Laboratório VISGRAF
Instituto de Matemática Pura e Aplicada

A Graph Cut Approach to Texture Synthesis on 3D Surfaces

Fernanda Andalo
Luiz Velho (orientador)

Technical Report TR-2014-02 Relatório Técnico

May - 2014 - Maio

A Graph Cut Approach to Texture Synthesis on 3D Surfaces

Fernanda A. Andaló[1], **Luiz Velho**[1]

[1]Instituto Nacional de Matemática Pura e Aplicada – IMPA

Abstract. *We present a novel method for synthesizing texture over a 3D model from 2D photographs of the modeled object. By considering an optimization approach, we subdivide the mesh into patches and assign, to each patch, a stack of possible textures taken from the input images. For each image stack, we employ a multilabel graph cut algorithm to generate a single texture. With this simple and flexible approach, it is possible to create textured models for use in computer graphics, overcoming some problems in camera registration and geometry reconstruction.*

1. Introduction

Some areas, such as Culture Heritage and Architecture, need to represent their studied objects as 3D models. To give these models a more realistic appearance, not only shape information needs to be encoded, but also reflectance properties. However, not all acquisition methods are able to capture such information together with the object geometric data to produce, for example, RGB texture maps. In this case, the process can be made *a posteriori* using image-based modeling.

A possible pipeline to model objects based on images is: take several photographs around the object that needs to be modeled; register and calibrate the photographs using bundle adjustment [11]; generate a dense point cloud that represents the object [8]; and reconstruct the surface to obtain a dense triangle mesh [5]. This pipeline was successfully used to recover 3D footwear impression from photographs [1].

The image-base modeling pipeline provides the set of camera parameters associated with the input images, which allow us to map the photographs onto the modeled surface. This information can be used, in several ways, to also generate texture.

The problem of generating a seamless texture over a surface from photographs have been studied in literature, comprehending several techniques that create a single texture by blending the input images [2], usually resulting in ghosting and blurring artifacts when the textures are geometrically misaligned [9]; by image stitching [10], where additional criteria must be used to handle image calibration errors; and semi-automated tools that allow users to assign texture to meshes [12].

Our proposed texturing process starts from a 3D model of an object and several photographs of it, with known camera parameters. The goal is to subdivide the mesh in patches, and apply a texture to each one. A patch texture is generated by combining RGB information from all the image parts that cover the related patch.

2. Image-based texturing

Our strategy has two stages: mesh subdivision into patches; and texture synthesis for each patch.

To subdivide the mesh into patches we have applied a variational surface partition scheme [13]. For each subdivision, we generate a stack of image parts that cover it. And to generate a single texture from the image stack, we apply a multilabel graph cut algorithm [3].

Each one of the stages is detailed in the next subsections.

2.1. Surface partitioning

Our subdivision strategy is the one described in [13], where the authors adopt a variational approach to partition surfaces and construct atlas structures. Here an atlas structure is a set of charts defined by a piece-wise parametrization, such that each chart is associated with regions of the input images through a projective mapping.

Assuming that we have a triangle mesh of an object and several photographs of it with known camera parameters, the goal is to partition the mesh surface into a set of patches. The surface partitioning is posed as a global optimization problem that looks for a partition that minimizes the number of patches and the atlas mapping distortion through a distortion-based metric that takes into consideration the projective mapping of each camera.

The primary objective in [13] is to minimize distortion while building the atlas, such that large texture distances are not mapped onto small surface distances.The strategy is to define an energy functional for the mapping and minimize it. In [13], the authors adopted the atlas mapping distortion as the energy functional, and a heuristic in order to not produce too many charts.

The process has two stages: visibility calculation and the surface partition itself. The visibility is computed by an extension of the z-buffer algorithm [6], based in the hardware-accelerated OpenGL rendering.

The second stage, surface partitioning, considers a trade-off between the number of charts and the mapping distortion. In order to accomplish this, the scheme iteratively tries to add charts to the solution and to grow them. The scheme considers the low-distorted face to each camera to be a seed for the chart growing process, based in the distortion-based metric

$$\mathcal{D}(f, c) = ||n_f + n_c||^2 \cdot a_f, \tag{1}$$

where c represents the camera with direction n_c, and f the face with direction n_f and area a_f.

For each seed, a chart is created and the chart growing step is performed, assigning every face of the mesh to a chart. A new chart is added to the atlas by setting the face with the biggest distortion error $\mathcal{D}(f_i, c_i)$ as seed. New charts are added until we cannot find faces with their best camera different from its neighbors' best cameras (those with low-distortion in respect to the neighbors).

The chart growing process consists in assigning each face to a chart represented by a seed. The partition is created by growing all the charts simultaneously using a flooding algorithm [7] and the $\mathcal{D}$ metric. The process clusters faces with low distortion $\mathcal{D}(f, c)$, in order to obtain a better partition. Figure 1 shows an example of the surface partition output.

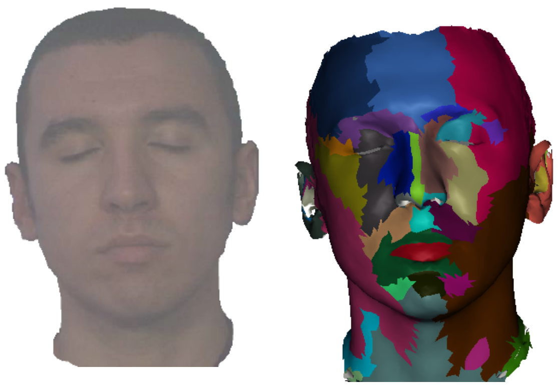

Figure 1. One of the input images and the subdivided mesh. Each colored region represents a patch.

2.2. Single texture generation

The texture generation process starts from the subdivided mesh and the input images. For each patch generated in the previous process, we create a stack of images that cover entirely or a region of the patch. The mapping between the images and the surface domain is done by projecting each input image to the patch boundary, using the correct camera parameters.

To create a single seamless texture for a patch, we apply a multi-label graph cut algorithm to the correspondent image stack, as a multidimensional optimization tool. And even though graph cuts provide an inherently binary optimization, they can be used for multi-label energy minimization [3].

The idea is to create a single image I_{tex} from a stack of N images, $I_1, \ldots, I_N$, of size $M \times M$. For each pixel $I_{tex}(i, j) \in I_{tex}$, where $i, j = 1, \ldots, M$, we must assign an intensity value, or label, from a set of possible intensity values $\mathcal{L} = \{0, \ldots, 255\}$. This assignment takes into consideration the intensity values of pixels $I_k(i, j) \in I_k$, where $k = 1, \ldots, N$, with same 2D locality (i, j). Figure 2 illustrates this formulation.

Apart from images $I_1, \ldots, I_N$, we create N additional images, $A_1, \ldots, A_N$, that represent the angle between the normal of correspondent mesh face and each camera. For example, the value $A_k(i, j)$ is the angle between the normal of the mesh face projected onto pixel $I_k(i, j)$ and the camera represented by input image k. Figure 3 shows a complete example of this formulation, i.e., the input images, the image stack for one of the generated patches, and the images representing the angles.

Let $f(p)$ be the intensity label assigned to pixel $p \in I_{tex}$, and f be the collection of such assignments for all pixels in I_{tex}. Then the energy to be minimized is

$$E(f) = \sum_{p \in I_{tex}} D(p, f(p)) + \sum_{\substack{(p,q) \in \mathcal{N} \\ p,q \in I_{tex}}} V(p, q, f(p), f(q)), \tag{2}$$

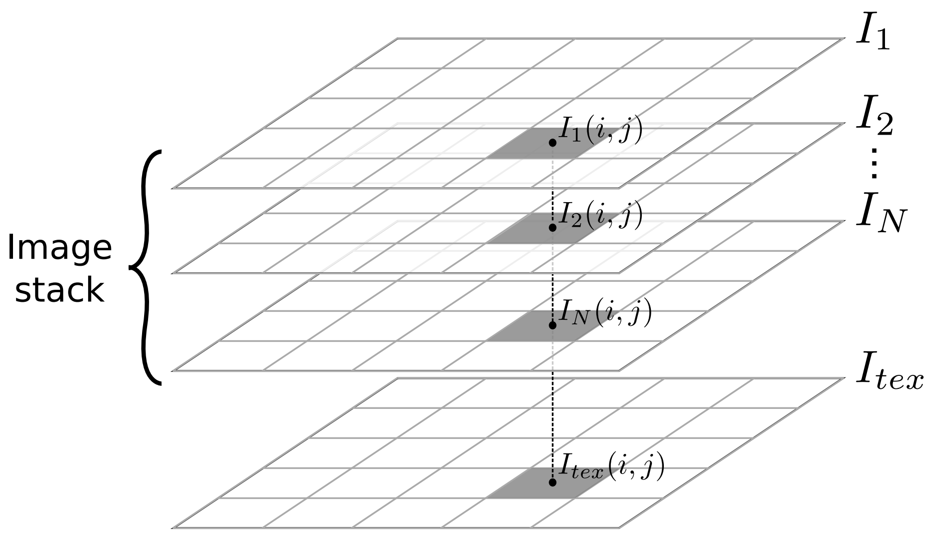

Figure 2. Problem formulation.

where the relation $(p,q) \in \mathcal{N}$ defines that p and q are neighboring pixels in I_{tex}. The term $D(p, f(p))$ is derived from the image stack and expresses the preference of pixel p for label $f(p)$; and typically, V is a non-decreasing potential function.

For each pixel $I_{tex}(i,j) \in I_{tex}$, we define the data term for label α as

$$D(I_{tex}(i,j),\alpha) = ||\alpha - \overline{I(i,j)}||^2. \tag{3}$$

The weighted mean $\overline{I(i,j)}$ is computed as

$$\overline{I(i,j)} = \frac{\sum\limits_{k=1}^{N} I_k(i,j) * w_k(i,j)}{\sum\limits_{k=1}^{N} w_k(i,j)}, \tag{4}$$

where the weights are defined as

$$w_k(i,j) = ||max(A_i(i,j),\ldots,A_N(i,j)) + 1 - A_k(i,j)||^2. \tag{5}$$

The potential function is defined here as $V(p,q,\alpha,\beta) = min(T, ||\alpha - \beta||)$, where T is a positive constant that represents the maximum penalty for a discontinuity. Function V is a piecewise smooth model. This means that small deviations in labels incur only a small penalty, thus the smoothness is encouraged. However sharp jumps in labels are only permitted when penalty T is not too severe to prohibit them [3].

The optimum labeling, i.e., the minimum of function $E(f)$, is found by employing the *extension* algorithm [3] until convergence, and it was implemented using a C++ multi-label optimization library [4].

3. Experimental results

To test the proposed method, we considered a 3D mesh with several photographs of it, in different camera positions and with known parameters. First we generated the atlas

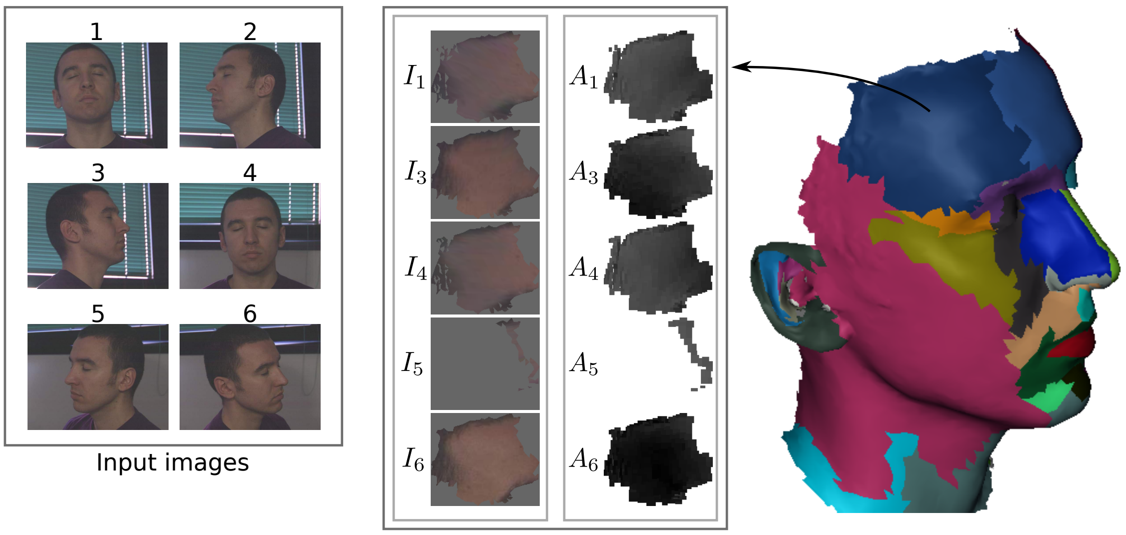

Figure 3. This example consists of 6 input images. They are projected into the patch generating a image stack. Note that image 2 does not cover any region of the patch. In the images representing the angles, darker pixels mean lower angles.

as explained in Subsection 2.1. For each patch, we constructed a stack of correspondent image parts. This result is shown in Figure 3. For each image stack, we applied the graph cut method explained in 2.2.

Figure 4 shows the result for the same example. Note that, because the graph cut method was only applied on the image stacks, there are visible texture transitions between patches, with color discontinuities. This is caused by different illumination conditions during the photographs acquisition. To address this problem, we applied the smoothing algorithm described in [13], which diffuses the color difference between frontier zones. A more related and sophisticated approach would be to apply the same graph cut method in image stacks that represent the transition zone between patches. This will be incorporated in a future study.

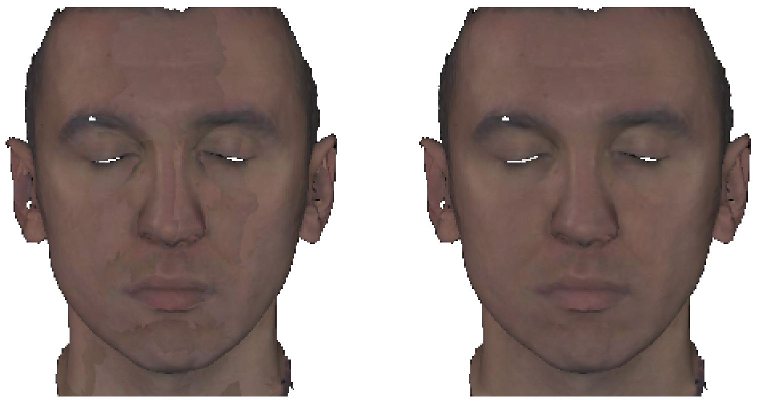

Figure 4. Result of the proposed method before and after the smoothing algorithm.

4. Conclusions

In this paper, we have studied the problem of synthesizing a texture for a 3D model, assuming that we have the correspondent dense triangle mesh of the object and photographs of it in different camera positions and with known camera parameters.

Our methodology to tackle the problem was to first subdivide the mesh surface into patches, which was done by the method described in [13], then generate a stack of correspondent image regions for each patch and, finally, apply a graph cut algorithm to optimally merge the image regions into one coherent texture.

The proposed method is promising because it is simple, and its background theory is widely studied and applied for low-level vision problems that depend on global energy formulations. Besides that, the quality of achieved results are similar to those present in related literature.

As a future study, we want to apply the graph cut algorithm to smooth color discontinuities between patches.

References

[1] ANDALÓ, F., CALAKLI, F., TAUBIN, G., AND GOLDENSTEIN, S. Accurate 3D footwear impression recovery from photographs. In *4th International Conference on Imaging for Crime Detection and Prevention (ICDP '11)* (2011), pp. 1–6.

[2] BAUMBERG, A. Blending images for texturing 3D models. In *Proceedings of the Conference on British Machine Vision Association* (2002), vol. 3, p. 5.

[3] BOYKOV, Y., AND VEKSLER, O. Graph cuts in vision and graphics: Theories and applications. In *Handbook of mathematical models in computer vision*. 2006, pp. 79–96.

[4] BOYKOV, Y., VEKSLER, O., AND ZABIH, R. Fast approximate energy minimization via graph cuts. *IEEE Transactions on Pattern Analysis and Machine Intelligence 23*, 11 (2001), 1222–1239.

[5] CALAKLI, F., AND TAUBIN, G. SSD: Smooth signed distance surface reconstruction. In *Computer Graphics Forum* (2011), vol. 30, pp. 1993–2002.

[6] CALLIERI, M., CIGNONI, P., AND SCOPIGNO, R. Reconstructing textured meshes from multiple range RGB maps. In *7th International Fall Workshop on Vision, Modeling, and Visualization* (2002), pp. 419–426.

[7] COHEN-STEINER, D., ALLIEZ, P., AND DESBRUN, M. Variational shape approximation. In *ACM Transactions on Graphics (TOG)* (2004), vol. 23, pp. 905–914.

[8] FURUKAWA, Y., AND PONCE, J. Accurate, dense, and robust multiview stereopsis. *IEEE Transactions on Pattern Analysis and Machine Intelligence 32*, 8 (2010), 1362–1376.

[9] GAL, R., WEXLER, Y., OFEK, E., HOPPE, H., AND COHEN-OR, D. Seamless montage for texturing models. In *Computer Graphics Forum* (2010), vol. 29, pp. 479–486.

[10] LEMPITSKY, V., AND IVANOV, D. Seamless mosaicing of image-based texture maps. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR '07)* (2007), pp. 1–6.

[11] SNAVELY, N., SEITZ, S., AND SZELISKI, R. Photo tourism: exploring photo collections in 3D. *ACM transactions on graphics (TOG) 25*, 3 (2006), 835–846.

[12] TZUR, Y., AND TAL, A. Photogrammetric texture mapping using casual images. In *Proceedings of ACM SIGGRAPH* (2009), vol. 113.

[13] VELHO, L., AND SOSSAI JR., J. Projective texture atlas construction for 3D photography. *The Visual Computer 23*, 9-11 (2007), 621–629.