Texturing 3D models with low geometric features

Ricardo Marroquim, Gustavo Pfeiffer, Felipe Moura de Carvalho, Antonio A. F. Oliveira
Laboratório de Computaçao Gráfica (LCG) - COPPE - UFRJ

***Abstract*—This paper presents a texture projection and alignment method specifically tailored for objects with low geometric features. A common way to build virtual replicas is to acquire the geometry with a 3D scanner and model its reflectance properties (i.e. colors) by projecting photos onto the 3D surface. To correctly align each photo it is necessary to retrieve the camera's extrinsic matrix, which is usually achieved by optimization algorithms that match geometric features of the model with their corresponding ones on the photos. However, when dealing with cultural heritage artifacts or mechanical pieces, for example, some models inherently lack significant geometric detail.**

We propose a method to automatically extract these features from high resolution photos and register them in a sequential manner using a variation of the contour-based approaches. The algorithm requires minimum user intervention as only an initial guess of the camera's direction is required for each photo. In addition, we describe an improved formulation of the optimization method based on a smooth function that avoids typical local minima found in this scenario. The results are illustrated with high-resolution textured models produced by our system.

***Keywords*-texture-to-geometry registration; 3D virtual replica; high-resolution texture mapping; least-squares minimization;**

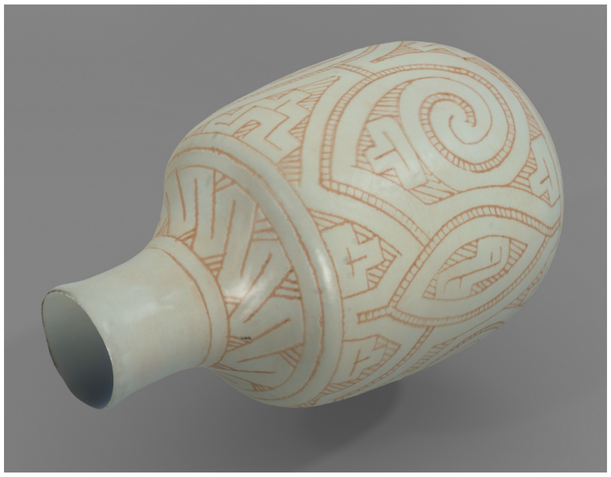


Fig. 1. A rendering of a model textured with the proposed method.

I. Introduction

Building 3D virtual replicas of real objects is a challenging topic and has received a broad attention during the last years: with modern scanners it is possible to acquire surface geometry with great fidelity. Even though the process is not generally fully automatic, and may be laborious depending on the subject, it has reached a mature level and many commercial systems are able to produce high resolution models for a wide range of applications.

Unfortunately, the acquisition and mapping of reflectance properties has lagged behind. Since most digitalization equipments lack quality color registration embedded systems, this stage is usually carried out separately through one of two methods: acquiring BRDFs or projecting photographs by calibrating the camera's parameter. 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.

Methods to solve the camera calibration problem are mainly based on finding correspondences between images or between an image and a 3D surface. Most approaches rely on geometric features to guide an optimization method, specially when the model does not yet contain any color information. However, in some cases it lacks significant, or almost any, geometric characteristics invalidating, or severely limiting, these strategies. Even when the model is partially textured, matching issues may still arise when there are no well defined correspondences, such as drawings, characters or nonrepetitive themes. Examples of such situations can be found in many applications: *cultural heritage* (e.g. vases), *engineering* (e.g. mechanical pieces), or even both when dealing with historical scientific instruments.

Another consideration is the ability to perform well in an out of the lab environment. In many digitalization campaigns the object in question cannot be displaced – specially in the context of *cultural heritage* – and it becomes important to have an immediate feedback of the progress and the quality of the acquired photos. This not only may avoid another visit to the site in order to repair any misregistration or uncovered regions, but throughout the interventions of a restoration process, for example, there might not be a second chance to acquire the color information at a specific stage. Registering a photo right away also helps to deal with subjects that have motifs or no predominant geometric direction, as it might be a difficult and tedious task to latter distinguish between the photos in order to complete the alignment.

Given this scenario, the present paper focus on three specific problems related to the calibration of the camera parameters: mapping reflectance information onto 3D models with low surface details; handling textures with no predominant characteristics; and building an efficient method that works incrementally, i.e. in a **“shoot and register”** fashion.

Our solution resorts to only the model’s external contours coupled by texture features from previous registered photos (Section IV). The alignment is carried out by an optimization method specifically designed to handle the aforementioned conditions (Section V). The approach works in a sequential manner during the acquisition process, and with the registered photos we were able to produce high quality texture maps (or per-vertex color) as depicted in Section VI.

Contributions: Briefly, the two main proposed contributions are:

1) an incremental camera calibration method to map high-resolution photographs to 3D models lacking relevant geometric features, and/or without obvious texture correspondences;
2) and the formulation of an optimization approach for texture registration based on a smoothing function, and its detailed derivation.

II. Related work

The use of contours to drive the image-to-geometry registration is not new. Neugebauer and Klein [1] start with an user interaction point correspondence 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.

Matshushita and Kaneko [2] 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 these locations. 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. [3]. 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. [4] 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 [6], while others rely on a local texture blending [7]. 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 ([8], [9], [6], [10]).

III. Method overview

The input for the system is a set of photos and the 3D geometry. Fig. 2 depicts two input photos and the untextured coffee mug input model used to produce the final colored model.

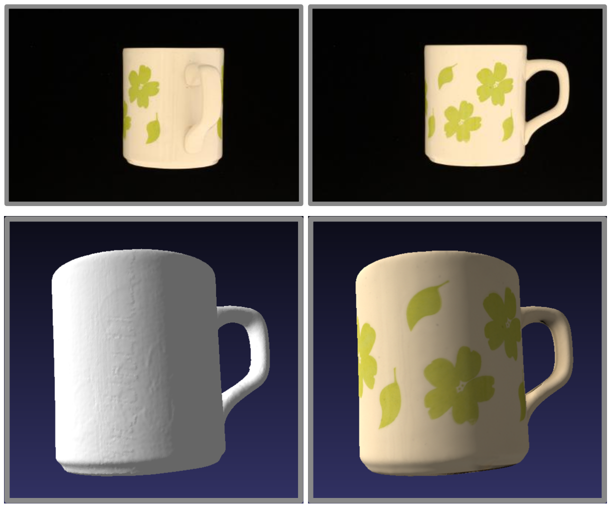


Fig. 2. Top row: two of the size photos registered onto the 3D model (bottom row left) to produce the final colored model (bottom row right).

Assuming that at least one image has already been mapped – registering the first is actually the easier case – we start by choosing the next target photo and placing the model in a similar position. The initial camera parameters are given by the current view and by selecting a point on the model’s surface. Then, the target image’s contours are extracted and the distance transform computed once, followed by an optimization process. For each iteration the model’s contours are projected onto the target image plane to compute the minimization error. At last, after convergence, the calibrated camera is used to register the photo onto the model. Fig. 3 illustrates this process.

IV. Contours

The model’s contours are a set of points on the surface satisfying:

$$n_i^T(p_i - c) = 0 \tag{1}$$

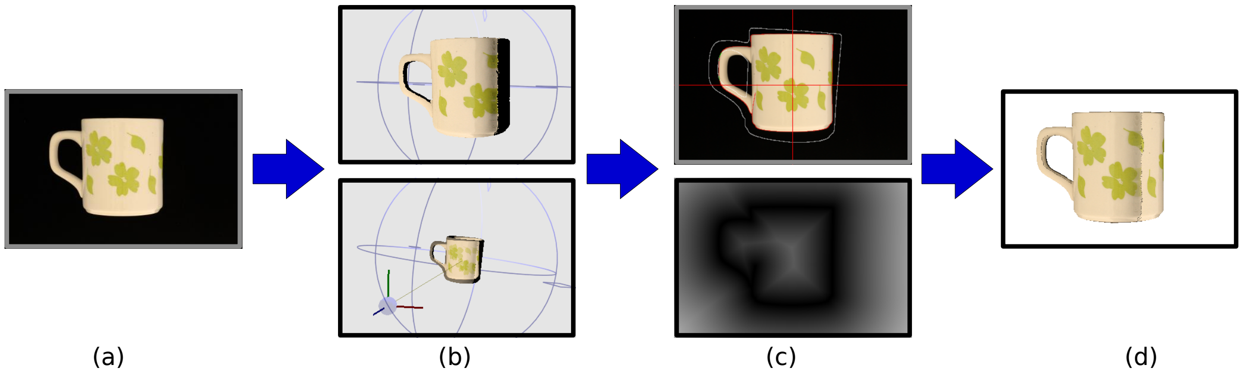


Fig. 3. System overview: **(a)** the target photo to be registered; **(b top)** the partially textured model is placed in a position that resembles the pose from the target photo, **(b bottom)** and the camera direction is estimated with a simple mouse click; **(c top)** the optimization system in the process of matching the projected model's contour with those of the photo, **(c bottom)** using a smoothed distance transform; **(d)** a rendering of the model using the calibrated camera matrix matching the target photo (a); note that the seams are intentionally left for visual feedback during the alignment, they are removed in a final texture blending stage.

where p_i is a point on the surface S with normal n_i, assuming that the camera is positioned at c and directed at the center of the object. Contours are typically a set of disconnected loops on the surface, however our method considers only the subset of these loops that separates the model from the background, i.e. the outer contours, or outlines. These boundaries are extracted by a simple two-pass rendering technique [11], resulting in a bitmap representation of the contour. Nonetheless, the corresponding 3D coordinates are still required for the optimization phase (Section V), hence, in a last rendering pass a simple shader program is employed to gather this information in a framebuffer.

The same idea follows on to the photos' contours, i.e. the outlines between the object's representation and the background. To this end, common image processing techniques are applied: a simple thresholding method is used to separate the background, while the inner contours are extracted using a standard Canny [12] method. After the completion of at least one registration, the inner contours from the already registered textures are used as reference for the next alignments.

V. Optimization

In this section the objective function used for the least-squares minimization is derived, and the introduced variations are explained. Briefly, it is a variation of a Levenberg-Marquardt implementation [13] that avoids common local minima encountered 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*).

A. *objective function*

There are many forms of modeling the similarity between two contours. One intuitive model is to minimize the sum of the squares of the distances between each projected point of the model contour and the respective nearest point of the photograph contour. Note that this function is not a standard least squares objective function since the target point (i.e. the closest point) changes throughout the optimization. Thus, let $y(p,k)$ be a function that maps a camera configuration k and a 3-dimensional point p in global coordinates to the point it generates on the model's contour, and let $q(y)$ be the point in the photograph's contours that is closest to y. The function we would like to minimize is:

$$f(k) = \frac{1}{N}\sum_{i=1}^{N} ||y(p_i,k) - q(y(p_i,k))||^2 \tag{2}$$

where we assume there is one 3-dimensional point p_i for each pixel in the model's contour (Section IV). Also, note that the number of points N depends on the view direction and may change through the optimization, so we divide the objective function by N. In this way, the optimizer is not biased towards decreasing the number of contour points.

The y function can be written as

$$y(p,k) = \pi(x(p,k)) \tag{3}$$

$$x(p,k) = R(k)p_i + t(k) \tag{4}$$

where π is the projection matrix operator, x is the correspondent point to p in camera coordinates and $R(k)$ and $t(k)$ are, respectively, the rotation matrix and translation vector that we are optimizing.

The operator π is defined by the perspective division and the intrinsic matrix multiplication:

$$\pi(x) = Y\frac{x}{e_3^T x} \tag{5}$$

where e_3 is the vector $(0,0,1)^T$ and Y is the intrinsic matrix. This expression can be trivially derived as will be shown.

The $q(y)$ function is more delicate. First, because it is not a continuous function, since target points are given in pixel coordinates (integers); so we would have to interpolate it by a smooth function. In derivative-based optimization methods (such as Newton, Quasi-Newton and Levenberg-Marquardt), it is sufficient to approximate the contour around q_i by a quadratic curve, i.e. with normal and curvature defined. Here, we approximate it by a linear curve.

The R and t functions form the current extrinsic matrix. We will not formally define them in function of k (since, in fact, we do not even represent k in our model, only the extrinsic matrix itself), but we can consider that, in the j-th iteration of the method we have a camera configuration k_j where $R(k_j) = R_j$ and $t(k_j) = t_j$. Without loss of generality, we consider that the rotation always precedes the translation, so we can model $R(k)$ and $t(k)$ as:

$$\begin{bmatrix} R(k) & t(k) \\ 0^T & 1 \end{bmatrix} = \begin{bmatrix} \tilde{R}_j(k) & \tilde{t}_j(k) \\ 0^T & 1 \end{bmatrix} \begin{bmatrix} R_j & t_j \\ 0^T & 1 \end{bmatrix} \tag{6}$$

where $\tilde{R}_j(k_j) = I$ and $\tilde{t}_j(k_j) = 0$. In other words, we are optimizing the camera increment at each iteration, instead of always optimizing in regards to the initial conditions. This manipulation reduces the problem of deriving $R(k)$ and $t(k)$ at any point k to deriving them only where $R = I$ and $t = 0$, which can be done via Taylor series.

B. Derivatives

1) $\pi(x)$: The jacobian of $\pi(x)$ (eq. 5) is given by:

$$\partial_x \pi(x) \delta x = Y \frac{\delta x}{e_3^T x} - Y \frac{x}{(e_3^T x)^2} e_3^T \delta x \Leftrightarrow$$

$$\partial_x \pi(x) = Y[\frac{I}{e_3^T x} - \frac{x e_3^T}{(e_3^T x)^2}] \tag{7}$$

2) $q(y)$: If y is not on the middle axis of the contour, the jacobian of $q(y)$ can be calculated using its linear approximation (i.e. the orthogonal projection on the line tangent to the contour):

$$q(y) = q_i + (I - \hat{n}_i \hat{n}_i^T)(y - q_i) + O(||y - y_i||^2) \Rightarrow$$

$$\partial_y q(y) \delta y = (I - \hat{n}_i \hat{n}_i^T) \delta y \Leftrightarrow \partial_y \{y - q(y)\} = \hat{n}_i \hat{n}_i^T \tag{8}$$

where $\hat{n}_i$ is the normal to the curve at q_i. If $y_i - q_i \gg 0$ we can assume $\hat{n}_i = \frac{y_i - q_i}{||y_i - q_i||}$, but if $y_i - q_i \approx 0$, this approximation may fail, since q is given as a pair of integers.

When the normal is not defined (for instance, at a corner, or at the end of an open curve), the target q will not change with y, so we should consider $q(y)$ constant, which means $y - q(y)$ behaves as a distance-to-point function, and $\partial_y \{y - q(y)\} = I$. One may consider the corner as a smooth curve and tend the curvature at q to infinite, so that using simply $\hat{n} = \frac{y-q}{||y-q||}$ helps keeping the jacobian continuous. There is also a heuristic that interpolates between $\hat{n}_i \hat{n}_i^T$ and I according to the curvature at q, giving a greater weight to the $I - \hat{n}_i \hat{n}_i^T$ portion as the curvature increases, thus yielding $\partial_y \{y - q(y)\} \approx I$ on corners.

3) $\tilde{R}(k)$ and $\tilde{t}(k)$: The jacobian matrices of $\tilde{R}(k)$ and $\tilde{t}(k)$ are more complicated to demonstrate. We want to apply a 4x4 transformation on the left of the current extrinsic matrix, which can be represented by a rotation around a unitary vector $\hat{r}$ of θ degrees and a translation t:

$$x^+ = \tilde{R}(k)x + \tilde{t}(k) \Leftrightarrow$$

$$x^+ = \hat{r}\hat{r}^T x + \cos(\theta)(I - \hat{r}\hat{r}^T)x + \sin(\theta)\hat{r} \times x + t \tag{9}$$

where x^+ should be interpreted as the position of x after this left multiplication on the extrinsic matrix.

If we relax the condition that $\hat{r}$ has to be unitary, changing $\hat{r}$ to $\frac{r}{||r||}$ and using $\theta = ||r||$, then we obtain:

$$x^+ = \frac{rr^T}{r^T r} x + \cos(||r||)(I - \frac{rr^T}{r^T r})x + \sin(||r||)\frac{r}{||r||} \times x + t$$

This equation is singular at $r = 0$, although we know $\lim_{(r,t)\to 0} x^+ = x$. If we replace $\sin ||r||$ and $\cos ||r||$ for their Taylor series, we obtain:

$$x^+ = x + r \times x + t + \frac{(rr^T - ||r||^2 I)x}{2} - \frac{||r||^2 r \times x}{3!} + ... \Rightarrow$$
$$x^+ = x + r \times x + t + O(||r||^2) \tag{10}$$

yielding the derivative:

$$\partial_{(r,t)} x^+ \begin{bmatrix} \delta r \\ \delta t \end{bmatrix} = \delta r \times x + \delta t \Rightarrow$$

$$\partial_{(r,t)} x^+ = \begin{bmatrix} -(x\times) & I \end{bmatrix} \tag{11}$$

where $(x\times)$ should be interpreted as the matrix of the cross product with x.

4) Final expression: The objective function (eq. 2) can now be derived with respect to the r and t vectors of the subsection above:

$$\partial_{(r,t)} f = \frac{2}{N} \sum_{i=1}^{N} (y_i - q_i)^T J_i$$

$$J_i = (\partial_y \{y - q\})(\partial_x \pi)(\partial_{(r,t)} x^+) \Rightarrow$$

$$J_i = \hat{n}_i \hat{n}_i^T Y(\frac{I}{e_3^T x_i} - \frac{x_i e_3^T}{(e_3^T x_i)^2}) \begin{bmatrix} -(x\times) & I \end{bmatrix} \tag{12}$$

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 $(A + \lambda diag(A))x = b$, where:

$$A = \sum_{i=1}^{N} J_i^T J_i \tag{13}$$

$$b = \sum_{i=1}^{N} J_i^T (q_i - y_i) \tag{14}$$

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 y and q at k_j. Thus, the expression for ρ we use is:

$$\rho = \frac{f(k_j) - f(k_{j+1})}{\frac{1}{N}\langle x, \lambda diag(A)x + b\rangle} \tag{15}$$

C. distance transform

The target image does not change during the minimization process, however, for every iteration the source points are displaced and their corresponding image contour pixel (q_i) must be retrieved. To this end, a distance transform, as described by Felzenszwalb and Huttenlocher [14], is computed once for every photo serving as target. The distance transform returns for every image pixel the closest pixel on the target photo's contour and the corresponding distance.

D. Smoothed function approach

The disadvantage of minimizing the distance to the closest point is that the point q_i is generally not the ideal correspondence point to y_i, and, depending on the initial guess, it may be very far from the actual point y_i should match. This aspect turns this objective function seriously vulnerable to local minima.

For instance, consider the situation where we have an internal contour with two parallel lines, and we try to translate the object perpendicularly to those lines. In a very simplified mathematical example for this situation, we are optimizing 2 object points to 2 target points in one dimension, where the target points are located at a and $-a$, and the object points are located at $x + a$ and $x - a$, as in Fig. 4:

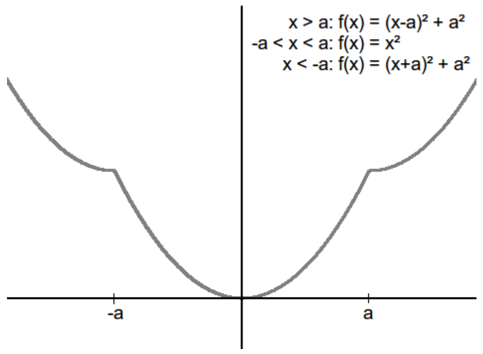
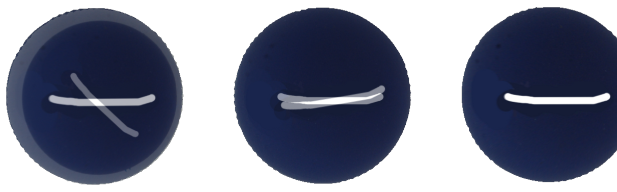


Fig. 4. Objective function matching points $x - a$ and $x + a$ to a and $-a$.

This function is not derivable at $x = a$, since $\lim_{x\to a^-} f'(x) = 2a$ and $\lim_{x\to a^+} f'(x) = 0$. Thus, if the initial guess has $x > a$, then a behaves as a local minimum, because the damping factor λ forces the Levenberg-Marquardt method to take a step that is lesser than the necessary to solve the linearized function.

A simple way of solving this problem is to smooth the distance function. In other words, we apply a Gaussian filter to the distance transform of each image (call it $d(y)$) and compute its gradient ($g(y)$) to use on the derivative. We replace $\hat{n}\hat{n}^T$ for $g(y)^T$ in the jacobian J_i (note that the height of the matrix also changes), and $q_i - y_i$ for $-d(y)$ in the formula for b.

Fig. 5 illustrates the typical situation where the smooth function is able to arrive at the correct configuration, while the standard method stops at a local minimum. Note that the white stripe is turned into practically two parallel lines after extracting the inner contours.



(a) initial setup (b) closest-point distance (c) smoothed distance

Fig. 5. Comparison of the standard closest-point distance with the smooth function: **(a)** 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; **(b)** the standard method gets stuck in a local minimum, **(c)** while the proposed smooth function is able to achieve the exact match.

E. initial guess

To compute the initial guess the system must be supplied with the camera's direction and position. Aiming at minimum user interaction, the first is obtained by the vector from the current view point to an interactively selected point on the model's surface; while a distance along this vector is given by the estimated 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 information – there is enough information to build the camera's initial extrinsic matrix.

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.

F. iterations

For each iteration the partially textured model is projected onto the plane of the current target photo. The rendered image is read back, and for each contour pixel found, the original 3D coordinates are retrieved from the framebuffer, as explained in Section IV. Following, the smoothed versions of equations 13 and 14 (as described in Section V-D) are computed and the system is solved by matrix decomposition using the Eigen library [15].

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 Equation 15, 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. [13].

To speed up the convergence, a multi-resolution strategy is employed by down-sampling 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.

VI. Results

The results are illustrated with three datasets: a coffee mug, a bottle, and a flower vase. We have intentionally chosen objects that were either handmade or hand-painted to avoid very clear contours, which would make the correspondences 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 latter 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 multi-resolution scheme: 400×268, 800×536, and the original 2200×1474.

Once all photos have been registered, the final colored model were generated using the approach by Callieri et al. [7]. All models were produced with a per-vertex color, but texture maps could be similarly created. Fig. 6 shows renderings of the final results.

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 top 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.

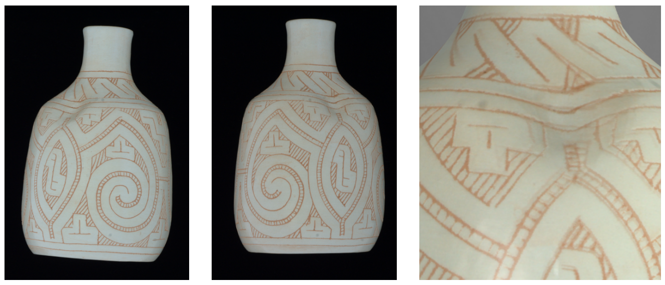


Fig. 7. Two photos used to texture the bottle model (left and middle), where it is possible to perceive the depression on the surface. Misalignment of the produced textured model caused by the dent on the surface (right).

Bottle: This handmade bottle would be the simplest of all three 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. 7 left and middle). Even so, from Fig. 7(right) it is observable that the method could not deal perfectly with this issue.

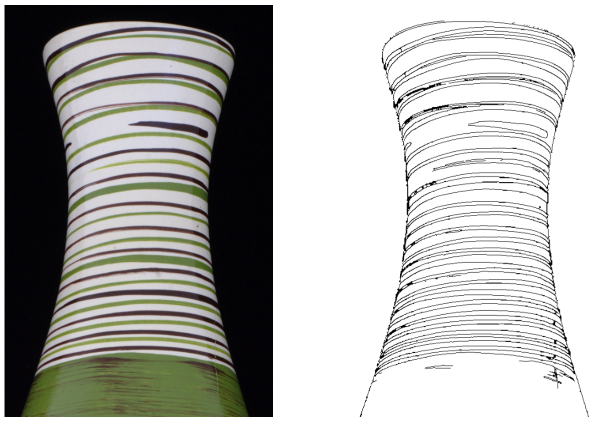

(a) detail of a photo (b) extracted contours

Fig. 8. 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. 8. 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. 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.

We have dedicated only a minimum amount of effort on what regards performance optimizations. Nonetheless, after setting the initial guess, each photo is registered in approximately 2-4 minutes. Table I 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. 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.

As can be noticed from these values, counterintuitively, the coffee mug had the highest errors. This was probably due to the straight cylindrical shape (while the others are curved

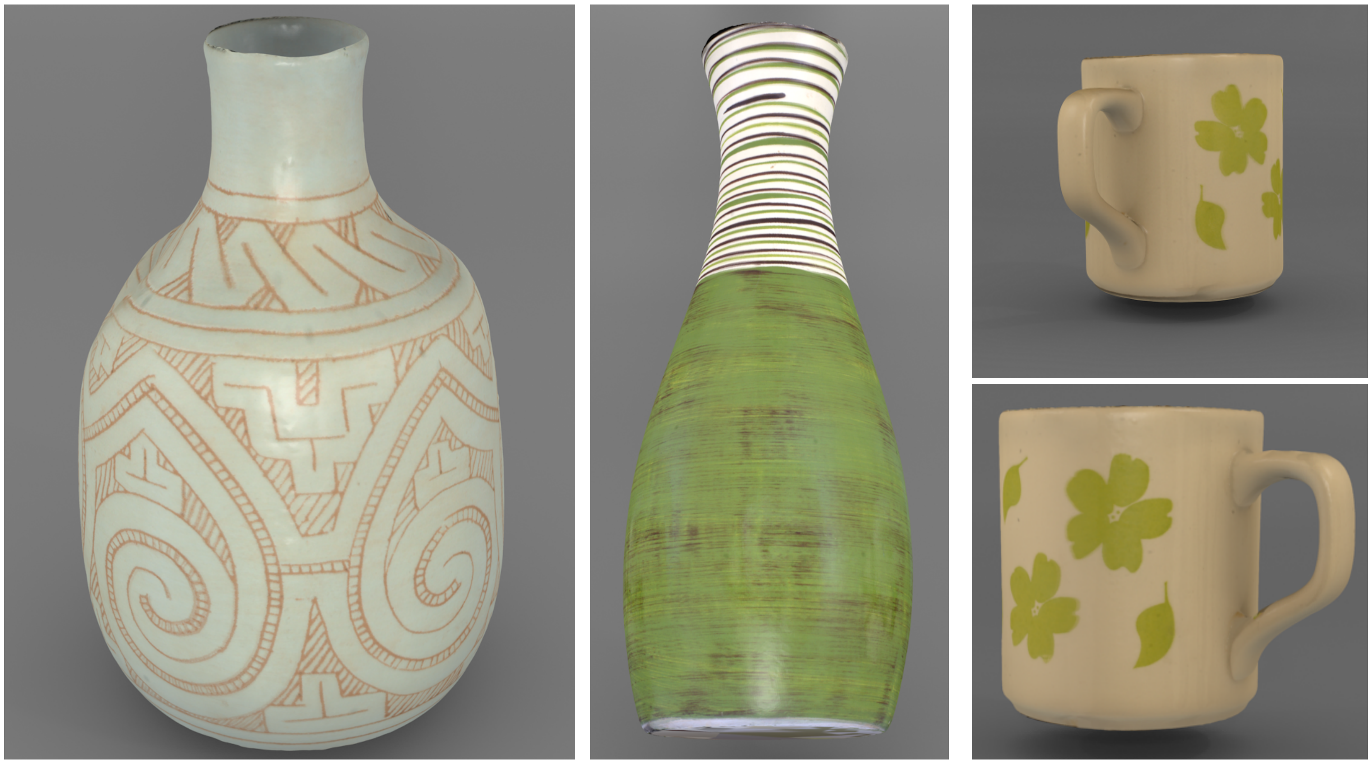

(a) handmade bottle (b) hand-painted vase (c) coffee mug (top) and same model with the inverse alignment sequence (bottom)

Fig. 6. Ray-traced colored models.

in regards to the vertical axis), or the blurry drawings that, when viewed from different angles, might mislead the contour extraction algorithm.

Another remark concerns the ordering in which the photos are aligned. To make it clear that it should not have a large impact on the method - unless of course in cases like the bottle as stated above - we have also texture the coffee mug using the inverse photo ordering. From Table I 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. 6 (c)).

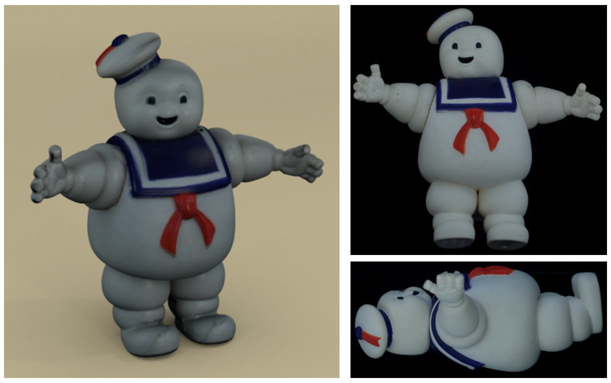


Fig. 9. Rendering of the final toy model (left) and two input photos (right).

Finally, to demonstrate that the method can also be extended to a more generic scenario, we have produced a model of a toy character (Fig. 9). 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.

A. *Limitations*

In this section we discuss the limitations only inside the scope of the low-geometric features proposal. In the following section we also suggest some improvements to the method.

First, the initial condition must not be very far from the ideal registration. All the same, it usually does not take more than 15-20 seconds to set the worst case photos. From this configuration to the final registration the camera parameters undergo great variations, hence, it would be virtually impossible (or at least extremely laborious) to achieve the same result manually.

Another limitation is that 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

TABLE I
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 (inverse)	240K	1.9M	6	3.34	10.2	9	10	15
Bottle	115K	2.7M	6	1.12	3.14	10	12	14
Vase	216K	3.4M	5	0.92	2.10	9	10	16
Toy	1.1M	813K	6	2.43	6.41	7	9	13

amount of misalignment on other regions, even though in a much finer scale. As discussed previously (Section II), 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 pos-processing stage [6], [10].

Finally, if no relevant information can be extracted that is preserved throughout the photos, the method will probably fail. We believe it is possible to align based not only on painted or carved contours, but also on features such as paint weariness or scratches for example; however, the method will only work if it is possible to clearly extract the same contours from different angles.

VII. 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" way, being highly suitable for on site digitalization campaigns. After setting an initial approximative camera position for each photo, the registration is automatic, and immediate feedback is possible.

The elected framework uses 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. We are able to produce high-quality textured models in the order of tens of minutes.

A. Future works

We strongly believe that the correspondences could be greatly improved with better descriptors, such as extracting corners or parametric curves from the textures, for example. This point would specially improve the interactive process by driving the initial guess towards a more automatic approach. Likewise, another option is to attribute different weights to the outer and inner contours. Excess of texture details might strongly influence the optimization method, if the inner contours have higher priorities over the external ones, the camera parameters might be wrongly matched due to incorrect perspective correspondence.

Acknowledgment

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

References

[1] P. J. Neugebauer and K. Klein, "Texturing 3d models of real world objects from multiple unregistered photographic views," *Computer Graphics Forum*, vol. 18, no. 3, pp. 245–256, 1999.
[2] K. Matsushita and T. Kaneko, "Efficient and handy texture mapping on 3d surfaces," *Comput. Graph. Forum*, vol. 18, no. 3, pp. 349–358, 1999.
[3] H. P. A. Lensch, W. Heidrich, and H.-P. Seidel, "A silhouette-based algorithm for texture registration and stitching," *Graphical Models*, vol. 63, no. 4, pp. 245 – 262, 2001. [Online]. Available: http://www.sciencedirect.com/science/article/B6WG3-45FC0BR-4/2/48af481b257e05268d210ebafdd8221d
[4] L. Liu, I. Stamos, G. Yu, G. Wolberg, and S. Zokai, "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*. Washington, DC, USA: IEEE Computer Society, 2006, pp. 2293–2300.
[5] M. Corsini, M. Dellepiane, F. Ponchio, and R. Scopigno, "Image-to-geometry registration: a mutual information method exploiting illumination-related geometric properties," *Computer Graphics Forum*, vol. 28, no. 7, pp. 1755–1764, 2009. [Online]. Available: http://vcg.isti.cnr.it/Publications/2009/CDPS09
[6] R. Gal, Y. Wexler, E. Ofek, H. Hoppe, and D. Cohen-Or, "Seamless montage for texturing models," *Computer Graphics Forum (Eurographics)*, vol. 29, no. 2, pp. 479–486, 2010.
[7] M. Callieri, P. Cignoni, M. Corsini, and R. Scopigno, "Masked photo blending: mapping dense photographic dataset on high-resolution 3d models," *Computer & Graphics*, vol. 32, no. 4, pp. 464–473, Aug 2008, for the online version: http://dx.doi.org/10.1016/j.cag.2008.05.004. [Online]. Available: http://vcg.isti.cnr.it/Publications/2008/CCCS08
[8] N. Bannai, A. Agathos, and R. B. Fisher, "Fusing multiple color images for texturing models," in *Proceedings of the 3D Data Processing, Visualization, and Transmission, 2nd International Symposium*, ser. 3DPVT '04. Washington, DC, USA: IEEE Computer Society, 2004, pp. 558–565. [Online]. Available: http://dx.doi.org/10.1109/3DPVT.2004.67
[9] M. Chuang, L. Luo, B. J. Brown, S. Rusinkiewicz, and M. Kazhdan, "Estimating the laplace-beltrami operator by restricting 3d functions," in *Proceedings of the Symposium on Geometry Processing*. Eurographics Association, July 2009, pp. 1475–1484.
[10] M. Dellepiane, R. Marroquim, M. Callieri, P. Cignoni, and R. Scopigno, "Flow-based local optimization for image-to-geometry projection," *IEEE Transactions on Visualization and Computer Graphics*, vol. 99, no. PrePrints, 2011.
[11] D. DeCarlo, A. Finkelstein, S. Rusinkiewicz, and A. Santella, "Suggestive contours for conveying shape," *ACM Trans. Graph.*, vol. 22, pp. 848–855, July 2003. [Online]. Available: http://doi.acm.org/10.1145/882262.882354
[12] J. Canny, "A computational approach to edge detection," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 8, pp. 679–698, November 1986. [Online]. Available: http://portal.acm.org/citation.cfm?id=11274.11275
[13] K. Madsen, H. B. Nielsen, and O. Tingleff, "Methods for non-linear least squares problems," Richard Petersens Plads, Building 321, DK-2800 Kgs. Lyngby, p. 60, 2004, 2nd edition.
[14] P. F. Felzenszwalb and D. P. Huttenlocher, "Distance transforms of sampled functions," Cornell Computing and Information Science, Tech. Rep., 2004. [Online]. Available: http://hdl.handle.net/1813/5663
[15] G. Guennebaud, B. Jacob *et al.*, "Eigen v3," http://eigen.tuxfamily.org, 2010.