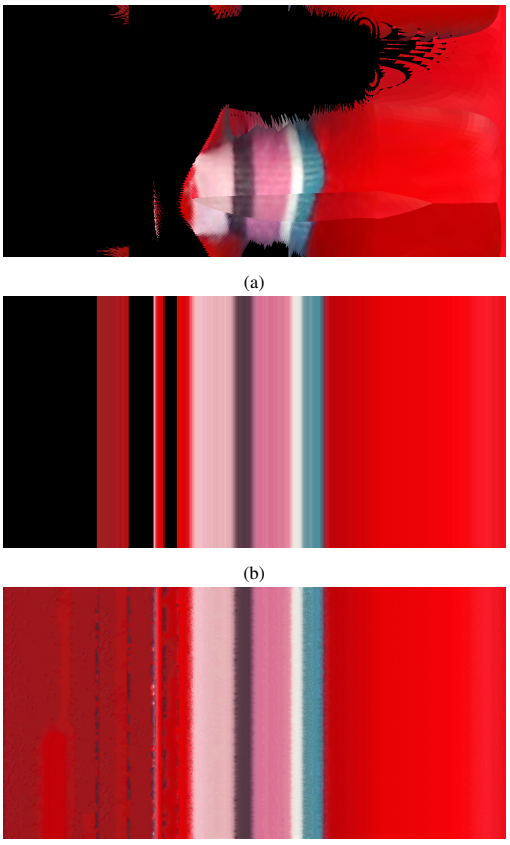
4.3. Completing textures with inpainting

In our settings, the reconstruction does require only a limited number of images w.r.t. the classic MVS pipelines. Since the object is modeled by a set of canal surfaces, two to five images are usually sufficient to completely reconstruct the model. Moreover, the images do not need to cover the entire object, thus allowing to reconstruct parts of the object that are not visible. This is particularly useful when capturing *e.g.* objects lying on some support. However, this leads to an obvious limitation in the texturing: as some parts of the object may not be visible from any viewpoints, no information can be retrieved for the texture. Thus, some pixels of the reference image I_b may not be colored. For texturing the whole object we then fill the missing regions of I_b through two completion methods depending on the nature of texture.

We consider two kinds of textures: **circular textures**, where the color of a pixel $p(t, \theta)$ is independent of θ, and **regular textures**. As an example, on Fig. 7, the legs of the doll have, in the original object, a rotationally symmetric pattern, thus are of circular type. At the opposite, both the back of the head or the face have regular texture type. The missing regions in both cases are the back side which has not been captured by the images as the object was lying on a plane. In both cases we apply the PatchMatch inpainting algorithm [22], but with a different initialization step of the algorithm.

In the case of the legs, and more generally for rotationally symmetric textures, we rely on the symmetry of the texture for the completion. The advantage of the symmetry is that it solves the offset issues along the skeleton direction. Fig. 9 shows the different steps of our procedure for one of the legs. Starting from the original reference image I_b (Fig. 9a), we create a new image I_s, called the *structure* image (Fig. 9b): for each column of I_b, we select the pixel value with the maximum confidence along the column, and we assign such value to the entire column. Note that the *structure* image may still have unfilled regions if an entire column of I_b had no data. We then use the PatchMatch correspondence algorithm in order to create a correspondence map between the original reference image I_b and I_s: for each patch of I_s we find the best matching patch in the original image. We apply the computed mapping to I_s to generate a new image which has a texture more similar to the original one, yet with the same unfilled regions. We then apply the classic PatchMatch algorithm to complete the unfilled regions (Fig. 9c). Fig. 10 shows the 3D branch with and without the texture completion.

As for the back of the head, and more generally for regular textures, we use a diffusion inpainting as initialization, as used in common implementations of PatchMatch inpainting algorithm [34].



(a)

(b)

(c)

Figure 9: Completion of mesh with texture structure: (a) the input reference image I_b; (b) the generated structure image I_s; (c) the final completed texture image (*c.f.* Fig. 10 for the final rendering).

5. Experimental results

5.1. Implementation details

Once the 3D object is reconstructed, we use the OpenCV library to create the texture images from the acquired images. The optimization problem of the exposure correction is solved with Ceres Solver library [35]. The texture completion is implemented with the image processing G'MIC framework [34]. The user selects for each branch which completion method should be used.

The reconstruction method and the texturing method are implemented in C++. Table 1 details the running time for some plushes, with a break-down for each significant step. On average, texturing takes 4 min for our CPU implementation on a Linux Mint machine equipped with an Intel i7-6700HQ CPU at 2.6 GHz with 8 GB of RAM.

5.2. Results

Plushes are good candidates for being a union of a set of canal surfaces. They are mostly filled with foam, which is an isotropic material: this creates shapes that expand in an isotropic way around the axis, leading to surfaces that are circular around the direction of the main axis. No assumptions on the number of legs, or tail, are necessary.

Figure 10: Texturing of one of the legs: (left) the original texture without inpainting and (right) the result of the texture inpainting process.

Plush	Blue	Mouse	Red	Bear	Rabbit
Perspective skeleton estimation	1.7 s	1.4 s	1.7 s	1.8 s	1.8 s
Triangulation	1.3 s	2.4 s	1.3 s	1.4 s	1.6 s
Z-buffering (CPU)	29 s	39 s	27 s	26 s	28 s
Texture extraction from images	9.9 s	15.1 s	9.6 s	8.4 s	9.8 s
Exposure correction	76 s	120 s	71 s	78 s	20 s
Registration correction	88 s	137 s	145 s	106 s	153 s
Texture completion	58 s	128 s	89 s	51 s	57 s

Table 1: Computation times for some plushes. The skeleton based reconstruction is separated in three main steps: first, a perspective skeleton is computed, then the user associates the different extremities and finally, the triangulation is done. Before extracting the texture from the different images, a z-buffering is used to characterize the front canal surface on each pixel. The z-buffering computation time is highly improvable, as it is done on CPU and not GPU. On average, it takes 73 s for the exposure correction, 120 s for the registration correction and 77 s for the texture completion.

Fig. 13 shows some results on the completion process of the texture for some branches. The first column shows for a single branch, the raw reference image when projecting all the images onto the reference image I_b without any correction. We observe brightness discontinuities in the texture due to different image exposition. The second column shows the same reference image after exposure adjustment and after the texture completion process described in Section 4.3. Depending on the branch and the type of texture we either propagate the texture around the axis of symmetry (second and forth row) or the PatchMatch inpainting. The third row shows the 3D branch with the final texturing.

Fig. 14 shows, for each row, the final result on three different plushes. The first two columns show the model rendered under two different viewpoints with the original texture, without any completion and exposure adjustment. In the second column, note that the back of the plushes is not textured as they were lying on a support during the capture, no images are available for those regions. Moreover, differences in exposure and texture misalignments are noticeable in all the models. In the last two columns shows the reconstructed textured model: most exposure discontinuities have been correctly handled by our adjustment algorithm and the textures are smoother. Globally, each model has a better global exposure and the structure of the regular textures is respected (legs, ears of the second and third models). We provide additional results: models with the original texture [36] and after applying our pipeline for improving textures [37].

Some issues remain, and are discussed in the next section.

(a) (b) (c)

Figure 11: An example of object (a) which does not satisfy our assumptions: the head is a surface that cannot be properly modeled as a canal surface, thus affecting the texture mapping (); the method described in Fig. 11b helps to obtain a better texturing of the object (c).

5.3. Limitations

The completion method is chosen manually by the user according to the type of texture (regular, rotationally-symmetric, stochastic *etc*.). In order to automate the task it would be interesting to analyze the original branch image I_b in order to classify the texture nature into circular or regular. Another possibility is to first apply the two inpainting methods and then choose the method that gives better results.

Our method depends on the accurate estimation of the camera positions to have a reliable reconstruction. Incorrect or poor calibration may affect the results, especially when projecting the texture of the branch on the reference image I_b. Moreover, the reconstruction method assumes that the object is composed of a set of canal surfaces and it is based on a least square triangulation, which is sensitive to noise. The reconstruction of objects that do not satisfy this constraint may be inaccurate. Fig. 11 shows an example of a plush that does not satisfy the tubular geometry, especially for the head which is not a typical canal surface (and would have a surface medial axis). Even if the reconstruction gives a complete, satisfying model, the offset to the real shape affects the mapping of the texturing (Fig. 11b). Nevertheless, thanks to the misalignment correction described in Section 4.2 we can obtain fair results for texturing (Fig. 11c).

Concerning the exposure correction, we only work on lighting: the blue plush on Fig. 14 (second row) has different variation of blue depending on the image. Our correction is not able to unify or smooth different coloring.

We report in Fig. 12 the reconstruction results obtained with the open-source MVS pipeline AliceVision [38] with our input images. A direct comparison with our results is not fair as the assumptions are different: our work is limited to tubular objects for which we reconstruct a full, textured, parametric model that approximates the real surface, while MVS approaches reconstruct more general objects by triangulation, thus obtaining, in general, models with finer geometric details. The advantage of our method is that from very few images is able to reconstruct a textured model that has a good quality and can be a good basis, *e.g*., for a graphic designer to work on. Since the reconstruction process does not rely on finding image correspondences, our method is able to deal with smooth or poorly textured surfaces.

6. Conclusion

In this work, we have proposed a texturing and inpainting method for models reconstructed as a set of canal surfaces. The parametric nature of the model leads to good results concerning the texturing, and we are able to correct the exposure and misalignment, and inpaint the missing texture information. From just a few images of a tubular 3D shape, a 3D model is reconstructed and can be textured by the proposed method. The future directions of this work are driven by the limitations: classifying the texture into circular or regular, and generalizing the correction of the exposure to smoothing of color. Moreover, most plushes are matte. Additional work would be needed to handle shiny surfaces.

Acknowledgements

This research was supported by the CIFRE ANRT 2016/0139 and the regional project FEDER-FSE Midi-Pyrénées et Garonne REALISM n.15056690.

References

[1] Y. Furukawa and C. Hernández, "Multi-View Stereo: A Tutorial," *Foundations and Trends® in Computer Graphics and Vision*, vol. 9, no. 1-2, pp. 1–148, 2015.

[2] N. Snavely, S. M. Seitz, and R. Szeliski, "Photo tourism: exploring photo collections in 3D," *ACM Transactions on Graphics*, vol. 25, pp. 835–846, jul 2006.

[3] M. Waechter, N. Moehrle, and M. Goesele, "Let there be color! Large-scale texturing of 3D reconstructions," in *Proceedings of the 2014 European Conference on Computer Vision (ECCV 2014)*, vol. 8693 LNCS, pp. 836–850, 2014.

[4] D. Hilbert and S. Cohn-Vossen, *Geometry and the Imagination.* New York: Chelsea: Chelsea Publishing Co., 1932.

[5] B. Durix, G. Morin, S. Chambon, C. Roudet, and L. Garnier, "Towards Skeleton Based Reconstruction: From Projective Skeletonization to Canal Surface Estimation," in *Proceedings of the IEEE International Conference on 3D Vision*, pp. 545–553, 2015.

[6] B. Durix, G. Morin, S. Chambon, C. Roudet, and L. Garnier, "Skeleton-based Multiview Reconstruction," in *Proceedings of the IEEE International Conference on Image Processing*, pp. 4047–4051, 2016.

[7] D. Luebke, B. Watson, J. D. Cohen, M. Reddy, and A. Varshney, *Level of Detail for 3D Graphics*. New York, NY, USA: Elsevier Science Inc., 2002.

[8] F. Bernardini, I. Martin, and H. Rushmeier, "High-quality texture reconstruction from multiple scans," *IEEE Transactions on Visualization and Computer Graphics*, vol. 7, no. 4, pp. 318–332, 2001.

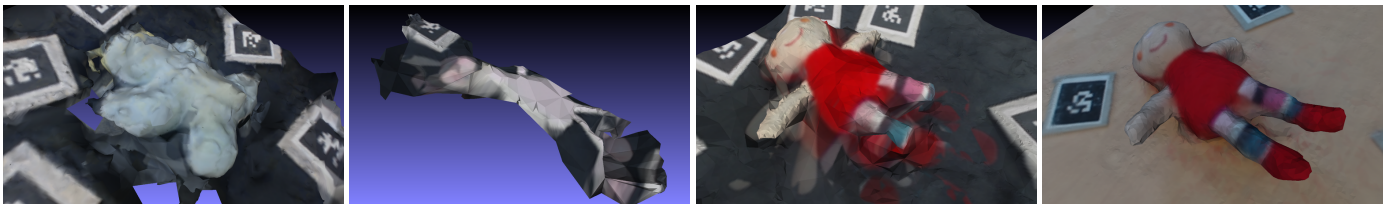
[9] M. Corsini, M. Dellepiane, F. Ganovelli, R. Gherardi, A. Fusiello, and R. Scopigno, "Fully Automatic Registration of Image Sets on Approximate Geometry," *International Journal of Computer Vision*, vol. 102, pp. 91–111, mar 2013.

[10] I. Garcia-Dorado, I. Demir, and D. G. Aliaga, "Automatic urban modeling using volumetric reconstruction with surface graph cuts," *Computers & Graphics*, vol. 37, pp. 896–910, nov 2013.

[11] S. N. Sinha, D. Steedly, R. Szeliski, M. Agrawala, and M. Pollefeys, "Interactive 3d architectural modeling from unordered photo collections," *ACM Trans. Graph.*, vol. 27, pp. 159:1–159:10, Dec. 2008.

[12] M. Callieri, P. Cignoni, M. Corsini, and R. Scopigno, "Masked photo blending: Mapping dense photographic data set on high-resolution sampled 3D models," *Computers & Graphics*, vol. 32, pp. 464–473, aug 2008.

[13] S. Bi, N. K. Kalantari, and R. Ramamoorthi, "Patch-based optimization for image-based texture mapping," *ACM Transactions on Graphics*, vol. 36, pp. 1–11, jul 2017.



(a) (b) (c) (d)

Figure 12: Some results obtained with the open-source state-of-the-art MVS pipeline AliceVision [38] using our images as input. Poorly textured objects (a) or objects with thin parts (b) are difficult to reconstruct from few images with classic MVS pipelines. Increasing the number of images and a better coverage of the whole space around the object improve the quality of the final reconstruction: (c) and (d) have been reconstructed from 4 and 30 images, respectively. More 3D examples of MVS reconstruction for our dataset of objects are available here [39].

[14] C. Hernández Esteban and F. Schmitt, "Silhouette and stereo fusion for 3D object modeling," *Computer Vision and Image Understanding*, vol. 96, pp. 367–392, dec 2004.

[15] T. Chen, Z. Zhu, A. Shamir, S.-M. Hu, and D. Cohen-Or, "3sweep: Extracting editable objects from a single photo," *ACM Trans. Graph.*, vol. 32, pp. 195:1–195:10, Nov. 2013.

[16] C. Xiao, M. Liu, N. Yongwei, and Z. Dong, "Fast Exact Nearest Patch Matching for Patch-Based Image Editing and Processing," *IEEE Transactions on Visualization and Computer Graphics*, vol. 17, pp. 1122–1134, aug 2011.

[17] M. Bertalmío, V. Caselles, S. Masnou, and G. Sapiro, "Inpainting," in *Computer Vision*, pp. 401–416, Boston, MA: Springer US, 2014.

[18] T. F. Chan and J. Shen, "Nontexture inpainting by curvature-driven diffusions," *Journal of Visual Communication and Image Representation*, vol. 12, no. 4, pp. 436 – 449, 2001.

[19] A. Criminisi, P. Perez, and K. Toyama, "Object removal by exemplar-based inpainting," in *Proceedings of the 2003 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2003. Proceedings.*, vol. 2, pp. II–721–II–728 vol.2, June 2003.

[20] P. Buyssens, M. Daisy, D. Tschumperle, and O. Lezoray, "Exemplar-based inpainting: Technical review and new heuristics for better geometric reconstructions," *IEEE Transactions on Image Processing*, vol. 24, pp. 1809–1824, June 2015.

[21] A. Efros and T. Leung, "Texture synthesis by non-parametric sampling," in *Proceedings of the Seventh IEEE International Conference on Computer Vision*, pp. 1033–1038 vol.2, 1999.

[22] C. Barnes, E. Shechtman, A. Finkelstein, and D. B. Goldman, "PatchMatch: A randomized correspondence algorithm for structural image editing," *ACM Transactions on Graphics (Proc. SIGGRAPH)*, vol. 28, Aug. 2009.

[23] G. Chican and M. Tamaazousti, "Constrained patchmatch for image completion," in *Proceedings of the 10th International Symposium on Advances in Visual Computing ISVC*, Springer International Publishing, Dec. 2014.

[24] K. He and J. Sun, "Statistics of patch offsets for image completion," in *Proceedings of the 12th European Conference on Computer Vision - Volume Part II*, ECCV'12, (Berlin, Heidelberg), pp. 16–29, Springer-Verlag, 2012.

[25] M. Köppel, M. B. Makhlouf, K. Müller, and T. Wiegand, "Fast image completion method using patch offset statistics," in *Proceedings of the 2015 IEEE International Conference on Image Processing (ICIP)*, pp. 1795–1799, Sept 2015.

[26] Y. Liu and V. Caselles, "Exemplar-Based Image Inpainting Using Multiscale Graph Cuts," *IEEE Transactions on Image Processing*, vol. 22, pp. 1699–1711, may 2013.

[27] V. Caglioti and A. Giusti, "Reconstruction of Canal Surfaces from Single Images Under Exact Perspective," in *Proceedings of the 2006 European Conference on Computer Vision (ECCV 2006)*, pp. 289–300, 2006.

[28] C. Rother, V. Kolmogorov, and A. Blake, "GrabCut: Interactive foreground extraction using iterated graph cuts," *ACM Trans. Graph.*, vol. 23, pp. 309–314, Aug. 2004.

[29] M. Peternell and H. Potmann, "Computing rational parametrizations of canal surfaces," *Journal of Symbolic Computation*, vol. 23, pp. 255–266, feb 1997.

[30] J. Fayer, G. Morin, S. Gasparini, M. Daisy, and B. Coudrin, "Radiometric confidence criterion for patch-based inpainting," in *Proceedings of the International Conference on Pattern Recognition (ICPR 2018)*, 2018. to appear.

[31] R. I. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*. Cambridge University Press, ISBN: 0521540518, second ed., 2004.

[32] E. Zhang, M. F. Cohen, and B. Curless, "Emptying, refurnishing, and relighting indoor spaces," *ACM Trans. Graph.*, vol. 35, pp. 174:1–174:14, Nov. 2016.

[33] M. Arikan, R. Preiner, C. Scheiblauer, S. Jeschke, and M. Wimmer, "Large-scale point-cloud visualization through localized textured surface reconstruction," *IEEE Transactions on Visualization and Computer Graphics*, vol. 20, pp. 1280–1292, Sept 2014.

[34] D. Tschumperle, "G'MIC - GREYC's Magic for Image Computing: A Full-Featured Open-Source Framework for Image Processing," 2016.

[35] S. Agarwal, K. Mierle, and Others, "Ceres solver." `http://ceres-solver.org`, 2016.

[36] `https://skfb.ly/6yCCG`, 2018.

[37] `https://skfb.ly/6yCCE`, 2018.

[38] AliceVision, "Photogrammetric Computer Vision Framework," 2017.

[39] `https://skfb.ly/6yxQO`, 2018.

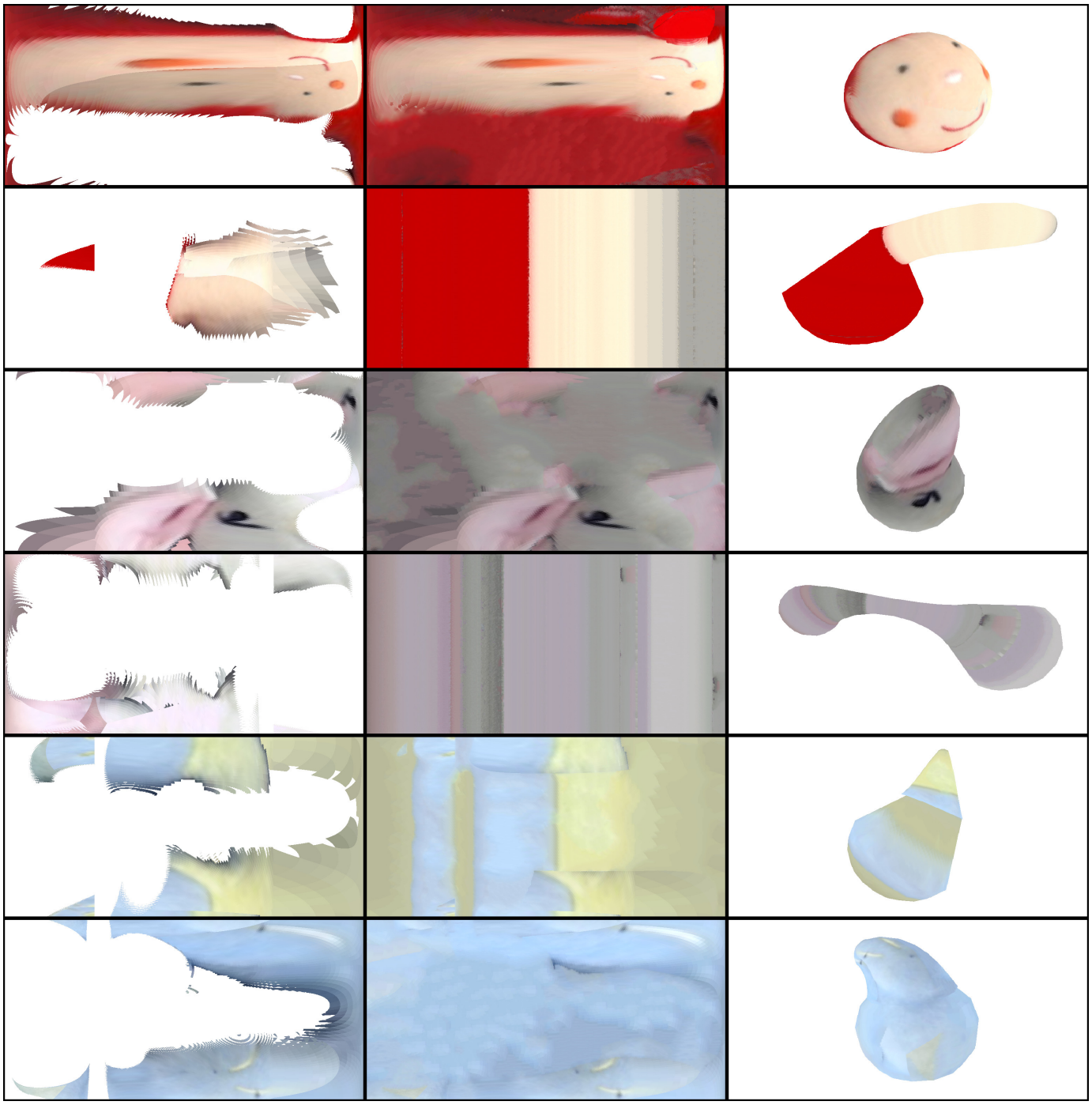


Figure 13: Texture enhancement of several branches: the first column shows the original texture as projected on the reference image I_b while the second column shows the completed texture. The third column shows the completed texture applied to the relevant 3D branch.

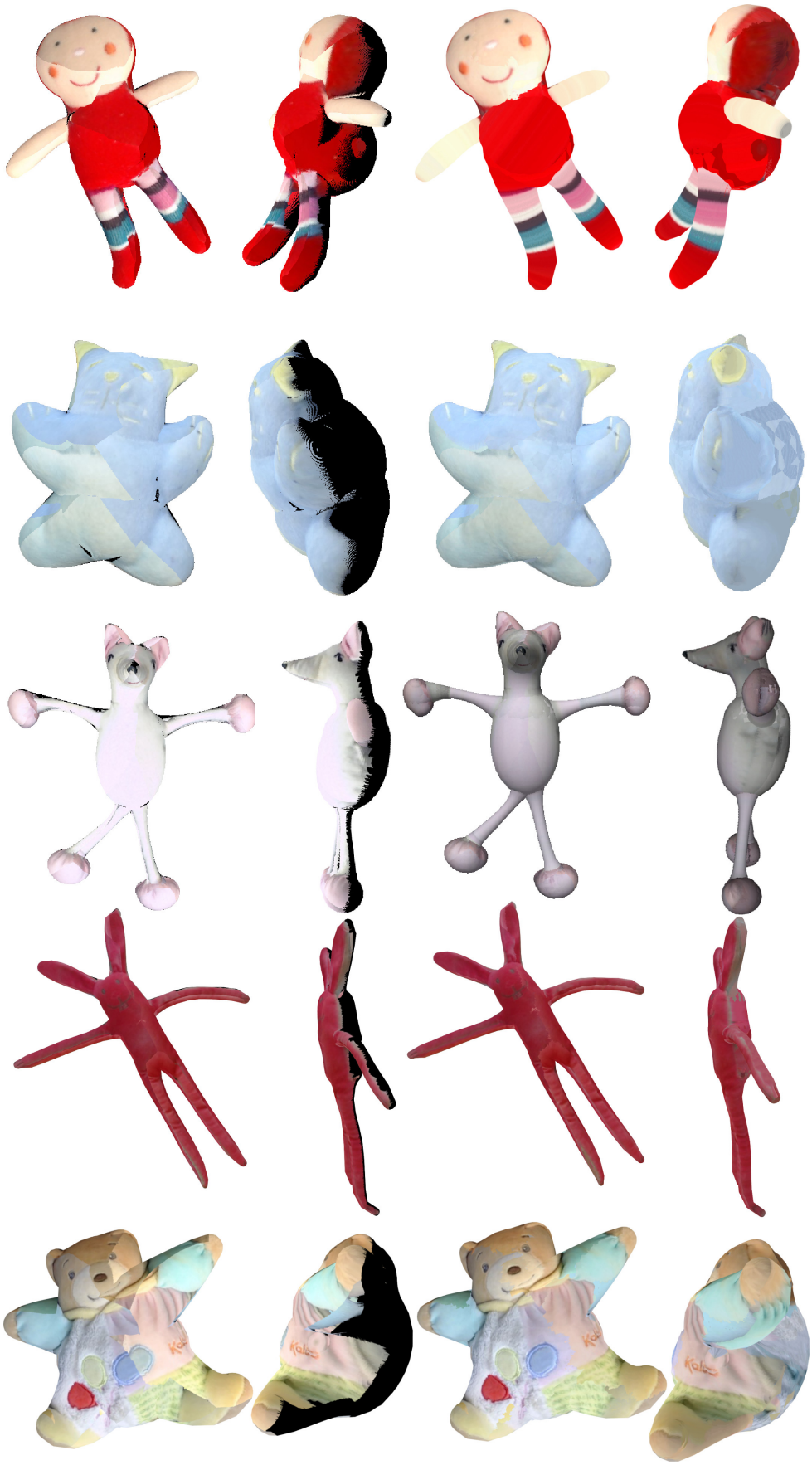


Figure 14: Texture completion results on five plushes: the first two columns show the row model with the direct texture mapping, the last two columns show the model with the texture improved (w.r.t. exposure and alignment) and completed through inpainting. The reconstructions of the first and second rows used 4 input images, and the one of the third row (the mouse) used 3 input images.