Vis Comput (2012) 28:983–993
DOI 10.1007/s00371-012-0743-7

ORIGINAL ARTICLE

Texturing 3D models from sequential photos

Ricardo Marroquim · Gustavo Pfeiffer · Felipe Carvalho · Antonio A.F. Oliveira

Published online: 28 June 2012

Abstract This paper proposes a methodology to texture 3D objects with low geometric features from sequentially taken photos. These models pose a challenge to current approaches since they are mainly driven by geometric features—such as contours—that can be extracted from the photographs and uniquely matched with the 3D model. However, when dealing with certain types of objects, such as vases or mechanical equipments, for example, it is not uncommon to find cases where the geometric information is insufficient.

Our method compensates for the lack of geometric features by using a variation of a contour-based approach that is guided not only by external contours, but also by the internal ones extracted directly from the photos. To align the features a custom-made optimization method is described that avoids common convergence pitfalls encountered in this scenario. In addition, pursuing a fully automatic solution, a linear approach based on feature matching is employed to generate a first guess for the nonlinear optimization. The overall goal is to facilitate an on-site registration process where the photos are taken in a sequential manner and aligned as they are acquired.

Keywords Texture-to-geometry registration · 3D virtual replica · High-resolution texture mapping · Least-squares minimization

Fig. 1 An example of a model textured with our method: rendering of the final model (*top*), the first three photos that were sequentially registered (*middle*), and three other randomly selected photos that were registered (*bottom*)

1 Introduction

Recently, the generation of 3D virtual replicas of real physical objects has received a great attention, and is becoming an incorporated technology in many fields. With recent 3D scanners, it is possible to build high resolution models with great fidelity for a broad variety of objects. Nevertheless, in most cases, this process is still far from being fully automatic, and may require a laborious manual effort depending on the subject.

This is specially true when high resolution reflectance properties, usually color, are required. In comparison to the acquisition of geometry, precise color measurements is usually harder to acquire since it is more susceptible to ambient conditions. Moreover, precise control of the illumination demands complex setups, and in many cases—such as on-site scanning campaigns—it is not possible at all. Even worst, ideal light conditions for acquiring geometry are usually different than those for acquiring color.

R. Marroquim (✉) · G. Pfeiffer · F. Carvalho · A.A.F. Oliveira
Computer Graphics Lab (LCG), COPPE-UFRJ, Rio de Janeiro, Brazil
e-mail: marroquim@cos.ufrj.br

Since most digitalization equipments lack high quality color registration embedded systems, the mapping of reflectance properties is usually carried out separately through one of two methods: acquiring BRDFs or projecting photographs by calibrating the camera's parameters. Even though the former is known to achieve high-quality results, the process still requires complex lab setups and expensive apparatus. On the other hand, notwithstanding the relative low cost of photographic equipments, most camera calibration approaches demand a considerable amount of user interaction, and there is no single procedure that can deal robustly with a great variety of scenarios. This paper is inserted in this latter discourse.

We propose a method to texture models using a sequential approach, where the photos are registered as they are acquired, i.e., in a "shoot and register" fashion (see Fig. 1). This is an important resource during on-site digitalization campaigns for many reasons: it provides an immediate view of the photos' coverage, as well as their quality, and avoids future visits to the site to correct or complete missing regions; in some cases, specially in the *cultural heritage* context, a second chance to register the object's appearance at a given state might not be possible when it is undergoing a restoration intervention, for example; finally, sometimes registering a large number of photos at a later time might present difficulties in identifying the corresponding regions on the geometric model, specially when the photos are taken from a close-up view for high resolution texturing, or the object has many similar parts.

Another contribution of this paper is the description of an approach to texture objects lacking significant geometric features. Since most methods rely on geometric features of the 3D model to find correspondences on the photos, they are not suitable for a class of objects that lack these characteristics, such as vases or mechanical pieces. We address this problem with a new formulation of a nonlinear optimization approach based on a smoothing function. In addition, we describe possible approaches to generate a starting position for this iterative method.

A quick overview of the system is given in Sect. 3, followed by detailed descriptions and formulations of the registration method (Sects. 4 and 5). Implementation details are given in Sect. 6. Finally, we depict some results of models textured with our approach in Sect. 7, and discuss some limitations and conclusions of our work in the last sections.

2 Related work

The use of contours to drive the image-to-geometry registration is not new. Neugebauer and Klein [18] start with point correspondences created by user interaction to set the initial estimation of the parameters, followed by a Levenberg–Marquardt minimization approach. Their objective function is a combination of these point correspondences, a contour-based criterion, and an attribute-based one where information from the object's surface is compared to those of the images.

Matsushita and Kaneko [17] compute the minimum set of photographs that covers all the geometry. In fact, their methodology takes the inverse direction of most related works, where synthetic views are first generated using only the geometry, and then photos are taken from viewpoints matching as close as possible the camera locations of the synthetic views. Since it is almost impossible to exactly match them manually, an optimization algorithm minimizes the contour errors during a final stage.

Another variation was proposed by Lensch et al. [13]. Instead of comparing distance from curves, they compare the filled areas inside the contours by superposing them, which they refer to as the silhouettes. A fast hardware XOR method is computed over the overlapping projections to serve as the error criterion, which in the case of a perfect match would be zero. An optimization method is then run and restarted several times to avoid local minima. The initial guess is given by retrieving some information from the camera and by sampling possible angular directions.

Liu et al. [14] proposed a method that combines a dense point cloud from 3D range scan data, with a sparse point cloud build from a 3D reconstruction of 2D photos. A subset of the 2D photos are aligned with the dense set, followed by the registration of the complete set using multiview geometry.

Corsini et al. [5] employ mutual information to develop an error metric based on surface properties such as normals, ambient occlusion and reflection directions. Even though they achieve fast and high-quality results, their method is only suitable for models containing high geometric details to extract relevant illumination-based information.

In fact, this last remark is what mainly differentiates our methods from the previous approaches, since all of them require significant geometric features to drive the optimization.

After calibrating the camera parameters for all photos, a method to blend the textures is required. Among these, some treat the issue as a global optimization problem [10], while others rely on a local texture blending [3]. Finally, since it is not always possible to achieve a perfect result with only the calibrations methods—in view of issues such as lens distortion or imprecise geometry—a final warping step might be necessary to correct small misalignments or discontinuities due to illumination variations [1, 4, 7, 10].

3 Method overview

The input for the system is a set of photos and the 3D geometry (see Fig. 2). For each target image, we start by setting

an initial guess for the camera parameters using a manual or automatic method based on feature correspondences.

From this point, a nonlinear optimization method is employed to retrieve the calibrated camera that matches as close as possible the projected 3D model with the target image. To this end, we require a measure to compare how good a certain camera configuration is; here we use the distance of each pixel of the projected model's contours to the target photo's contours via a distance transform. However, instead of using only external silhouettes, we also make use of the already textured photos, i.e. the projected rastered source image contains the pixels of the model's outer contours as well as the inner contours from the aligned photos. With these distances an error is evaluated in order to guide our nonlinear optimization method. After convergence, the calibrated camera is used to register the target photo onto the model, and the process starts again for the next photo that overlaps an already textured region. Figure 3 illustrates this process, and the details of each step are described in the following sections.

Fig. 2 The six photos used to produce the final coffee mug 3D model (*top two rows*). Only the geometric model (*bottom left*) and the final colored model (*bottom right*)

4 Initial guess

The intrinsic parameters are obtained by the camera's specification (e.g., CCD size) and the focal length (a lens with fixed focal length was used in our experiments). Further precise calibration could be obtained, but for this work we did not consider any radial or tangent distortions, and assumed the center of the image as the principal point.

To compute the extrinsic parameters, three different approaches may be used: manually placing the initial configuration, manually selecting 2D–3D correspondences, or an automatic method to retrieve the correspondences and remove outliers.

In the first case, the external orientation is defined within a 3D visualization interface, where the 3D model should be roughly oriented to match the photo being aligned, from which the camera's view vector can be defined by clicking on the 3D surface. The camera is then translated along this view axis by an estimate of the focus distance extracted from the photo's EXIF file (note that this value is very imprecise, but suitable for our needs). Adding the up vector—also acquired from the current view matrix—there is enough information to build the camera's initial extrinsic matrix. Even though this manual method in most cases does not require a long user interaction, for some models it becomes a very tedious task, specially when there are no well-defined contours to be extracted from the textures. Nevertheless, this

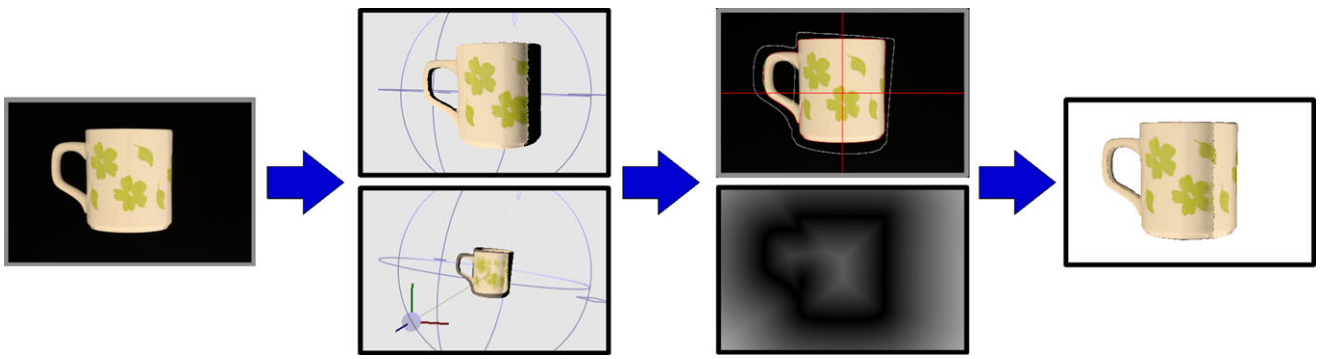

Fig. 3 System overview: (*left*) the target photo to be registered; (*middle left*) the partially textured model is positioned to roughly align the photo with the geometry; (*middle right top*) the optimization system in the process of matching the projected model's contour with those of the photo, using a smoothed distance transform (*middle right bottom*); (*right*) a rendering of the model—with the new registered texture included—using the calibrated camera matrix matching the target photo (*left*); note that the seams are intentionally left for visual feedback during the alignment, they are removed in a final texture blending stage

approach is specially useful for aligning the first photo, where the model does not contain any color information yet.

Feature based methods, such as SIFT [15] or Harris Feature Detector [12] together with NCC (Normalized Cross Correspondence), are able to automatically generate 2D–2D correspondences. We start by extracting feature points using one of these classical methods from the literature, retrieving a set of correspondences between an already aligned photo (source), and a to-be-aligned photo (target). In any case, the source's feature points are back-projected onto the model's surface to recover their 3D positions using a simple ray-casting procedure. The problem that is left is to retrieve the exterior parameters of the camera so that the projection of these 3D points matches their 2D correspondences on the target image.

We proceed with RANSAC to remove outliers, where at each iteration six correspondences pairs of 2D–3D features are randomly chosen, and the external orientation problem solved using a linear algorithm as described by Fiore [9]. Briefly, the method first solves for the depths of the 2D points by eliminating the scale, rotation and translation components, reducing the problem to finding the absolute orientation of the camera. This second step is accomplished by singular values decompositions and solving the Procrustes Orthogonal Problem, i.e., finding the closest rotation matrix to a given matrix. From this point, the translation and scale components can be trivially computed.

For each iteration, we count the number of inliers for the whole correspondence set, and keep the solution with highest consensus. In addition, to improve our matches, we compute the mean distance between the pixel correspondences and remove those outside the standard deviation. Even though there is no guarantee that this will remove any remaining outlier, we have noticed that this simple procedure greatly improves the initial alignment in some cases. When there is a poor distribution of the correspondences over the image the linear method sometimes fails, i.e., when the SIFT or Harris features are concentrated in some parts of the image because it contains large uniform regions (without automatically extractable features). We have thus included the option to insert correspondences manually to arrive at a better solution. For more controlled environments and simpler cases, the Harris method works well enough and produces a better distribution of the corners, however, SIFT is usually more robust for most situations.

From this point, we proceed with the nonlinear iterative method described in the following section. Figure 4 illustrates the result from both steps with a detail of the bottle model.

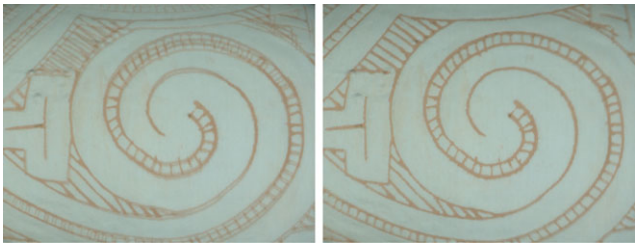

Fig. 4 The initial alignment using the linear method (*left*) and the final result after the full nonlinear optimization (*right*)

5 Optimization

In this section, the objective function used for a least-squares minimization is derived, and the introduced modifications are explained. Briefly, it is a variation of a Levenberg–Marquardt implementation [16] that improves convergence in the proposed scenario. For every iteration, the model is rendered using only its own outer contours combined with the inner ones of the already registered textures: the set of points that generated the pixels of this projection serve as our source points (p_i). These points are matched against the target points (q_i), which in this case are the outer and inner contour pixels of the photo being aligned. In short, we search for the camera parameters that best fits the projection of the partially textured model (*source*) with the current photo (*target*).

5.1 Objective function and its derivative

A standard way of evaluating how two contours (C_1 and C_2) are similar is integrating along C_1 the squared distance between each one of its points (y) and the point on C_2 ($q(y)$) that best corresponds to y according to criteria that can involve distance, similarity between geometrical features at previously marked points and even the textures around y and $q(y)$. To make the correspondence almost everywhere differentiable and $y - q(y)$ suitable to be efficiently computed, we will simply take $q(y)$ as the point on C_2 that is closest to y. In the context where C_1 is the projection of a set of model edges relative to a camera configuration K and C_2 consists of distinguishable contours on a photograph, the similarity measure which we would like to minimize can be expressed as

$$f(K) = \frac{1}{N} \sum_{i=1}^{N} D\big(C_2, y(p_i, K)\big)^2 \tag{1}$$

where $D(C_2, .)$ is the distance to C_2 and $p_i, i = 1, \ldots, N$ discretize the contours projected onto C_1. The function is divided by N since the number of points varies with the view direction and can change during an optimization process.

To minimize $f(K)$ through a derivative based process like Levenberg–Marquardt, we need to derive f in relation

to the parameters chosen to represent K, which is done by applying the chain rule to a sequence of functions whose composition is the ith component of the sum defining f. In the following, we will focus on each one of these functions.

Function y is the composition of the projection matrix operator (π) with the extrinsic matrix one, formed from a rotation matrix $R(K)$ and a translation vector $t(K)$.

$$y(p_i, K) = \pi\big(x(p_i, K)\big) = \pi\big(R(K)p_i + t(K)\big) \tag{2}$$

where $x(p_i, K)$ is the point corresponding to p_i in camera coordinates. π can be expressed by $\pi(x) = Y\frac{x}{x_3}$, where x_3 is the z-coordinate of x and Y is the intrinsic matrix. From that expression, we derive that $\partial_x \pi(x) = Y[\frac{I}{x_3} - \frac{xe_3^T}{(x_3)^2}]$.

While K is not concretely represented in our model, we would rather derive our objective function with respect to incremental rotations and translations $\tilde{R}$ and $\tilde{t}$ as in:

$$\begin{bmatrix} R(K) & t(K) \\ 0^T & 1 \end{bmatrix} = \begin{bmatrix} \tilde{R} & \tilde{t} \\ 0^T & 1 \end{bmatrix} \begin{bmatrix} R(K_{\text{current}}) & t(K_{\text{current}}) \\ 0^T & 1 \end{bmatrix} \tag{3}$$

We write then $x(p_i, K) = x^+(r, t, x(p_i, K_{\text{current}}))$, where r and t parameterize $\tilde{R}$ and $\tilde{t}$, using:

$$\begin{aligned} x^+(r, t, x) = {} & \frac{rr^T}{r^T r}x + \cos\big(\|r\|\big)\left(I - \frac{rr^T}{r^T r}\right)x \\ & + \sin\big(\|r\|\big)\frac{r}{\|r\|} \times x + t \end{aligned} \tag{4}$$

where r represents a rotation around vector $\hat{r}$ by $\|r\|$ radians, and t is a translation. Replacing sine and cosine by their Taylor series, we obtain a Taylor series for x^+ that yields $\partial_{(r,t)}x^+ = [-(x\times)\ I]$ at $r = t = 0$, with $(x\times)$ representing the matrix for the cross product with x on the left.

As it is well known, $D(C_2, .)$ is a differentiable function outside the medial axis of C_2, whose gradient at y_i is the unitary of $y_i - q(y_i)$. The objective function (1) can now be derived with respect to the vectors r and t:

$$\begin{aligned} \partial_{(r,t)} f &= \frac{2}{N}\sum_{i=1}^{N} D(C_2, y_i)J_i \\ J_i &= \big(\nabla_y D(C_2, y_i)\big)(\partial_x \pi)\big(\partial_{(r,t)}x^+\big) \\ &= \nabla_y D(C_2, y_i)Y\left(\frac{I}{x_{i3}} - \frac{x_i e_3^T}{(x_{i,3})^2}\right)\big[-(x_i\times) \quad I\big] \end{aligned} \tag{5}$$

where x_i is the point p_i in camera coordinates.

Thus, in the Levenberg–Marquardt context, we want to solve a system of the form $(A + \lambda\,\text{diag}(A))x = b$, where:

$$A = \sum_{i=1}^{N} J_i^T J_i \quad \text{and} \quad b = \sum_{i=1}^{N} -J_i^T D(C_2, y_i) \tag{6}$$

Usually, the Levenberg–Marquardt algorithm updates the damping factor λ according to a parameter ρ, which is defined as the ratio between the decrease of the objective function ($f(K_j) - f(K_{j+1})$) and its expected decrease ($L(0) - L(x)$), where $L(x)$ is a local quadratic approximation of f based on the derivatives of D at K_j. Thus, the expression for ρ we use is

$$\rho_{j+1} = \frac{f(K_j) - f(K_{j+1})}{\frac{1}{N}\langle x, \lambda_j\,\text{diag}(A)x + b\rangle} \tag{7}$$

The output of one iteration is a candidate increment of the rotation matrix and translation vector. These are applied to the current extrinsic matrix, and the projection process is repeated to validate the new solution. Then ρ is computed by (7), if $\rho > 0$ we accept the candidate parameters and decrease λ; otherwise, we increase λ and do not update the matrices for the next iteration. The expressions for updating λ are those proposed by Madsen et al. [16].

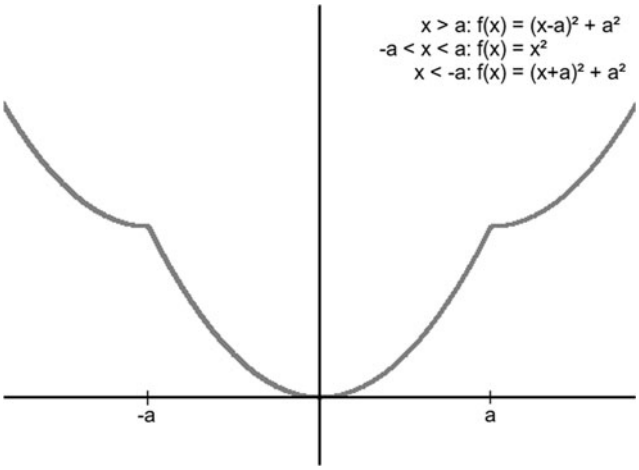

Fig. 5 Objective function matching a pair of points located at $x - a$ and $x + a$ with the pair $\{a, -a\}$. Observe that, when using the Levenberg–Marquardt algorithm, x converges to a if the initial guess $x_0 \geq a$

5.2 Smoothed function approach

Making a point correspond to the its closest point on a target's contour can impair the matching process in some aspects. The problems derive from clear correspondence flaws like not being 1–1, not considering the similarity between local features and having discontinuities along the target medial axis. Discontinuities, in particular, can generate singular points which, in spite of often not even being local minima for f, become optimal if the parameters are constrained to an adjacent region capable of containing the whole sequence produced by the Levenberg–Marquardt. Figure 5 presents an 1D example of such a situation.

A simple way of reducing this effect is to smooth the distance function. For every photo, we first compute a distance transform using the method described by Felzenszwalb and Huttenlocher [8], and then apply a Gaussian filter and numerically compute its gradient to use on the derivative.

Fig. 6 Comparison of the standard closest-point distance with the smooth function: (*left*) a sphere model with a horizontal white stripe, where a photo was registered at the correct position, and then another copy was placed in a different configuration (the diagonal stripe) to test the convergence in an ideal situation; (*middle*) the standard method converges to the wrong configuration; (*right*) while the proposed smooth function is able to achieve the exact match

Figure 6 illustrates the typical situation where the smooth function is able to arrive at the correct configuration, while the standard method fails. Note that the white stripe is turned into practically two parallel lines after extracting the inner contours.

6 Implementation

In order to have an efficient and practical system that can be used in a real digitalization campaign, each photo alignment should not take more than a few minutes. To this end, a series of GPU techniques are used to improve the performance and provide a fast visual feedback.

For every Levenberg–Marquardt iteration, a variation of a two-pass rendering technique [6] produces a bitmap representation of the model's outer contour. Since the corresponding 3D points in camera position are required for the optimization phase (Sect. 5), in the second rendering pass a simple shader program is used to gather this information in a framebuffer. This result is combined with another pass projecting the inner contours of the already registered photos. The final result is read back to the CPU and the buffer sequentially scanned to retrieve the contour pixels and create a list of source points (p_i) and their corresponding 3D positions.

Following, the list of contour pixels are passed to the optimization method, the smoothed versions of (6) (as described in Sect. 5.2) are computed, and the system is solved by matrix decomposition using the Eigen library [11].

To speed up the convergence, a multiresolution strategy is employed by downsampling the images: For the first resolution levels, a rough alignment is quickly obtained, which is incrementally refined during the higher resolution levels. The algorithm starts with low-resolution photos (e.g., 25 % of the original resolution), and doubles it once the optimization has converged for the current level. We conservatively consider 6 consecutive iterations without a decrease of at least 1 % of the mean error as stable. This value is based on our experiments.

Finally, to allow for an interactive manipulation and visualization of the texturing progress, a multirender pass is implemented for visualizing a large number of textures simultaneously. Note that a final unified texture, or per-vertex-color, is only generated in a final process after registering all photos, hence, at any given time during the acquisition and registration process a subset of n registered photos must be available as texture units. In fact, there are two corresponding subsets, one with the original textures for visualization, and another containing the extracted edges for optimization purposes. Thus, we create n texture coordinates buffers, and for each pass render to a framebuffer the maximum number of texture slots available for the graphics card. The result of one render pass is accumulated with the subsequent ones until all textures are rendered and the final result is ready.

7 Results

The results are illustrated with a few datasets, where we intentionally chose objects that were either handcrafted or made of natural material to avoid very clear and defined contours, which would make the correspondence matching easier. All geometries were acquired with a NextEngine scanner; the provided color information was discarded since we wish to achieve a solution for any type of acquisition device. All photos were taken with a Nikon D80 in full resolution mode, and later downsampled to 2200×1474. The chosen setup was to keep the camera fixed and rotate the objects over a black background. The camera was connected to the computer in order to shoot and download the photos directly through the system's interface. Three different levels were used for the multiresolution scheme: 400×268, 800×536, and the original 2200×1474. In average each photo was registered in 2–3 minutes.

Once all photos are registered, the final colored model is generated using the approach by Callieri et al. [3]. All models were produced with a per-vertex color, but texture maps could be similarly created. Figure 7 shows renderings of the final results.

Table 1 respectively gives for each column: the number of triangles for each input model; the number of triangles after resampling for producing a smooth per-vertex-color result; the number of registered photos; the average mean and maximum error (in pixels) of all photos; and the average number of iterations per resolution level. The error is associated to the distance $D(C_2, y(p_i, K))$ (1), i.e., mean and maximum distance from every pixel of the projected contour to the target one. Note that the errors account also for discretization issues during rasterization, and that it does not correspond to the final global registration error; it is the error considering the already registered photos when each target photo was aligned.

Fig. 7 Ray-traced colored models: bottle (*left*), vase (*middle*), coffee mug (*left top*), and the same model registered in inverse ordering (*left bottom*)

Table 1 Datasets description

Model	Tris (orig)	Tris (final)	Photos	Mean error	Max error	400 × 268	800 × 536	2200 × 1474
Coffee mug	240K	1.9M	6	3.04	8.48	9	11	16
Coffee mug (inv.)	240K	1.9M	6	3.34	10.2	9	10	15
Bottle	115K	2.7M	6	1.12	3.14	10	12	14
Bottle (automatic)	115K	2.7M	6	2.69	9.2	10	9	7
Vase	216K	3.4M	5	0.92	2.10	9	10	16
Toy	1.1M	813K	6	2.43	6.41	7	9	13
Pestle (automatic)	48K	780K	5	1.56	3.2	15	7	7
Meridian circle	3.1M	148K	41	4.1	14.8	21	18	12

Coffee mug This is apparently the simplest case, since it offers more geometric hints (i.e., the handle) than the other two. On the other hand, while the handle can be useful in some cases, it might mislead the optimization on others. When it is in front of the mug in a manner that it does not contribute to the outer contours (Fig. 2, middle row left), it becomes a possible source of misalignment, due to the projection of false contour lines. Another issue is that the drawings of the flowers and leaves are not very sharp (possibly hand painted), which makes the contour extraction error prone.

To make it clear that the ordering of the photo does not have a large impact on the method—unless of course in cases like the bottle as stated below—we have also textured the coffee mug using the inverse photo ordering. From Table 1, we note that it had a slight higher error, which is mostly attributed to the manual rough alignment. Visually both textured models are practically identical (Fig. 7, right).

Bottle This handmade bottle (Fig. 7, left) would be the simplest case if not for one detail: it has a dent on one side. Since the geometry of this depression is hard to capture from the photos, it was important to align the two images that contain it first (Fig. 8, left and middle). Even so, from Fig. 8 (right), it is observable that the method could not deal perfectly with this issue.

We have also textured this model using the automatic linear alignment, and were able, without any additional tweak-

Fig. 8 Two photos used to texture the bottle model (*left and middle*), where it is possible to perceive the depression on the surface. Misalignment of the produced textured model caused by the dent on the surface (*right*)

Fig. 9 A rendering of the 3D model of a wooden pestle textured entirely using the automatic linear method as initial step (*top*). Two sequential photos of the pestle that went through the automatic initial alignment (*bottom*)

ing, to align all images but one, where not enough correspondence inliers were found in regards to the previous image. On average, around 40 inliers were retrieved out of approximately 300 correspondences for each pair of overlapping photographs. Even though the automatic first alignment gives higher mean and maximum error in comparison to a careful manual alignment, visually the difference is not significant. The main reason is that the strongest misalignments occur where there are no correspondence inliers, which are usually regions near the image's border that are given very low weight during the texture blending phase.

Pestle This object was entirely aligned using automatic correspondences, which were extremely important for two reasons: First, we were not able to texture it using the manual approach, since visually it is it very difficult to find correspondences; and second, almost no well defined edges were extracted from two overlapping images, mainly because the scratches on the wood surface are very sensitive to illumination effects. Thus, for the nonlinear iterative phase, we also used the extracted correspondences as source and target points. The result is shown in Fig. 9.

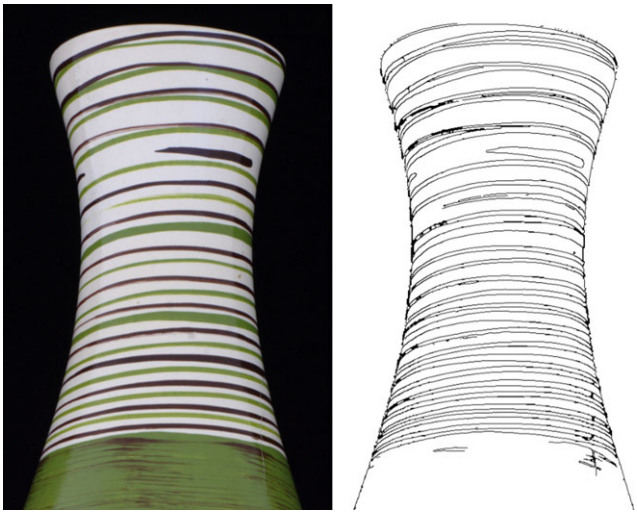

Fig. 10 A detail of the extracted contours from one photo of the vase. Note how there are almost no unique correspondence points, just a few at the stripes endpoints

Vase This is probably the most challenging model, because there are very few texture details that can be precisely tracked between photos. Most stripes complete the whole diameter, and only some endpoints can be used as good correspondences, as illustrated in Fig. 10. Even though the stripes can be aligned within two degrees of freedom, the model might rotate almost freely around the vertical axis. To achieve a good alignment, the effort for placing the initial guess for this model was considerably higher than the others as a more precise starting point was required. The automatic method also did a poor job in this case since it searches for corner points. Another note is that, since the spatial resolution of the stripes is very high, a dense mesh, of approximately 3.4M triangles, was generated to reduce aliasing artifacts when mapping the photos to per-vertex-colors.

Toy To demonstrate that the method can also be extended to a more generic scenario, we have produced a model of a toy character (Fig. 11) using the same setup. Differently from the previous datasets, in this case, the inner silhouettes of the model's geometry were also used to guide the alignment. However, since this was not the main goal of this work, these extra contours were extracted in a naive way—edge detection on the rendered image—leading to the small misalignments in the final model. Note that the final model was actually downsampled since it contains considerably less texture details in comparison to the others.

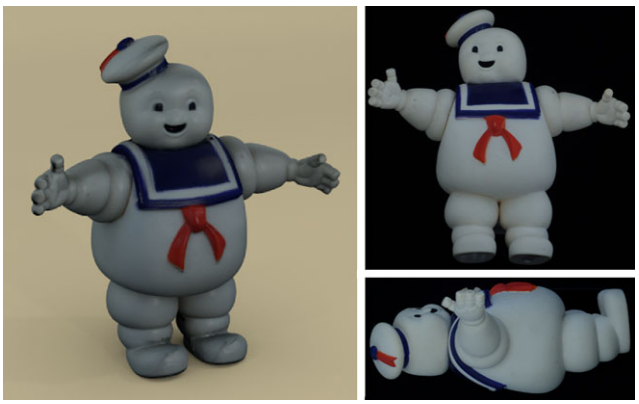

Fig. 11 Rendering of the final toy model (*left*) and two input photos (*right*)

7.1 A practical case

We put the system to test within a real digitalization campaign, where the target is a historical scientific instrument, more specifically a meridian circle. Here, we show the first results for one of its support base. The geometry was acquired using a Minotal Vivid9i scanner, and the photos using the same camera as before. However, in this case, a total of 41 photos were registered, where no single photo covers the whole object (some examples are illustrated in Fig. 1). Instead of refining the mesh to use the color-per-vertex approach, an unified texture was generated.

There are a series of factors that renders this a more challenging case than the controlled experiments above. First, the background was not homogeneous, so foreground extraction sometimes had to rely on manual hints; nonetheless, this could be further improved with more robust extraction methods. Second, since the object is extremely heavy the camera was displaced around it, so the distance and orientation of the camera relative to the object varied much more than before and we had less control over the illumination. Third, the scanner had difficulties in acquiring some parts due to the dark material, and the resulting geometric model contains artifacts which misguided the alignment of some photos producing a few ghosting effects.

Another point is that the automatic feature extraction methods, specially SIFT, had a high tendency to accumulate features in areas such as the drawn white numbers on the side, leading the linear calibration method to failure. Thus, for many photos manually selecting correspondences was the best way to achieve a good first guess, usually 7–10 points were enough. Note that, even though this model contains more geometric features than the previous ones, differently from the toy example we did not use the inner contours from the geometry in order to better certify that the proposed method was able to handle a real practical case.

In spite of all these obstacles, we were able to produce a textured model by sequentially aligning the photos. Not only the alignment was good enough for on-site evaluation but it does not contain any considerable ghosting artifact in most regions, especially those without geometric problems.

7.2 Limitations

The method cannot handle well some situations, like the dent on the bottle dataset, because the geometric feature is not apparent as a contour from any viewpoint. Apart from this special case, there is still a small amount of misalignment on other regions, even though in a much finer scale. As discussed previously (Sect. 2), there are no known methods that can achieve a perfect registration for most practical cases, and the fine-tuning is usually carried out in a post-processing stage [7, 10]. This would be specially important in cases such as described in Sect. 7.1, where a perfect alignment was not possible due to the approximated geometry.

A more important observation is that if no relevant information can be extracted that is preserved throughout the photos, the method will probably fail. This is even more critical for the automatic rough alignment, where the set of correspondences should have a minimum percentage of inliers for RANSAC to work, and a minimum distribution over the image for the linear external orientation algorithm.

8 Conclusions

We have presented a method to register high-resolution photos onto 3D models that lack geometric features. The strategy works in an incremental "shoot and register" manner, being highly suitable for on site digitalization campaigns where photos are usually taken in a sequential manner. An initial approximative camera position for each photo can be manually or automatically set using a linear method based on point feature correspondences. The registration proceeds using a combination of outer and inner contours from the rendered model and the photos. To avoid common pitfalls with the standard optimizations methods, we described a variation of the Levenberg–Marquardt formulation that uses a smoothed function as minimization criterion, and is able to handle difficult cases arising from the proposed target applications. In addition, we employ some GPU programming techniques to significantly accelerate the optimization process.

Regarding the fully automatic method, on one hand the initial registration is not yet as precise as a very careful user that takes his time to set a very precise first alignment. On the other hand, we have been able to texture objects that were not possible before with such manual approach, as the wooden pestle model, for example. Nevertheless, in both cases, the nonlinear alignment was able to significantly reduce the error.

8.1 Future works

To facilitate the use of this approach in real digitalization campaigns, more robust edge extraction and background removal methods should be implemented. However, a more pressing point is the distribution of the automatically detected features, where a way to guarantee minimum coverage would certainly lead to great improvements. Furthermore, even though it does not seem necessary for every model, a final bundle adjustment pass might help to achieve a better results.

Since we have accomplished a good speedup using GPU implementations for rendering the contours during each optimization iteration, a natural direction is to compact the buffer (using, for example, the Stream Compactor Cuda algorithm [2]) reducing it to a vector containing only the desired pixels, and build the optimization matrix directly in GPU. Then only the final summed 6×6 matrix should be transferred back and passed to the solver. We believe this could help achieve near interactive times for the convergence.

Acknowledgements The authors would like to thank the colleagues from the Visual Computing Lab (CNR-Pisa) for their help and support on using their texture blending software, and the researches and technicians from MAST (Museum of Astronomy and Affine Sciences—Rio de Janeiro) for the collaboration in the digitalization process of the meridian circle.

This work was supported by Rio de Janeiro's research funding agency—FAPERJ.

References

1. Bannai, N., Agathos, A., Fisher, R.B.: Fusing multiple color images for texturing models. In: Proceedings of the 3D Data Processing, Visualization, and Transmission, 2nd International Symposium, 3DPVT'04, pp. 558–565. IEEE Computer Society, Washington (2004)
2. Billeter, M., Olsson, O., Assarsson, U.: Efficient stream compaction on wide SIMD many-core architectures. In: Proceedings of the Conference on High Performance Graphics 2009, HPG'09, pp. 159–166. ACM, New York (2009)
3. Callieri, M., Cignoni, P., Corsini, M., Scopigno, R.: Masked photo blending: mapping dense photographic dataset on high-resolution 3d models. Comput. Graph. **32**(4), 464–473 (2008)
4. Chuang, M., Luo, L., Brown, B.J., Rusinkiewicz, S., Kazhdan, M.: Estimating the Laplace–Beltrami operator by restricting 3D functions. In: Proceedings of the Symposium on Geometry Processing, pp. 1475–1484. Eurographics Association, Geneva (2009)
5. Corsini, M., Dellepiane, M., Ponchio, F., Scopigno, R.: Image-to-geometry registration: a mutual information method exploiting illumination-related geometric properties. Comput. Graph. Forum **28**(7), 1755–1764 (2009)
6. DeCarlo, D., Finkelstein, A., Rusinkiewicz, S., Santella, A.: Suggestive contours for conveying shape. ACM Trans. Graph. **22**, 848–855 (2003)
7. Dellepiane, M., Marroquim, R., Callieri, M., Cignoni, P., Scopigno, R.: Flow-based local optimization for image-to-geometry projection. IEEE Trans. Vis. Comput. Graph. **18**, 463–474 (2012)
8. Felzenszwalb, P.F., Huttenlocher, D.P.: Distance transforms of sampled functions. Tech. rep., Cornell Computing and Information Science (2004)
9. Fiore, P.D.: Efficient linear solution of exterior orientation. IEEE Trans. Pattern Anal. Mach. Intell. **23**(2), 140–148 (2001)
10. Gal, R., Wexler, Y., Ofek, E., Hoppe, H., Cohen-Or, D.: Seamless montage for texturing models. Comput. Graph. Forum (Eurographics.) **29**(2), 479–486 (2010)
11. Guennebaud, G., Jacob, B., et al.: Eigen v3 (2010). http://eigen.tuxfamily.org
12. Harris, C., Stephens, M.: A combined corner and edge detection. In: Proceedings of the Fourth Alvey Vision Conference, pp. 147–151 (1988)
13. Lensch, H.P.A., Heidrich, W., Seidel, H.P.: A Silhouette-based algorithm for texture registration and stitching. Graph. Models **63**(4), 245–262 (2001)
14. Liu, L., Stamos, I., Yu, G., Wolberg, G., Zokai, S.: Multiview geometry for texture mapping 2d images onto 3d range data. In: CVPR'06: Proceedings of the 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, pp. 2293–2300. IEEE Computer Society, Washington (2006)
15. Lowe, D.G.: Distinctive image features from scale-invariant keypoints. Int. J. Comput. Vis. **60**, 91–110 (2004)
16. Madsen, K., Nielsen, H.B., Tingleff, O.: Methods for Non-linear Least Squares Problems, 2nd edn. (2004)
17. Matsushita, K., Kaneko, T.: Efficient and handy texture mapping on 3d surfaces. Comput. Graph. Forum **18**(3), 349–358 (1999)
18. Neugebauer, P.J., Klein, K.: Texturing 3d models of real world objects from multiple unregistered photographic views. Comput. Graph. Forum **18**(3), 245–256 (1999)

Ricardo Marroquim is a professor at the Department of Systems Engineering and Computer Science at the Federal University of Rio de Janeiro. Has MSc (2005) and DSc (2008) degree in the same institution, and was a postdoc fellow at the Visual Computing Lab—CNR, Pisa in 2009. Is a member of the Computer Graphics Lab (COPPE) and his main research interests include photo-realistic rendering, 3D reconstruction and imaging applications to cultural heritage.

Gustavo Pfeiffer is an under-graduate student at the Computer and Information Engineering course at the Federal University of Rio de Janeiro. He is a member of the Computer Graphics Lab (COPPE) and his main research interest are optimization methods for computer graphics, image processing, BRDF extraction and stochastic optimization methods.

Felipe Carvalho is a PhD student at the Systems Engineering and Computer Science at the Federal University of Rio de Janeiro where he received his MSc degree in 2010. He received his BSc degree in Computer Science at the Federal University of Piauí in 2007. His main research are interactive graphics, rendering, image processing, graphics hardware and game techniques.

Antonio A.F. Oliveira got his BSc and DSc degrees in Electronics and Systems Engineering, respectively, both from the Federal University of Rio de Janeiro (UFRJ). He also holds an MSc in Mathematics. He has been working in the Computer Graphics Laboratory of UFRJ since 1989 and has taken part in many research projects in the areas of Computer Graphics and Image Processing.