Seamless Texturing of Archaeological Data

Michael Birsak Przemyslaw Musialski Murat Arikan Michael Wimmer

Institute of Computer Graphics and Algorithms
Vienna University of Technology, Vienna, Austria
e-mail: {birsak,pm,arikan,wimmer}@cg.tuwien.ac.at

***Abstract*—In this paper we propose a framework for out-of-core real-time rendering of high-quality textured archaeological data-sets. Our input is a triangle mesh and a set of calibrated and registered photographs. Our system performs the actual mapping of the photos to the mesh for high-quality reconstructions, which is a task referred to as the labeling problem. Another problem of such mappings are seams that arise on junctions between triangles that contain information from different photos. These are are approached with blending methods, referred to as leveling. We address both problems and introduce a novel labeling approach based on occlusion detection using depth maps that prevents texturing of parts of the model with images that do not contain the expected region. Moreover, we propose an improved approach for seam-leveling that penalizes too large values and helps to keep the resulting colors in a valid range. For high-performance visualization of the 3D models with a huge amount of textures, we make use of virtual texturing, and present an application that generates the needed texture atlas in significantly less time than existing scripts. Finally, we show how the mentioned components are integrated into a visualization application for digitized archaeological site.**

I. Introduction

Cultural heritage is important to preserve tangible items as well as intangible attributes for future generations. Tangible items (e.g. buildings, monuments) are often too big to be saved in a secure environment like a display case in a museum in order to protect them from atmospheric conditions and natural breakup. Due to the immense computational power of today's hardware, those items can be preserved in a digital manner. For example, laser scanners are used to produce geometric 3D point clouds, often extended with more interesting information, like the surface color, that can be used for projective texture mapping. Often laser scanners allow the connection with a digital single lens reflex camera (*DSLR*) mounted on top in order to acquire color information from the same position.

During the registration process, the position and the orientation of the camera relative to the coordinate system of the laser scanner is registered. Then, provided that the internal parameters of the camera (sensor dimensions, focal length, distortion) are known, an accurate back-projection of the images onto the scanned geometry is possible. For color mapping for high-quality reconstructions of parts of the model, the registered camera is used for taking photos at every scan position in conjunction with scanning of the geometry. The point cloud alone is not suited for mapping of the photos onto the model in a continuous way, thus often the point cloud is converted into a triangle mesh which in turn can be used for piecewise continuous texture mapping. Nonetheless, this process is not trivial, since due to the size or shape of the scanned scene, a single photo is usually insufficient to cover it entirely. So when speaking about meshes, and assuming that every triangle of the mesh should receive its color information from exactly one photo, there have to be edges inside the model where regions, textured by different photos, adjoin to each other. Those edges express themselves as visible artifacts and are also referred to as *seams*.

The whole digitizing process leads to four major problems. The first problem is the finding of an optimal mapping of photos onto the mesh, so that as few seams as possible remain while at the same time choosing a high-quality photo for each triangle. The second major problem is the automatic adjustment in terms of color of the remaining seams. The third problem is the manual editing of the photos, which is often needed to handle remaining artifacts like highlights. Finally, the fourth problem is the high-performance visualization of a 3D model with a huge amount of textures.

II. Related Work

Our work is related to the three different fields of research labeling, leveling and virtual texturing. In the following we want to give an overview of some existing techniques concerning these fields.

Labeling. In the labeling procedures similar to our approach, every face F_i of the model gets a label P^j that corresponds to a particular photo. Assuming that there are K faces F_1 to F_K and N photos P^1 to P^N, a labeling is a mapping from the set of faces $\{F_1, ..., F_K\}$ onto the set of photos $\{P^1, ..., P^N\}$. A label P^i for a face F_j means that the face F_j is textured by the photo P^i.

A simple method to find a valid mapping is also referred to as the best fragment approach [1]. The principle of the best fragment approach is the calculation of weights for every photo-triangle-pair such that every triangle is textured by the photo corresponding to smallest weight. One possibility to calculate a weight for a photo-triangle-pair is to take the sine of the angle between the normal vector of the face and the viewing vector [1]. For better results, a more sophisticated method where the labeling problem is considered as a Markov Random Field (*MRF*) energy optimization [2] was proposed by Lempitsky and Ivanov [1]. For the minimization process they use α-expansion Graph Cuts [3], [4].

Abdelhafiz [5] follows a simpler approach for the labeling. His work is based on the calculation of the areas of the projected faces. Also the approach of Gal et al. [6] is similar the work of Lempitsky and Ivanov [1], but in contrast they use photo-vector-tuples as labels. Musialski et al. [7] presented a method for the generation of high-quality approximated façade

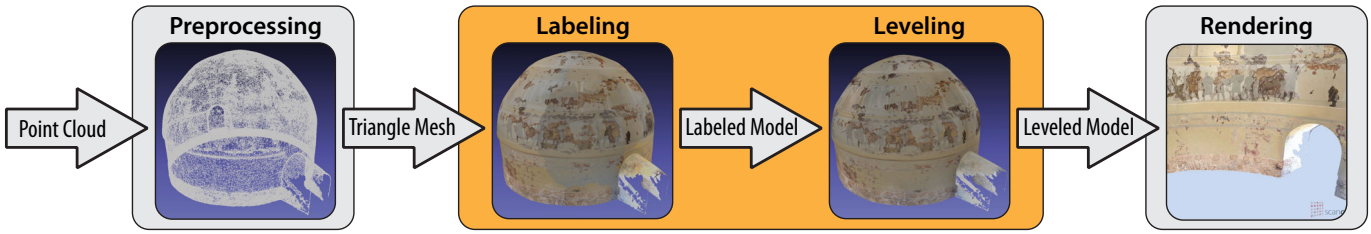


Fig. 1: The pipeline of our method. While our framework handles the entire pipeline, the main contribution of this paper are the two stages in the middle: leveling and labeling.

ortho-textures based on a set of perspective photos, where they use a simpler labeling term. For leveling they utilize a GPU-poisson solver [8]. Pintus et al. [9] directly work on (extremely dense) point clouds. The result is an assignment of colors to the points. The origin of the color information is a set of high-resolution photos. In contrast to our method where every basic element (in our case triangles) gets its color information from a single photo, they assign colors to points, that are weighted sums of several pixels.

Leveling. In the leveling stage, one tries to get rid of the visible artifacts in the model where regions that are textured by different photos adjoin each other. These seams arise because of the different lighting situations during the exposure of the photos.

There is much literature available concerning leveling. However, most of it does not directly deal with leveling on meshes, but only covers approaches used for stitching together two or more planar photos to get a panoramic image. Some proposed approaches [10]–[12], which are also referred to as optimal seam methods, search for a curve in the overlapping region of two photos, and use the curve as the border between the two photos. Other approaches [13] blend between the images under consideration to get a smoother intersection. A better idea than blending or searching for an optimal seam in order to stitch two images together is to keep the variation of the images, namely their gradients, while bringing their intensities onto the same level [8]. Lempitsky and Ivanov [1] adapted the findings of Pérez et al. [8] to the problem of seam leveling on meshes. Instead of solving a Poisson equation, they approximate the solution by solving a least squares problem. A disadvantage of their method is a missing mechanism to keep the calculated color values in the valid range.

Our approach is based on the method of Lempitsky and Ivanov [1]. In contrast, we introduce a further term into the least squares problem that pays attention on the final color values so that they reside in a valid range.

III. Approach

Our approach is a full workflow from the raw data that is gathered in the digitization process of an archaeological monument to a fully textured model free of visual artifacts. In the following we want first to give an overview of our method. Afterwards, we will explain the main parts in detail.

A. Overview

In Figure 1, an outline of our approach is shown. The first step is a preprocessing step where the point cloud of the model is transformed into a manifold triangle mesh. Furthermore, the photographs that were taken from the monument are undistorted using the documented intrinsic parameters of the camera. The goal of the labeling stage is to find a good mapping of the set of faces onto the set of photos, such that the number of seams is minimized while at the same time choosing a high-quality photo for every face. We use an MRF energy optimization approach to do the labeling and introduce occlusion detection using depth maps to prevent usage of image material showing parts of an occluder. In the leveling stage, a piecewise continuous function is calculated for the three color channels R, G and B and added to the input function given by the labeling result. The outcome is then a smooth textured model without discontinuities. The last stage in our approach is the Rendering stage, where we use virtual texturing for high-performance visualization of models containing a great amount of texture data.

B. Preprocessing

Laser scanners usually do not directly deliver triangle meshes but only point clouds. Our system only works on triangle meshes, since we want to calculate a continuous mapping of the photos onto the model. The point cloud has therefore to be transformed into a triangle mesh. The models shown in this paper were either triangulated using the approach of Bolitho et al. [14] (Centcelles cupola) or using the wrap-function of the commercially available tool Geomagic (Hanghaus 2).

Another important preprocessing step is the undistortion of the photographs. Certainly, it would be possible to directly use the original photos and account for the distortion at runtime. However, if the distortion information is just applied to the projected vertices, and the interpolation between them is done linearly, one would only account for the linear distortion. We therefore undistort the photos prior to the labeling stage and then only use the undistorted photos.

C. Labeling

Our approach for labeling takes a manifold triangle mesh and a set of registered photos as input and can be formulated as follows. Let $\{F_1,...,F_K\}$ be the set of faces in a model and let $\{P^1,...,P^N\}$ be the set of registered photos that are used for texturing of the model. The resulting labeling, which is also referred to as a *texture mosaic*, is then defined by a labeling vector $\mathbf{M} = (m_1,m_2,...,m_K)$, where $m_1,m_2,...,m_K \in \{0,1,...,N\}$. An element m_i in $\mathbf{M}$ states that the face F_i is textured by the photo P^{m_i}.

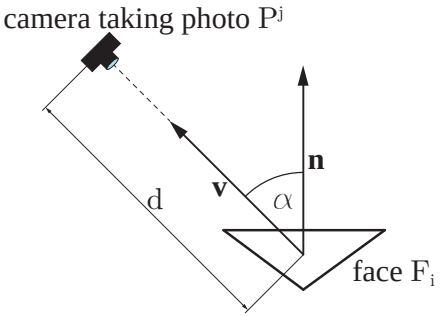


Fig. 2: A face F_i of the model with its normal vector $\mathbf{n}$ and view vector $\mathbf{v}$ to the camera taking the photo P^j. The cost value w_i^j is calculated by $\lambda \cdot \sin^2\alpha + (1-\lambda)\cdot\min(1.0, \frac{d}{d_{max}})$.

Data Cost. The best-fragment approach [1] can be written as $\mathbf{M} = (m_1, m_2, ..., m_K)$, where every $m_i = \arg\min_j w_i^j$. Here, the term w_i^j is the cost to texture the face F_i with the photo P^j. The term w_i^j is also referred to as the *data cost*. We also include the term $\sin^2\alpha$ into our cost value w_i^j, where α is the angle between the view vector and the corresponding face normal, and additionally, we also consider the distance between viewer and face. To keep it simple, we just take a linear combination of these terms to get our final cost value

$$w_i^j = \lambda \cdot \sin^2\alpha + (1-\lambda)\cdot\min(1.0, \frac{d}{d_{max}}), \qquad (1)$$

where $\lambda \in [0,1]$ is a scalar value chosen by the user to emphasize either a perpendicular view or a small distance, and d_{max} is the maximum distance between camera position and surface, such that the weight corresponding to the distance is clamped to 1.0 for all distances bigger than d. In our experiments we usually set $\lambda = 0.5$ and $d_{max} = 10$m. In Figure 2, the exposure scenario is illustrated.

Gal et al. [6] proposed a more complex way to calculate the data cost. Basically, they take the integral over the gradients inside the area the triangle is projected into. Their method incorporates the effect of foreshortening, image resolution, and blur.

Callieri et al. [15] also proposed a method to calculate the data cost for a particular photo-triangle-pair. They compute the quality of image material on a per-pixel basis. For a particular photo, they compute masks using different metrics. These masks are then multiplied in order to get a final mask so that every pixel in the final mask encodes the quality of the corresponding pixel in the photo. For triangle meshes, weights are not needed for the pixels of the photos but for all photo-triangle-pairs. To utilize the masks, the triangles can be projected into the photos, such that for a particular photo-triangle-pair the values in the final mask of the photo at the pixels the triangle is projected onto are considered. Some of the masks that are computed in the work of Callieri et al. [15] are:

- **Border Mask.** Every pixel in the border mask stores the distance of the pixel to both the image borders and discontinuities in the depth map. Higher distances correspond to better image material.
- **Focus Mask.** In the focus mask, the value of each pixel is a measure for the focusing. Higher values correspond to better image material.
- **Stencil Mask.** Often the user wants to exclude portions of the photos which are then not considered in the labeling procedure. The stencil masks are provided by the user and encode the areas that can be used for texturing.

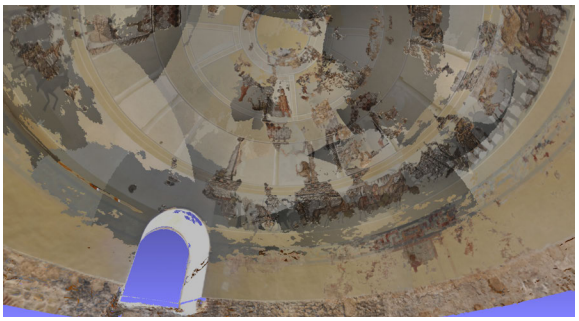


Fig. 3: Result of the labeling using the best fragment approach [1]. Every face is textured by the "best" photo, which leads to many seams.

The methods for calculation of the data cost, which are based on some kind of integral over the projected area [6], [15], are more sophisticated than the simple angle-based approach [1]. Nevertheless, our method is based on the simple approach [1] since it is easier and faster to calculate. However, we introduced the concept of stencil masks into our approach as we will see later.

Smoothness Cost. The best fragment approach [1] does not take the color differences of the photos into account. As a result there are many seams in the final texture as shown in Figure 3.

In order to deal with the seams, we introduce another cost value. Consider two adjacent faces F_i and F_j of the mesh sharing an edge E_{ij}. Given a labeling vector $\mathbf{M}$, the cost produced by this edge is calculated by:

$$w_{i,j}^{m_i,m_j} = \int_{E_{ij}} d(Pr_{m_i}(X), Pr_{m_j}(X))dX \qquad (2)$$

In Equation 2, Pr_i is a projection operator for the photo P^i. The operator $d(.,.)$ returns the distance between two color samples, where we use the Euclidean distance between RGB values. The minimum distance between two color samples is therefore 0.0 if the colors are identical. The maximum distance is $\sqrt{3}$, corresponding to the distance between a white and a black pixel in normalized RGB space. The photos are not continuous functions, so the integral of Equation 2 must be discretized into the sum of distances between color values along the projected edge. Certainly, this sum is 0.0 when the faces F_i and F_j sharing an edge E_{ij} are textured by the same photo. The term $w_{i,j}^{m_i,m_j}$ is also referred to as the *smoothness cost*.

Let $\mathcal{N}$ be the set of adjacent triangle-pairs in a mesh. Then, the final energy term that has to be minimized can be written

as:

$$E(\mathbf{M}) = \sum_{i=1}^{K} w_i^{m_i} + \lambda \sum_{\{F_i,F_j\}\in\mathcal{N}} w_{i,j}^{m_i,m_j} \; . \qquad (3)$$

The value λ in Equation 3 is typically ≥ 0.0. It defines the degree of penalizing of edges shared by faces that are textured by different photos. If 0.0 is chosen, the minimization of Equation 3 degrades to the best fragment approach, and every face is textured by the "best" photo. With increasing λ, the importance of quality of image material used for the faces decreases, since the whole effort goes into the establishment of smooth transitions between areas that receive its color information from different photos. This behavior is illustrated in Figure 4 on the Centcelles cupola model.

(a) $\lambda = 1.0$ (b) $\lambda = 50.0$ (c) $\lambda = 100.0$

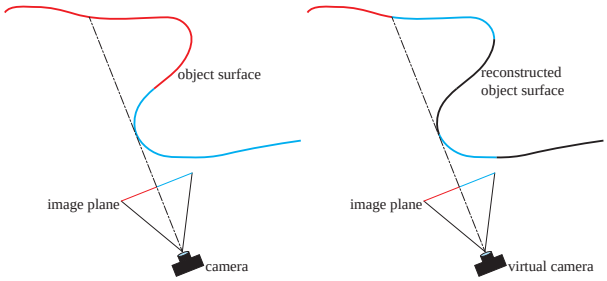
Fig. 4: Impact of the parameter λ on the final texturing result. Note how the quality of image material per face decreases with increasing λ, while the transitions become smoother.

In order to (approximately) solve Equation 3, we use the Graph Cut approach with α-expansion as it was presented by Boykov et. al [3].

Occlusion Detection. The described labeling approach gets into difficulties when complex geometry is considered. The work of Lempitsky and Ivanov [1] just shows results of nearly convex objects like a statue head and a model of the earth ball, but in the field of cultural heritage, however, often far more complex scenes have to be textured. A missing link in the proposed approach [1] is occlusion detection. Without occlusion detection, it is possible that wrong image material is back-projected onto the digitized model. This issue is depicted in Figure 5 in a simplified fashion.

We base our approach for occlusion detection on depth maps as they are also used in methods like shadow mapping. Before the actual labeling procedure begins, a depth map is created for every photo that is used as a label. Although the resolution of the depth maps is independent of the resolution of the photos, we use the resolution of the photos for the corresponding depth maps. A depth map is simply created by rendering the model using the extrinsic and intrinsic parameters of the camera, which are usually documented during the digitization process of a particular monument. We use OpenGL to create depth maps and store the depths in viewing space using a single channel float texture.

Let D^i be the depth map for the photo P^i. In order to decide which of the triangles of the model are visible in P^i, they are all projected into P^i. Not only the texture coordinates the vertices are projected onto, but also their depths in viewing space are evaluated. The depths are compared to the sampled values in the depth map. Only if all values in the depth map are bigger than the distances of the projected vertices, the particular



(a) exposure (b) back-projection

Fig. 5: Incorrectly back-projected image information when no occlusion detection is carried out. The taken image (a) is back-projected onto surface areas that to do not correspond to the image information (b).

triangle is considered as visible in P^i, otherwise it is considered as occluded. To avoid artifacts, we use an empirically evaluated threshold of 0.1 ($= 10cm$) when comparing the values.

Stencil Masks. Similar to the proposed stencil masks [15], we allow the user of our system to provide masks for the photos. These masks are basically black-and-white images providing information about the image material that can be used in the labeling procedure. Such stencil masks are needed since the photos often contain regions which are out of focus or objects that are not part of the model (e.g. persons).

During the labeling procedure, the triangles are projected into the photos. When a stencil mask is provided for a particular photo, the area the current triangle is projected onto is analyzed in the stencil mask. When there is a black pixel in the area corresponding to "masked out", the triangle is considered as "unlabable" by the photo. Only when the entire region consists of white pixels, the photo can be used as origin of texture material for the triangle. In Figure 6, a photo of the Centcelles cupola containing parts of the crane that was used in the digitization process and the corresponding stencil mask, in which the crane is masked out, is shown.

(a) photo (b) stencil mask

Fig. 6: Photo of the Centcelles cupola containing image material that should not be used in the labeling procedure. Corresponding mask. White regions are shown in grey for visibility reasons.

Discussion. The output of our approach is a labeled triangle mesh. The labeling information consists of the mapping of triangles onto the set of photos. Furthermore, the texture coordinates for the projected triangles are provided. In particular, we generate an OBJ model file in conjunction with a material file.

In practice there remain seams in the model where regions that get color information from different photos adjoin. In Figure 7, those seams are shown for the Centcelles cupola model that was textured using our application. As already mentioned, these seams arise because of the different lighting situations during the exposure of the photos. For a high-quality model, those seams have to be handled. Because an entirely manual editing is unacceptable, we will show our automatic approach for leveling in the next section.

D. Leveling

The result of the labeling procedure described in the previous section is a labeling vector $\mathbf{M}$ that defines a mapping of the set of triangles of the model onto the set of photos used for texturing. We now have a set of connected components $\{C_1,...,C_T\}$, where each component consists of a set of connected triangles that receive their color information from one particular photo. Assuming that the photos are continuous functions for each color channel (R, G and B), the mapping of just one color channel of the photos onto the 3D model with respect to the labeling vector $\mathbf{M}$ results in a piecewise continuous function f on the mesh M. Only at the edges where two different components C_i and C_j adjoin, there are points of discontinuity. What we are looking for in the leveling procedure is a piecewise smooth leveling function g that meets the two following criteria: (1) The magnitude of the gradients of the leveling function g is minimal. (2) The jumps of the leveling function g are equal to the negative jumps of f.

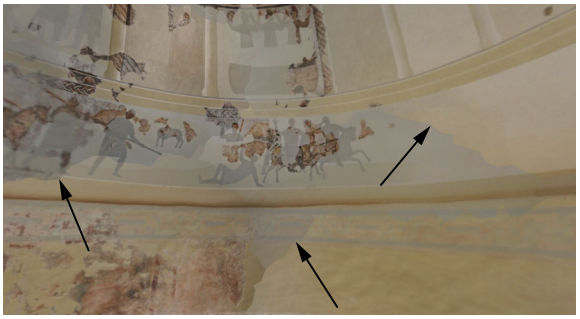


Fig. 7: Remaining seams in the model that was textured using our labeling approach.

The first one is important to preserve the high frequencies of the function f. The second one guarantees that the points of discontinuity are smoothed out at the edges where two connected components C_i and C_j adjoin. For leveling using photos with three color channels (R, G and B) each channel has to be calculated separately. In the following, we will describe the leveling only for a single channel (cf. Fig. 8).

1) Leveling method: Our approach for the leveling process is based on the method of Lempitsky and Ivanov [1]. They calculate the leveling function g only at the vertices of the mesh and then interpolate the function values along the edges and triangle surfaces. For the explanation of their approach, they consider the set $\mathcal{M}$ containing all the (i,j)-pairs prescribing that at least one triangle adjacent to the vertex V_i is part of the connected component C_j. For each of these (i,j)-pairs, the corresponding leveling function value g_i^j is computed. As an example, consider a vertex V_1 that is part of three connected components C_1, C_2 and C_3. By definition, the connected components are textured by different photos. When the vertex V_1 is projected into these photos, the intensity values of the pixels V_1 is projected into, may differ. These intensity values define the original texture function f. At V_1 we have therefore three different function values. We define f_i^j to be the original texture function value at the vertex V_i for the connected component C_j. In our example we have therefore the function values f_1^1, f_1^2 and f_1^3 at the vertex V_1. The differences between these function values then lead to the point of discontinuity at V_1. So three leveling function values g_1^1, g_1^2 and g_1^3 have to be calculated at the vertex V_1. These different leveling function values are necessary to smooth out the discontinuities at V_1.

We denote $\mathcal{L}$ to be the set of (i,j)-pairs prescribing an edge E_{ij} formed by the vertices V_i and V_j. Now the leveling function g, computed at the vertices of the mesh, can be approximated by the minimization of the following least-squares energy function:

$$E = \sum_{i_1,i_2,j} \left(g_{i_1}^j - g_{i_2}^j\right)^2 + \lambda \sum_{i,j_1,j_2} \left(g_i^{j_1} - g_i^{j_2} - (f_i^{j_2} - f_i^{j_1})\right)^2 . \tag{4}$$

The first term of 4 approximates the first condition we demanded of the leveling function g. This condition is the minimality of the magnitude of the gradients. The first term corresponds to all the edges $E_{i_1 i_2}$ in the mesh whose vertices V_{i_1} and V_{i_2} are part of the same connected component C_j. The second term of 4 approximates the second demanded condition of the leveling function g. To ensure smooth transitions at the points of discontinuity, the jumps of g need to be the negative jumps of f. The second term corresponds to all the vertices V_i in the mesh, where two connected components C_{j_1} and C_{j_2} adjoin. In practice, we use a large λ (e.g. 100) in order to give more importance to the second term. In our implementation we solve the system with the sparse CG solver of the IML++ library [16].

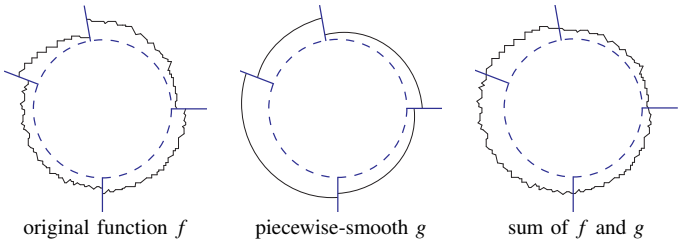


Fig. 8: Leveling procedure: discontinuities are smoothed out while at the same time high frequencies are preserved (Figure based on [1]).

With the proposed method, an adequate leveling function can be calculated up to an additive constant. This additive constant can be set to an appropriate value to maintain the mean gray value of the current color channel. One problem that we observed with the approach of Lempitsky and Ivanov [1] is the divergence of the leveled color values: when the procedure leads to a range for the color values of e.g. $[-0.37, 1.67]$, there is no additive constant that would avoid clamping. In order to resolve this issue we introduce an additional penalty term to the equation.

2) Penalized leveling method : Because of the shown issues, a strategy is necessary to avoid the divergence of the leveled color values. We decided to adapt the least-squares problem shown in Equation 4. A disadvantage of this least-squares problem is the absence of a term that penalizes big leveling function values. It only places importance on small differences between adjacent g_i^j and small color differences between adjoining connected components C_i.

Therefore, we introduce a new term into the least-squares problem such that big leveling function values are penalized:

$$E = \sum_{i_1,i_2,j} \left(g_{i_1}^j - g_{i_2}^j \right)^2 + \lambda \sum_{i,j_1,j_2} \left(g_i^{j_1} - g_i^{j_2} - (f_i^{j_2} - f_i^{j_1}) \right)^2 + \mu \sum_{i,j} \left(g_i^j \right)^2 . \quad (5)$$

This term can not fully avoid that the leveled color values exceed the valid range, but we observed a much better behavior. Our adapted least-squares problem is shown in Equation 5.

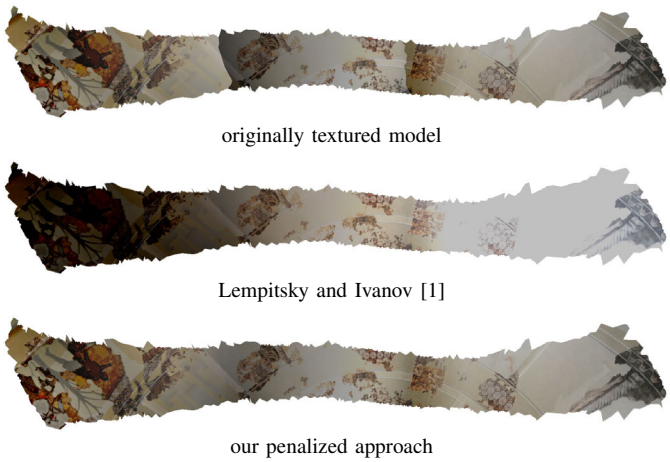


Fig. 9: Comparison of the results of different leveling approaches (cf. Section III-D2).

We tested the new term extensively in our experiments and observed that the least-squares system reacts very sensitive to the parameter μ that controls the contribution of the penalty term. When a value of 100.0 was chosen for the parameter λ, a value of 0.01 for μ already brought considerable results. When the value for μ was chosen too high, the whole leveling procedure completely failed, and all the g_i^j got a value of 0.0. Nonetheless, when choosing an appropriate μ carefully, the method delivers significantly better results than the original approach [1].

Figure 9 shows a comparison on a mesh textured by three different photos, where the intensity values in each photo increase from left to right. Note the big loss of contrast on the left and right side of the model when there is no penalizing of big values (Figure 9b). In contrast, in our version the seams are smoothed out while at the same time the contrast of the overall texture is not lost (Figure 9c).

The reason for the divergence of the color values, as it is shown in Figure 9b, is the original texture function f. We observed a diverging behavior when f had nearly the structure of a sawtooth function (see Figure 9a) so that the image intensity increases along each of the adjoining connected components C_i.

E. Rendering

Although some of the input textures can be filtered out in the labeling step, since they are not needed to completely cover the model, a labeled and leveled data set of an archaeological monument often contains a big amount of texture data. Using a standard approach used by most 3D model viewers (e.g. Meshlab [17]) leads to bad performance and cluttered movement through 3D space. This performance drop is caused by continuously streaming of texture data that does not fit into the GPU's memory. However, visualization of the model at acceptable frame rates is crucial for any post processing steps that are done manually after the leveling stage. Such manual texture adaptation is usually done by a graphic artist in applications like Adobe Photoshop.

Therefore, we introduce virtual texturing into the workflow. With virtual texturing, only the needed texture parts used for rendering of the current frame are streamed to the graphics card. We use the LibVT virtual texturing library [18], which has been integrated into our visualization application that is primarily used for visualization of out-of-core point-based and polygonal data sets. This method uses an atlas which is a single texture that consists of all the textures that belong to the virtually textured models in the scene. Texture coordinates of the virtually textured models have to be altered to reference the correct positions inside the atlas. The basic element of the tile store is a so-called *tile*. These tiles are the atomic elements that are streamed to the graphics card.

IV. Results and Applications

Our whole pipeline is implemented in C++. We have used the OpenCV library in the version 2.4.2 for all image operations. Further we make use of the gco-v3.0 graph cuts library [3] for the labeling stage, and of the SparseLib++ library in conjunction with IML++ [16] for the leveling stage.

In Figure 10 the results after labeling (left side) and leveling (right side) are shown for two models. While after the labeling stage there are still visible seams in the model, there are no seams in the final result after the leveling stage. The upper model is the Centcelles cupola in Catalonia which consists of 1 million triangles and 70 input photos. The labeling took about 20 minutes, the leveling 8 minutes. The lower model is the "Hanghaus 2" in Ephesos which consists of 1.2 million triangles and 90 input photos. The labeling took about 34 minutes, the leveling 12 minutes. All tests have been done on a PC with an Intel i7 920 CPU with 2.66 GHz, 6 GB of RAM and an nVidia 580GTX GPU.

In Figure 11, we show the performance of our implementation of the atlas and tile store generation. As can be seen, our implementation is at least 3.6 times faster regarding the atlas, and about 10 times faster regarding the tile store generation. This can easily explained, since LibVT [18] calls a command line based tool (Image Magick) for every single image that is generated. In contrast, our C++ implementation holds as much image data as possible in memory to achieve good performance.

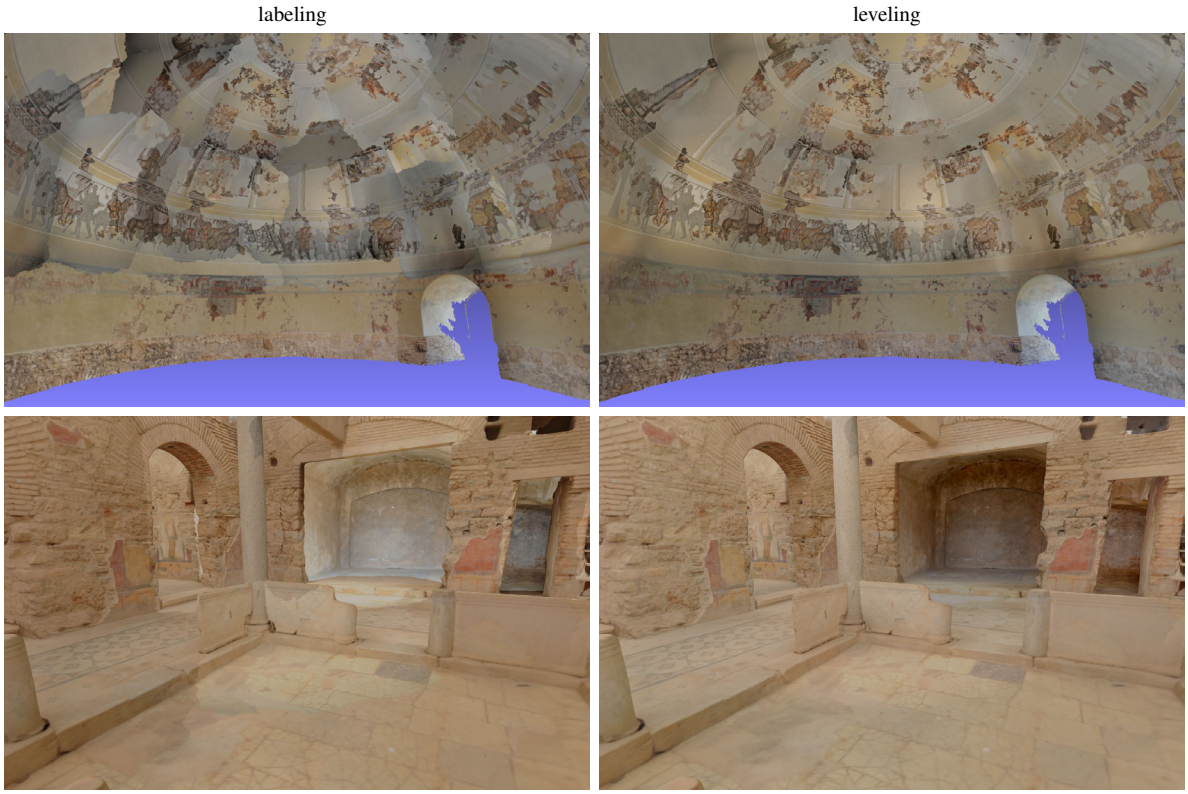


Fig. 10: Final results after labeling and leveling using our method. Top row shows the cupola in Catalonia, bottom row the Hanghaus 2 in Ephesos.

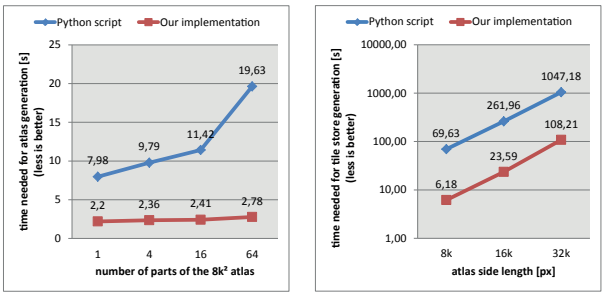


Fig. 11: Running-time of LibVT scripts [18] vs. ours.

In Figure 12, we compare our approach to the method of Lempitsky and Ivanov [1]. Note that there is incorrectly mapped image material in their results. In contrast, our approach delivers correct labeling results by the use of stencil masks and occlusion detection using depth maps.

V. Conclusions

We presented a full workflow for the post-processing of a digitized archaeological item. The input of our approach is a triangle mesh of the model and a set of registered photos. We showed that we could further improve the results of a proposed labeling method. We introduced an occlusion detection approach using depth maps to prevent texturing of surface areas with image material that does not contain the corresponding image information. For the automatic adjustment in terms of colors of the photos used for texturing of the model, we showed how we could improve a proposed method used for leveling. We introduced a new term into a least squares problem in order to prevent the leveled colors to leave the valid range.

There are some situations, where our leveling method can fail. Since the leveling function is only calculated at the vertices of the model and linearly interpolated in between, this can lead to significantly color differences of adjoining triangles that receive its color information from different photos, also when the vertexes are perfectly leveled. A solution would be an adaptive refinement of the mesh at the borders of regions that are textured by different photos.

An interesting way of labeling was presented by Dellepiane et al. [19], who calculate the optical flow for surface regions where projected photos overlap in order to warp these photos together. A global adaption of the camera registrations, so that no visible misalignments in the model are the result, is either very difficult to calculate or even impossible. Therefore, we think that research in the fields of local photo adaption as it is done in the mentioned paper [19] in order to compensate for camera registration errors is the right way to go.

Acknowledgements

We want to thank Norbert Zimmermann for providing the models and Claus Scheiblauer for his assistance concerning the framework. This work has been partially supported by projects FFG 825842 - FIT-IT and FWF P23237-N23.

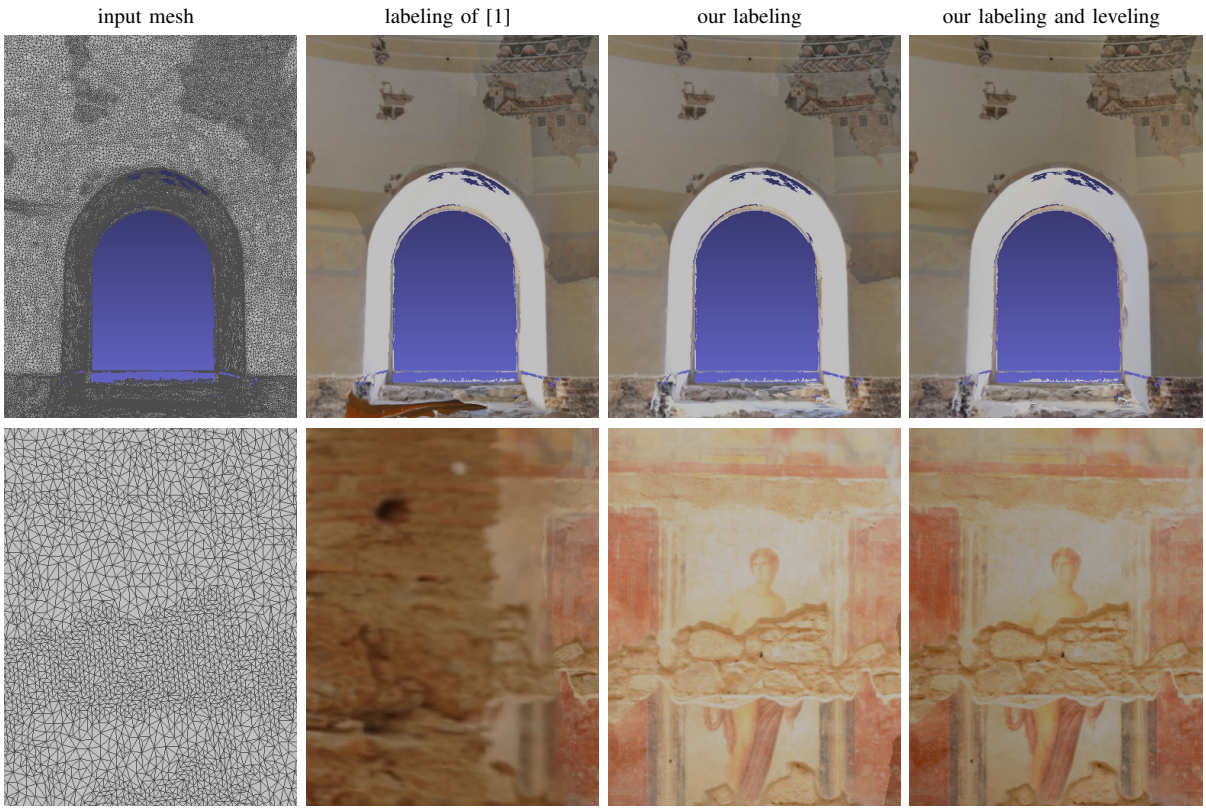


Fig. 12: Top row: Centcelles cupola in Catalonia, bottom row: Hanghaus 2 in Ephesos. Column 2 shows labeling results of Lempitsky and Ivanov [1]. Note the incorrectly mapped image material in both cases. Column 3 shows our labeling result that avoids mapping of incorrect image material. Last column shows our final result after labeling and leveling.

REFERENCES

[1] V. Lempitsky and D. Ivanov, "Seamless mosaicing of image-based texture maps," in *Computer Vision and Pattern Recognition, 2007. CVPR '07. IEEE Conference on*, june 2007, pp. 1–6.

[2] S. Z. Li, *Markov random field modeling in computer vision*. London, UK: Springer-Verlag, 1995.

[3] Y. Boykov, O. Veksler, and R. Zabih, "Fast approximate energy minimization via graph cuts," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 23, no. 11, pp. 1222 –1239, nov 2001.

[4] V. Kolmogorov and R. Zabih, "What energy functions can be minimized via graph cuts?" *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 26, no. 2, pp. 147–59, Feb. 2004.

[5] A. Abdelhafiz, "Integrating digital photogrammetry and terrestrial laser scanning," Ph.D. dissertation, Technical University Braunschweig, 2009.

[6] R. Gal, Y. Wexler, E. Ofek, H. Hoppe, and D. Cohen-Or, "Seamless montage for texturing models." *Comput. Graph. Forum*, pp. 479–486, 2010.

[7] P. Musialski, C. Luksch, M. Schwärzler, M. Buchetics, S. Maierhofer, and W. Purgathofer, "Interactive multi-view façade image editing," in *Vision, Modeling and Visualization Workshop 2010*, Nov. 2010.

[8] P. Pérez, M. Gangnet, and A. Blake, "Poisson image editing," in *ACM SIGGRAPH 2003 Papers*, ser. SIGGRAPH '03. New York, NY, USA: ACM, 2003, pp. 313–318.

[9] R. Pintus, E. Gobbetti, and M. Callieri, "A streaming framework for seamless detailed photo blending on massive point clouds," in *Proceedings of Eurographics Conference - Cultural Heritage Papers*, ser. 32nd Eurographics Conference, Llandudno, Wales, UK, Eurographics. Wiley-Blackwell, April 2011.

[10] D. L. Milgram, "Computer methods for creating photomosaics," *IEEE Trans. Computers*, vol. 24, no. 11, pp. 1113–1119, 1975.

[11] A. A. Efros and W. T. Freeman, "Image Quilting for Texture Synthesis and Transfer," in *SIGGRAPH 2001, Computer Graphics Proceedings*, E. Fiume, Ed. ACM Press / ACM SIGGRAPH, 2001, pp. 341–346.

[12] J. Davis, "Mosaics of scenes with moving objects," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, ser. CVPR '98. Washington, DC, USA: IEEE Computer Society, 1998, pp. 354–.

[13] M. Uyttendaele, A. Eden, and R. Szeliski, "Eliminating ghosting and exposure artifacts in image mosaics." in *CVPR (2)*. IEEE Computer Society, 2001, pp. 509–516.

[14] M. Bolitho, M. Kazhdan, R. Burns, and H. Hoppe, "Multilevel streaming for out-of-core surface reconstruction," in *Proceedings of the fifth Eurographics symposium on Geometry processing*, ser. SGP '07. Aire-la-Ville, Switzerland, Switzerland: Eurographics Association, 2007, pp. 69–78.

[15] M. Callieri, P. Cignoni, M. Corsini, and R. Scopigno, "Masked photo blending: Mapping dense photographic data set on high-resolution sampled 3d models." *Computers & Graphics*, vol. 32, no. 3, pp. 464–473, 2008.

[16] J. Dongarra, A. Lumsdaine, X. Niu, R. Pozo, and K. Remington, "Sparse matrix libraries in c++ for high performance architectures," 1994.

[17] P. Cignoni, M. Corsini, and G. Ranzuglia, *MeshLab: an open-source 3D mesh processing system*, Apr. 2008.

[18] A. J. Mayer, "Virtual texturing," Master's thesis, Institute of Computer Graphics and Algorithms, Vienna University of Technology, Favoritenstrasse 9-11/186, A-1040 Vienna, Austria, Oct. 2010.

[19] 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. 18, pp. 463–474, 2012.