EUROGRAPHICS 2021 / N. Mitra and I. Viola
(Guest Editors)

Volume 40 (2021), Number 2

Texture Defragmentation for Photo-Reconstructed 3D Models

Andrea Maggiordomo[1], Paolo Cignoni[2], and Marco Tarini[1]

[1]University of Milan, Italy
[2]ISTI – CNR, Italy

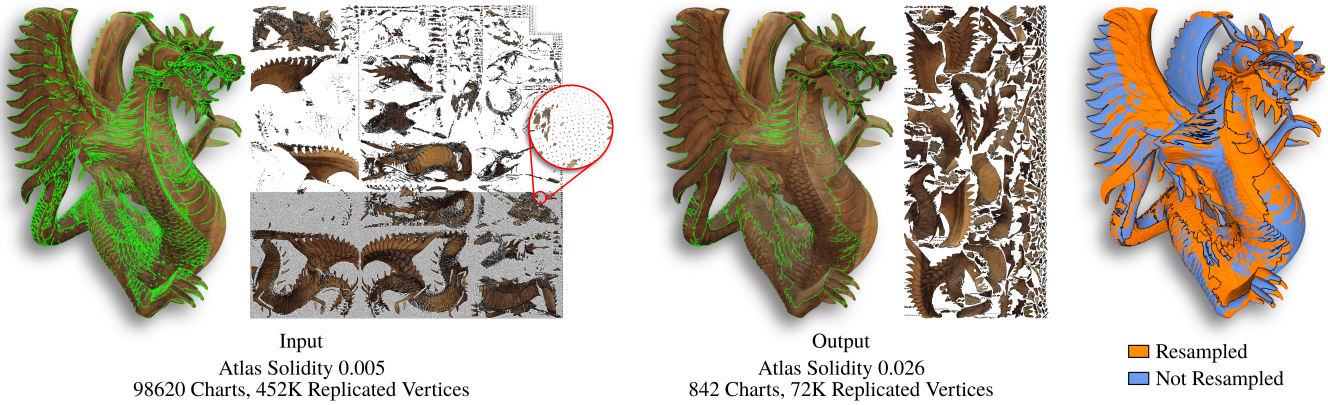


Figure 1: Our algorithm takes as input a textured 3D model (left), and iteratively merges texture charts to reduce the UV-map fragmentation (middle), leveraging the existing parametrization to minimize and possibly avoid resampling artifacts in the output map.

Abstract

We propose a method to improve an existing parametrization (UV-map layout) of a textured 3D model, targeted explicitly at alleviating typical defects afflicting models generated with automatic photo-reconstruction tools from real-world objects. This class of 3D data is becoming increasingly important thanks to the growing popularity of reliable, ready-to-use photogrammetry software packages. The resulting textured models are richly detailed, but their underlying parametrization typically falls short of many practical requirements, particularly exhibiting excessive fragmentation and consequent problems. Producing a completely new UV-map, with standard parametrization techniques, and then resampling a new texture image, is often neither practical nor desirable for at least two reasons: first, these models have characteristics (such as inconsistencies, high resolution) that make them unfit for automatic or manual parametrization; second, the required resampling leads to unnecessary signal degradation because this process is unaware of the original texel densities. In contrast, our method improves the existing UV-map instead of replacing it, balancing the reduction of the map fragmentation with signal degradation due to resampling, while also avoiding oversampling of the original signal. The proposed approach is fully automatic and extensively tested on a large benchmark of photo-reconstructed models; quantitative evaluation evidences a drastic and consistent improvement of the mappings.

CCS Concepts

• ***Computing methodologies*** → ***Texturing;*** *Mesh models;*

1. Introduction

We present a technique that improves the existing parametrization (UV-map layout) of a given textured 3D model, by reducing the amount of texture seams and texture charts while at the same time striving to maintain the original parametrization as much as possible, and avoid resampling the original texture images.

The objective of starting with an existing UV-map and exploiting some of the information it brings places our method in a different class, outside of the well studied, although still open, problem of *ex-novo* construction of a parametrization for a given 3D model. Before we proceed to present the details of our solution (Sec. 2), we illustrate the context that motivates this objective and the intended scope of our technique.

1.1. Context and motivations

Techniques for reconstructing a 3D model from a set of photographs, such as Photogrammetry or Structure-From-Motion [HZ04, SCD*06, RSN*14] are now popular among the general public, with well-understood groundbreaking applications in many different areas such as cultural heritage (for digital documentation of artifacts and sites), the entertainment industry (assets production), electronic commerce (virtual product presentation), digital art, leisure, and others. A variety of commercial and free software packages are now available, only requiring basic photography skills and minimal technical background from users to produce faithful-looking textured 3D reconstructions.

These 3D models are becoming increasingly common and come with a distinctive set of technical characteristics. The geometric resolution, texture resolution, and geometric accuracy are, generally, fully adequate for high-quality renderings; the textures in particular — directly stemming from the original photographs — contribute greatly to the photo-realism of the renderings.

In stark contrast, the underlying UV-map is typically very low quality: in this class of meshes, it is produced as an integral part of the automatic reconstruction process. Often, the UV-map fragments the surface into an excessive number of "texture islands", many of them small, others larger but presenting long, twisted boundaries; overall, the models present an extremely large number of *texture seams* (also called texture cuts), as empirically evidenced by a recent study [MPCT20].

While texture seams are unavoidable in any textured mesh, their presence imposes a small toll on performance, memory resources, and quality of a rendering [YLT19] as summarized below. In normal meshes, the effect is usually negligible for most applications; in typical photo-reconstructed models, however, the disproportionate number of texture seams can result in significant strain on downstream applications, which can detract from usability, especially in resource critical contexts such as web-applications or video-games.

This situation is not accidental, and should not be attributed merely to an oversight or lack of engineering of current photo-reconstruction tools. Rather, it results from peculiar requirements and constraints of the UV-map construction inside the photo-reconstruction pipeline compared to generic authored 3D models.

UV-map construction for generic models When a generic mesh is to be textured, a UV-map needs be constructed starting from its "naked" geometry, for example by appropriately cutting the surface into disk-like regions, unfolding each of them into planar charts, and packing these charts on the texture rectangle. During this process, a blend of objectives is pursued, including limiting the geometric distortions (undistorted mappings are desirable because they imply an even texel distribution on the surface), as well as limiting the amount of cuts. Only after the UV-map is ready, the final texture is filled accordingly, by generating color values for each texel (for example, painting them, or extracting them from a high-resolution model [TCS03]).

With typical photogrammetry models, the above procedure is neither desirable, nor practical. First, in this case the final color information comes entirely from the photographs themselves; this means that an undistorted (area and angle preserving) mapping would not in any way be beneficial in this scenario: regardless of how well texel samples are distributed on the surface, any color information they contain will have to be extracted from the available samples, i.e. the pixels on the original images. Ideally, the texture samples should precisely match the input pixels. A related consideration is that rendering a textured mesh resamples the texture signal when the object is presented to the screen. If the texture data was already resampled from the original photographs then the original signal would be resampled twice, deteriorating it further. On top of this, an additional technical difficulty is that the mesh is typically high-resolution, extremely irregular, and affected by inconsistencies such as small holes, topological noise, or lack of manifoldness. These factors hinder both automatic algorithms and artist-assisted tools for UV-map creation. This is evidenced by the empirical experiment in [MPCT20], which indicates that UV-creation tools, from either commercial software suites or research labs, fail to produce a satisfactory parametrization on more than 50% of photogrammetry models.

For these reasons, the UV-map creation phase of a photogrammetry 3D reconstruction follows a different approach, as follows.

UV-map construction for photogrammetry models In photo-reconstructed models, each part of the 3D surface is visible, by definition, from a portion of one or more photographs. Therefore, a natural mapping is already defined from each portion of the 3D surface into the 2D photo, defined by the view projection. The final texture is simply a collage of portions of the original photographs. The resulting UV-map features a local sampling density that reflects the actual availability of color data. For example, it assigns more texels to areas captured from close-up shots then to areas seen from afar. It also avoids any undue oversampling, or indeed any resampling of the original photographs.

As a bonus, this approach is extremely reliable, scalable, and fully automatic. The difficult task of creating a global parametrization is substituted by the much simpler task of just identifying the best photograph for each triangle, then clustering triangles according to the assignment, and finally packing the resulting regions in a final texture. This scales very well with the complexity of the mesh, and does not require any assumption on its consistency (e.g. manifoldness), although care must be taken to hide visible seams and ghosting artifacts emerging from small inconsistencies in the global registration of the available images [LI07, GWO*10].

The drawback is that, as noted, the mapping will present excessive fragmentation, because models are usually reconstructed from many photographs form different angles, and the assignment of triangles to photographs depends on occlusion and face normals, both of which vary on a per-triangle basis (especially for complex or noisy geometries).

Problems induced by texture fragmentation The penalty imposed by the excessive number of texture seams affecting typical photogrammetry models can be summarized as follows (see [YLT19] for a more detailed analysis):

- *Unused texture space*. The gaps unavoidably left by the imperfect packing of charts represent a waste of texture memory; in

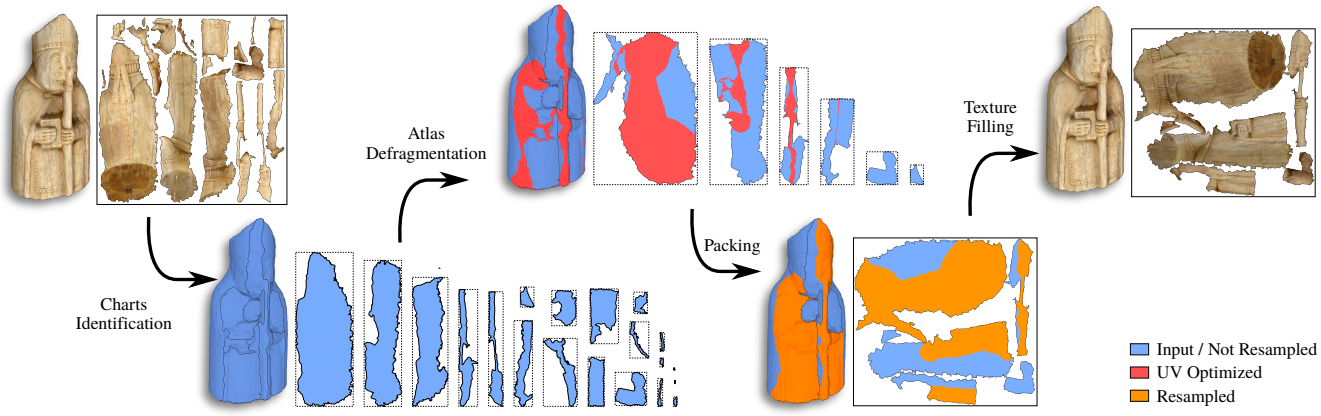


Figure 2: Overview of our method. We begin by extracting the charts from an input textured model, then iteratively merge them by optimizing a small area in neighborhood of the fused boundaries (in red). Then, we detect and repack the texture atlas, striving to limit the necessary resampling in the following texture filling phase (orange areas are resampled, blue areas are grid-preserving [RNLL10] and copied verbatim from the original).

many photogrammetry models, this accounts for a significant percentage of the total texture memory on the GPU;

- *Replicated texels.* To minimize the visual artifacts at seams due to bilinear interpolation, color values must be replicated on the two sides of the texture in order to avoid the visual artifact known as "texture bleeding". This means that the number of *unique* texture samples can be much smaller than the number of stored texels;
- *MIP-mapping incompatibility.* The above problem is exacerbated by MIP-mapping, which requires a much wider gap around charts to avoid texture bleeding when minification filtering is used;
- *Incompressible residual visual artifacts.* Irrespective of any amount of texel replication, the seam will always be visible under extreme magnification, because of the inconsistent pattern of bilinear interpolation on the two sides of the cut (see [RNLL10]);
- *Vertex duplication at seams.* To represent texture seams, the seam vertices must be duplicated in the buffered mesh geometry, directly impacting the GPU memory usage, the per-vertex rendering workload, and also vertex-cache efficiency on the GPU;
- *Diminished texture-cache coherency.* Texture seams cause the pattern of access to texture to be less cache-coherent, affecting the efficiency of the hard-wired texture look-ups.

The magnitude of these drawbacks grows with the number of seams; therefore, by substantially alleviating the fragmentation, we expect to reduce each of them significantly. In Sec. 5, we offer an empirical assessment of the improvements obtained by our method, as measured on a large benchmark of real-world examples.

2. Overview

We want to improve an existing texture UV-map by pursuing two conflicting goals:

1. Mitigate its fragmentation, i.e. reduce the number of seams;
2. Preserve the original distortion properties, ideally avoiding altogether the need to resample the original texture in as many areas as possible.

Algorithm 1

```
1:  procedure GREEDY TEXTURE DEFRAGMENTATION
2:      Identify all candidate merge operations          ▷ Sec. 4.1
3:      Score each operation                             ▷ Sec. 4.3
4:      Insert operations in priority queue Q
5:      repeat
6:          Extract top operation S from Q
7:          Perform S                                    ▷ Sec. 4.2
8:          if S failed then
9:              Roll back S
10:             Insert S in failed operation set F
11:         else                                         ▷ Sec. 4.4
12:             Update score of impacted operations in Q
13:             Move impacted op. from F to Q
14:             Update sorting of Q
15:         end if
16:     until Q is empty
17:     Pack the final atlas and synthesize a new texture ▷ Sec. 4.5
18: end procedure
```

In order to remove a given texture seam, the two sides of the cut must be aligned in texture space, which in the general case requires deforming their shape, thus affecting the distortion of the UV-map; moreover, the change propagates to the affected triangles and, in cascade, to the neighboring triangles. This means that our two goals are conflicting, and our algorithm strives to find a balance between them.

This problem statement leads to a combinatorial explosion of the number of potential solutions that can be considered; finding the global optimum would clearly be impractical. Instead, we resort to a greedy approach which works by iteratively performing a

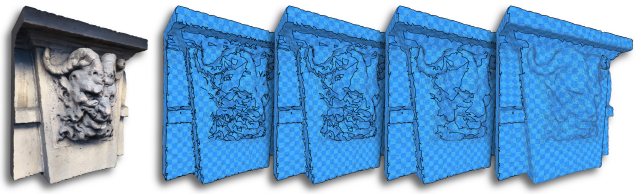


Figure 3: Our method produces a simplified cut layout for the existing UV-map by progressively dissolving texture seams.

sequence of local *merge operations* (summarized in Figure 2 and Algorithm 1).

Each merge operation consists in attempting to fuse two neighboring texture charts into one larger chart, dissolving all seams originally separating them (thus helping towards Goal 1). Due to the need to reduce the explosion of the number potential fusion operations, we disregard partial fusion operations where only a part of the seams are fused. Dissolving a seam in general implies the modification of the UV positions of the involved vertices, which introduces distortions and, potentially, new overlaps. To counter this, we re-optimize the UV-map in a *limited area* around the modified UV positions (Sec. 4.2). This optimization step resembles the distortion minimization procedure of standard parametrization approaches, with the difference that we target the shape of the original UV triangles rather than the shape of the 3D mesh-triangles (according to Goal 2 above). If, after the optimization, this merge operation results in overlaps or excessive distortions it is discarded and rolled back.

As in any greedy approach, the order of the attempted operations is crucial, as different orderings can lead to different results. All potential operations are identified and scored according to their predicted outcome towards our two goals, which is estimated with a heuristic (Sec. 4.3). Then, they are attempted from most appealing to least appealing, progressively removing the existing seams as shown in Figure 3. A successful operation affects the scores of other potential operations, which are updated accordingly (a priority queue is used to quickly identify the next operation to attempt).

Once all merge operations have been attempted, we pack the updated charts in a new texture area. When placing each chart, we can detect triangles that have been only rigidly transformed but *not* deformed, and constrain the chart placement to be *grid-preserving* [RNLL10] w.r.t. the input position of such triangles. In doing so, we can avoid resampling a possibly significant fraction of the input signal when generating the new texture data.

Lastly, (Sec. 4.5) we synthesize a new texture image for the updated UV-map (which is usually much smaller than the original image), by resampling color values only where necessary, and copying pixels verbatim in the other parts.

Before detailing each phase of this algorithm we review relevant previous literature.

3. Related Work

Single-patch Mesh Parametrization There is a vast literature on mesh parametrization, with applications in texture mapping among the most common motivations. A thorough discussion of this topic is beyond the scope of this work, and in the following we discuss only the more closely related issues; we refer the reader to [FH05, SPR*07, HPS08] for an introduction.

Single patch mesh parametrization seeks the best UV image of a given disk-like discretized surface, and is typically formulated as an energy minimization problem. The energy functional measures the amount of distortion, defined as the discrepancy in shape (angles), area, or both (isometric distortion, as in our case), between the 3D mesh faces and corresponding texture triangles. The distortion is minimized over the possible assignments of UV positions to mesh vertices using numerical methods.

Several approaches have been proposed, differing for example on whether the UV boundary needs to be known a priori [EDD*95, Flo97], whether only shape distortion is minimized [DMA02, LPRM02, SC17], and whether local self-overlaps of charts (also known as fold-overs) are prevented [Tut63]. In some techniques, global self-overlaps of charts, i.e. self-intersecting chart boundaries, are prevented [ZMT05, SS15, JSP17] or automatically fixed [LPRM02, LVS18]. The complexity of the numerical methods employed also varies considerably, ranging form simple least-squares solvers (only the case for conformal methods), to iterative descent algorithms from non-linear optimization theory. Early approaches [HG00, SSGH01] move a single vertex at a time, and exploit multi-resolution structures to accelerate the process. The ARAP algorithm [SA07, LZX*08] uses a local-global approach where UV triangles are first individually aligned to their target shape in a local step, before solving a global Poisson system over UV vertices. Similarly, the SLIM algorithm [RPPSH17] iteratively minimizes a quadratic proxy of the deformation energy. Other works focus on identifying good descent directions to improve the convergence rate of the energy minimization by preconditioning the gradient descent direction with a positive-definite Hessian approximation [SS15, SPSH*17] or the mesh Laplacian [KGL16].

For our purposes, we can in principle pick any existing single-patch optimization method, and employ it in the re-optmization process that follows a merge operation; we need to adapt it by changing its objective, simply by redefining the minimized functional to minimize the isometric distortion with respect to the original 2D UV triangle rather than the 3D triangle. Among all the possible choices, we choose to formulate our single-patch optimizer after [LZX*08], which matches our requirements. See Section 4.2.2 for detail. Other alternatives would be viable, for example to strike a different tradeoff between computational speed and result quality.

Global Mesh Parametrization Surfaces with a topology which is not disk-like, or presenting intrinsic curvature far from flat, must be cut before computing the flattening with single-patch parametrization techniques. Finding the optimal cut layout is at the core of the global mesh parametrization task, and is known to be NP-Hard [EHP04]; effective heuristics, however, are the subject of intensive research spanning multiple decades.

There is a huge number of automatic methods, approaching the problem for a variety of angles; their enumeration exceeds the scope of this work, and the reader is referred to the State of The Art sections of recent papers, such as [PTH*17]. In spite or recent

advancements, the search for a fully automatic solution is still an open problem, and the industry practice tends to leave artists partially in control of the process, which is among the motivations for our work.

In general terms, these methods share with ours the need to balance the amount of cutting with the total distortion. Many methods seek this balance implicitly [GGH02, SCOGL02, SH02], while others explicitly combine the number of seams and the parametrization distortion into a single energy functional to be minimized [PTH*17, LKK*18].

In order to approach this trade-off methodically, one needs first a measure of the two quantities that must be balanced. Several well-principled measures for distortion have been proposed in the context of single-patch mesh parametrization (see above), but measures for the amount of cutting have fewer predecessors. Many global parametrization methods implicitly minimize quantities like number of seam edges, or total length of seams (in either 3D or UV space), which is unsatisfactory as these quantities depend on either meshing resolution, texture resolution, or model scale. The definitions employed in [PTH*17] or [LKK*18] are carefully designed for efficient numerical minimization, and present similar problems. In this work, we adopt the definition of atlas "solidity" [MPCT20], which is scale and resolution independent.

Signal-Specialized UV Maps In a similar spirit to our work, a long standing idea in parametrization is that the content of the texture should be taken in account when optimizing a UV-map. For example, in [SWB98], an *importance map* is computed from an existing texture, and used to drive the texture coordinate optimization. In [SGSH02], the authors introduce a signal-stretch metric that also takes into account the signal approximation error introduced by the mapping, and [BTB02] presents a relaxation algorithm that evenly distributes frequency content across the parametrized image.

Our method does not experiment with signal specialization, considering each used pixel of the original texture equally important, and always aiming at their 1:1 preservation in the final textures. The two concepts are, however, not mutually exclusive; in the future, our approach could be augmented with signal specialization, but a trade off will be necessary to combine this with our objective of limiting the texture area to be resampled.

Mesh repairing One of the motivations behind our work is that typical photogrammetry meshes come with inconsistencies hindering most automatic parametrization methods. One alternative to our proposed strategy would consist, then, in addressing these inconsistencies prior to a standard re-parametrization of the mesh. Automatic mesh repairing has a long history in literature [ACK13], but it is not a completely solved problem. In addition, this route would still need to globally parametrize a mesh featuring high geometric complexity and resolution, which is also not a solved problem. Finally, there would be the drawback related to the change in texel density and the need of resampling, as discussed. While mesh repairing can be used in conjunction with our method to improve model quality, it is not a prerequisite and we validate our approach on unmodified real-world models.

Alleviating the effect of seams Our strategy is to strive to reduce the number of seams in photogrammetry models. A class of orthogonal alternative strategies, surveyed in [YLT19], consists instead in attenuating some or all of their many detrimental *effects*, such as the need for texel duplication (e.g. [LH06]) and vertex duplication (e.g. [Yuk17, Tar16]), visual artifacts (e.g. [RNLL10, LFJG17]), or texture packing inefficiency (e.g. [BL08]). Many of these techniques require substantial changes in the downstream application; among the others, [LFJG17] most closely resemble our approach in that an existing UV-map and texture is modified to produce seamless transitions. There are, however, important differences: first, only the problem of visual artifacts is addressed in [LFJG17]; second, the targeted meshes typically feature a much smaller number of seams compared to photogrammetry models, and it is not clear how the method performs on more complex inputs.

Packing of texture charts Packing 2D charts efficiently inside the texture rectangle in an important step of any atlas-based global parametrization method. It is a form of polygon packing, a combinatorial optimization problem in the field of Computational Geometry that can only be addressed by heuristics. In the context of texture mapping, a popular approach is to rasterize the charts and pack them in a coarse 2D grid while greedily minimizing various cost functions such as the height of the packing horizon [LPRM02], or the wasted space left between charts [SWG*03, NS11]. Packing efficiency can be improved by detecting empty areas induced by complex outlines and reduce them by splitting charts, so to obtain more regular and compact boundaries [LVS18]; this approach can be extremely effective, but introduces additional cuts, in direct contrast to our objective.

For our purposes, we can adopt any existing packing algorithm, on the condition that we are able to adapt it by imposing the grid-preserving constraints outlined in Sec. 2. We do not claim any advancement on the packing algorithm itself; rather, our strategy is to surpass the packing efficiency of the input UV-map by virtue of feeding fewer and more regular charts to the packing algorithm.

4. Phases of the algorithms

Our algorithm works entirely on the UV-layout of the original 3D mesh, which is a 2D triangular mesh. With "vertex" and "edge", we refer to a 2D vertices and 2D edges in this layout (we will use "mesh-vertex" and "mesh-edges" when we refer to the 3D mesh).

Most mesh-edges correspond to one edge each, but a mesh-edge on a texture seam corresponds to two distinct edges, one on each side of the cut, which we call "linked". A pair of linked edges can be "dissolved" by fusing together the vertices at their endpoints.

The UV-layout is partitioned into a number of connected components, called "charts". Two charts are said to be linked when they are bounded by (one or more) linked edges.

4.1. Enumeration of candidate merge operations

To enumerate the candidate operations for our greedy approach we consider, for every pair of linked charts, the merge operation that consists in fusing the two charts into one by dissolving all the linked

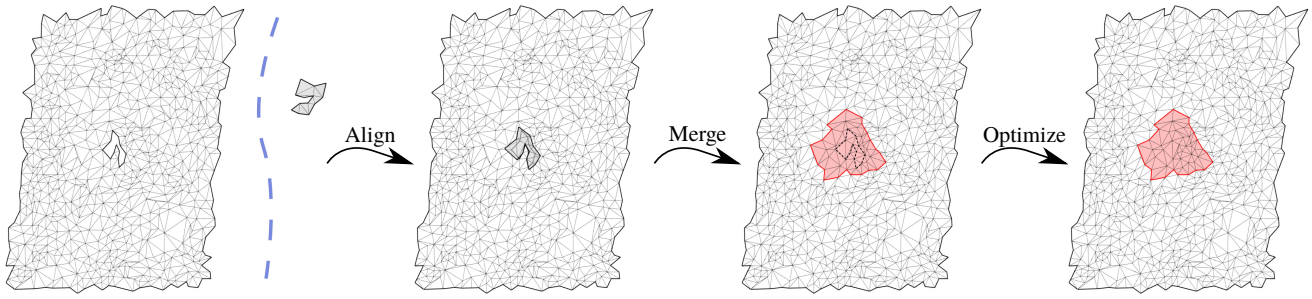


Figure 4: Steps required to perform a merge operation. The two charts are first aligned by rigidly moving the smaller one so to maximize the match between the edges to be fused (in this case, its entire boundary). Then, the edges are merged by displacing and fusing vertices, which can introduce distortions and overlaps. Finally, vertex positions are locally re-optimized in area proportional to the displacements (red shade), so that distortions with respect to the original shape, and possibly folds, are removed.

edges on their boundary. In principle, smaller operations could also be considered, dissolving only some subset of the linked edges shared by the two charts; however, in order to reduce the size of the search space, those are not considered by our greedy strategy. This choice is, again, justified by the combinatorial complexity of the problem being faced.

Note that pairs of linked edges can also be found on the boundary of the same chart. For each connected group of such linked edges, we list an additional "intra-chart" candidate operation, which consists in dissolving them without reducing the number of charts. Therefore, multiple distinct intra-chart operations can be considered for a single chart, one for each set of connected linked edges.

Any merge operation that would result in a chart that is not genus 0 or becoming a closed surface is deemed invalid, and always discarded.

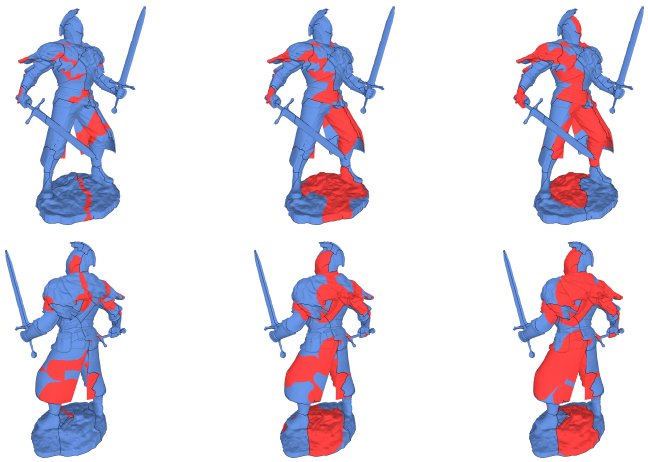


Figure 5: The impact of α on the size of the optimized area (highlighted in red) at the end of the defragmentation loop. In this example, using values larger than 5 increases the area undergoing UV re-optimization, for a small impact the removal of seams (black lines).

4.2. Performing a merge operation

A merge operation S dissolves a certain number of edge pairs, coming from two charts a and b. For intra-chart operations, in the following, we simply consider $b = a$. This operation (see Figure 4) is performed in three steps: aligning the chart boundaries, merging and minimizing distortion, and finally checking the consistency and quality of the updated UV-map; the process can ultimately either succeed or fail, according to a number of criteria.

4.2.1. Initial alignment

Operation S has the effect of fusing together a certain number of vertices, by first transforming them into a common 2D location and then topologically unifying them in the mesh structure. These displacements distort all affected triangles, against our second Goal. If the distortion surpasses a given threshold, the operation is rejected and rolled back. To increase the chance of success, we strive to attenuate the needed distortions with a number of countermeasures.

As a preliminary step of every operation, we apply a 2D rigid transformation M_S to the smallest of the two charts to minimize, in the least-squares sense, the distance between every pair of vertices to be fused. Finding M_S is a straightforward shape-matching problem that can be efficiently solved [SHR16], unless the operation is intra-chart, in which case the linked edges belong to the same UV-component and we simply set M_S as the identity transformation. Note that the rotation imposes a resampling of the smallest chart in the final image, which is traded for the reduction of the number and length of texture seams.

Once the linked edges are rigidly aligned, we move each pair of vertices to be fused into their average 2D position and merge them. In order to to mitigate the introduced distortions, we then optimize the position of a subset of vertices in the neighborhood of the dissolved seam, as follows.

4.2.2. Local UV re-optimization

Intuitively speaking, distortions caused by small displacements of a vertex can be removed by changing only immediately adjacent triangles, while large displacements require optimizing a correspond-

ingly wide band around the vertex. We follow this intuition by identifying all triangles with a 2D distance within α times the displacement of any fused vertex. The α parameter controls the trade-off between the size of the UV optimization area and residual distortion, and thus the likelihood for the operation to succeed and ultimately the removal of seams. In all our experiments we use a value of $\alpha = 5$, which we empirically identified as a good compromise (see Figure 5).

Our distortion minimization procedure closely follows the As-Rigid-As-Possible (ARAP) pararametrization approach [LZX*08], which targets isometric distortions. In our case we seek to minimize the isometric distortion with respect to the original UV layout: this simply means that we set the target shape and size of each triangle as the one in the original 2D layout rather than the one in the 3D mesh. Other single patch parametrization techniques could be adapted in a similar way. Iterative methods such as ARAP fit well with our approach because they can exploit the existence of a valid starting configuration.

In the optimization, we freeze (i.e. set as constant) the position of vertices outside the optimization area, while vertices inside the optimization area are free to move during the global phase of ARAP. This greatly reduces the size of the ARAP instance, dramatically speeding up the merge execution.

After the optimization, the value of the minimized energy, as defined in [LZX*08], is measured, and summed over the optimized area. If it surpasses a "local" distortion threshold, the operation is rejected because it is deemed to introduce too much distortion. Additionally, the accumulated distortion over the entire UV-map is also tracked, and operations are also rejected if they exceed a global distortion threshold (see Sec. 5 for the values employed in our experiments).

4.2.3. Consistency Checks

Performing a merge can infringe consistency constraints in various ways.

A merge S can introduce *global* UV overlaps when, after the transformation M_S is applied, the boundaries of the two charts a and b collide (disregarding the edges that are fused by the operation). This condition is detected early, before UV re-optimization takes place.

Another occurrence of global UV overlaps can be produced by the UV re-optimization, when the boundary of the combined chart, inside the optimized area, is made to intersect. Similarly, the re-optimization can introduce *local* UV overlaps (triangle folds), in spite of these configurations being penalized by the minimized energy. These conditions are detected after UV re-optimization.

Note that we reject operations that introduce *new* overlaps, but tolerate the ones that were already present in the input map, which are detected and labelled in pre-processing for this purpose. Our scenario justifies this choice. Even if the final UV-map fixed a pre-existing overlap condition, there would be no distinct color data to fill the no-longer-overlapping regions, and the final texture resampling would just replicate the artifacts of the input texture.

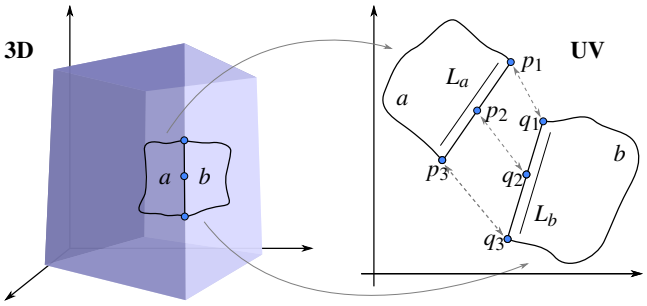


Figure 6: Quantities used to estimate the appeal score of a merge operation involving charts a and b.

4.3. Ranking of the candidate operations

Attempting merges, and in particular the UV re-optimization stage, is the most time consuming part of our pipeline. The global efficiency of our algorithm depends on the ability to identify sensible operations to be attempted first, and avoid spending times on operations that will fail and need to be reverted. In addition, the success or failure of an operation can depend on the operations that previously affected the same charts. For all these reasons, we devise a method to quickly evaluate an "appeal" score based on the geometric attributes of the charts involved (Figure 6), with higher scores assigned to operations estimated as more likely to succeed and more beneficial towards our two conflicting goals.

In order to reduce the fragmentation (Goal 1), we want the perimeters of the affected charts to be reduced. Therefore, we want to prioritize operations that dissolve a large fraction of the boundary of either chart. Specifically, an operation S, fusing edges in a with a total length L_a and edges in b with a total length L_b, has a potential benefit estimated as

$$\text{Benefit}(S) = \max\left(\frac{L_a}{\text{Perimeter}(a)}, \frac{L_b}{\text{Perimeter}(b)}\right). \quad (1)$$

The maximal value of 1 occurs for operations which place one chart completely inside a hole of the other, which, indeed, we want to attempt early in the greedy process, as they are likely to succeed, and simplify boundaries which otherwise risk causing global overlaps later.

Performing an operation S implies fusing a certain number of vertex pairs. The farther the vertices are after being rigidly aligned, the more they will have to be displaced, and the more severe distortions will likely be introduced by the operation, even after being minimized by the UV re-optimization. Therefore, we assign to S an estimated cost, computed as the average residual distance:

$$\text{Cost}(S) = \frac{1}{|V_S|} \cdot \sum_{(p,q)\in V_S} \|p - \text{M}_S(q)\|_2, \quad (2)$$

where V_S is the set of vertex pairs to be fused, and M_S is the rigid transformation matrix associated to the operation S. Note that at this stage the transformation M_S is only used to evaluate the matching error, although we cache it to avoid recomputing it later when the actual operation is executed.

Additionally, we observe that operations where either chart is

small are in general computationally less expensive to attempt, and less likely to fail. Therefore, we favor operation S with a bonus given by

$$\text{SizeBonus}(S) = 1/\min(\text{Area}(a), \text{Area}(b)) \tag{3}$$

Finally, an operation S that has already failed $k_S > 0$ times is likely (although not guaranteed) to fail again when re-attempted. Therefore, we introduce a back-off penalty factor to make it increasingly less appealing:

$$\text{BackOff}(S) = 2^{-k_S}. \tag{4}$$

The final appeal score of an operation S is then given by

$$\text{Appeal}(S) = \frac{\text{Benefit}(S)}{\text{Cost}(S)} \cdot \text{SizeBonus}(S) \cdot \text{BackOff}(S) \tag{5}$$

4.4. Updating candidate operations

If an operation S fails, it is removed from queue of potential operations Q, and placed in a set of "failed operations" F.

If it succeeds, then we need to update all the candidate operations in Q involving each of the affected charts a and b (which will now affect the unified chart resulting from the execution of S), and also to reevaluate their appeal and update the ranking in Q.

In this case, we also reinsert any failed operations affecting a or b, moving them from F to Q (although with a diminished appeal score, due to the back-off factor). There are a variety of reasons that can make a previously failed operation succeed after a chart has been updated. One occurring fairly easily is that is that global overlaps become less frequent when chart boundaries are simplified by previous operations, e.g. by filling holes.

4.5. Atlas Repacking

When all merge operations have been attempted, the resulting charts must be packed to obtain the final texture layout.

The space of 2D transformations that can be applied to the original texture coordinates without requiring a resampling of the input only includes rotations by multiples of 90-degrees and translations by an integer number of pixels, and their horizontal and vertical reflections. These transformations, which have been referred to as "grid-preserving" in previous literature [RNLL10], can be exploited in conjunction with our defragmentation procedure to avoid resampling possibly large portions of the new texture atlas and match the input texture reconstruction exactly when rendering the output model (Figure 7). Namely, we can identify within each chart the maximal set of rigidly transformed UV-triangles that share the same transformation matrix, and constrain the packing algorithm to be grid-preserving with respect to those triangles.

We begin by analyzing the final charts to determine if they should be constrained by the packing algorithm. In each chart, we detect the triangles that can be rigidly transformed back into their input position, and record the corresponding rotation angle. Let θ be the angle shared by the subset of triangles maximizing the chart area (we have chosen to maximize 3D area, hence visibility during rendering, rather than 2D area). If this area is larger than a fixed

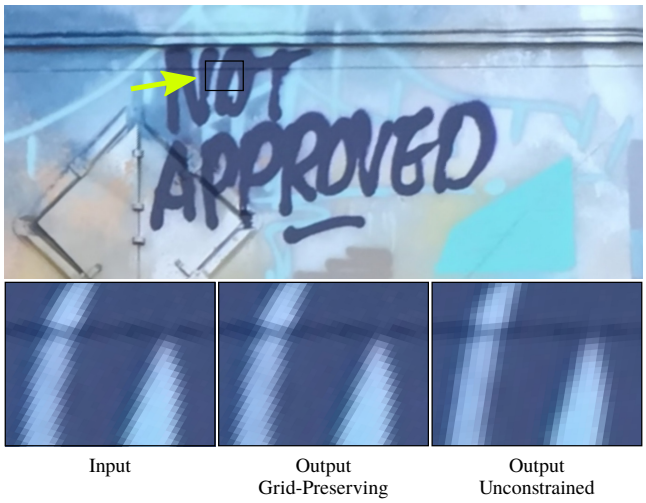


Figure 7: Exact correspondence between the input and output texels can be achieved by constraining a rigidly-transformed chart subregion to be grid-preserving. First closeup: a rendering of the input. Second closeup: a rendering of the output with grid-preserving packing. Third close-up: when the packing is unconstrained, the texture is resampled.

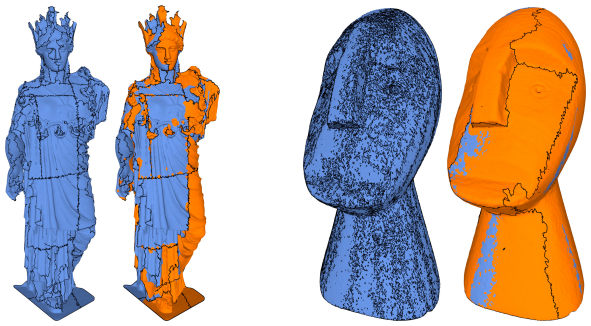


Figure 8: Two examples of texture defragmentation, with the resampled areas highlighted in orange. As expected, the resampled area increases as the models undergo a larger number of successful merge operations (right).

percentage of the chart (we used 5%), then we label all triangles associated to θ as "preserved", and the chart as constrained. This means that we will only allow the packing algorithm to rotate this chart by $k(\pi/2) - \theta$. Finally, after packing, we enforce the grid-preserving constraint of all the preserved triangles by applying a sub-pixel correction offset to each chart, ensuring that the final positions of the relevant vertices is indeed an integer translation of the rotated input. As highlighted in Figure 8, the impact of such grid-preserving packing is heavily affected by the input fragmentation, the chart size distribution and the overall efficacy of the defragmentation step.

Our packing algorithm, following [LPRM02, SWG*03], is based on rasterizations on a coarse grid. We approximate the region occupied by a chart with the grid cells it partially or completely

cover (accounting for an extra "safety border" of texels, needed to avoid bleeding).

We track the packing area and available space using multiple horizons [NS11]; for a given packing direction, we track both the upper horizon, and an inner horizon that maps the largest available gap below the upper horizon. Using multiple horizons allows placement of smaller charts between larger ones, which is particularly useful in presence of large charts (which our merging algorithm is designed to produce). Chart placement is performed in a greedy fashion, minimizing a cost function that penalizes wasteful placements. If the chart is placed using the inner horizon, we minimize the wasted space as [LPRM02, SWG*03], that is, the empty grid area left between a chart's bottom horizon and the current horizon profile. Otherwise, we place the chart in the position that minimize the horizon increase. The sequence of chart placements is usually determined by sorting charts from largest to smallest, as this provides reasonably good results without needing to enumerate all the possible permutations of the sequence. If we have fewer than 100 charts, we also attempt a number of randomized sequences, which we have found to help improve packing efficiency.

Finally, we detect the bounding square of the used texture space and we trim the texture to this size, which is usually smaller that the original size (see Results in Sec. 5). The final texture is simply constructed by either copying texels for the original texture verbatim, inside the triangles labeled as "preserved", and interpolating pixels in the other areas. In our prototype, we color all unused texels by extending and blending the colors of used texels, using push-pull algorithm in [GGSC96], in order to contain the bleeding artifacts at any MIP-map level.

4.6. Extension to Multiple Texture Sheets

Photogrammetry models frequently encode the surface color into multiple texture sheets, assigning different part of the model to different sheet. This can be done, for example, to circumvent the hard-coded upper bound on texture resolution, thus affording a larger number of texels for the model. Since our objective is to produce a UV-map with the same mapping resolution of the input, we adapted the packing algorithm to handle this occurrence. During the packing, we target a texture size equal to the maximal resolution of the sheet. when a chart cannot be accomodated in the current sheet, we initialize a new empty sheet.

5. Results

Implementation We have implemented our algorithm in C++ using the VCG Library and Eigen. The source code of this reference implementation is publicly available and provided as additional material.

Experiments We evaluate the performance and robustness of our method over a benchmark that reflects the textured models produced by the current generation of photo-reconstruction tools [MPCT20]. The benchmark consists of 568 models of varying resolution from several different sources and constructed with different photogrammetry tools. They exhibit complex geometry, dense and uneven meshing tessellation, inconsistencies such as non-manifoldness, non-coherent face orientation, degenerate faces, and so on. Textures are high-resolution and in many cases consist of multiple image files.

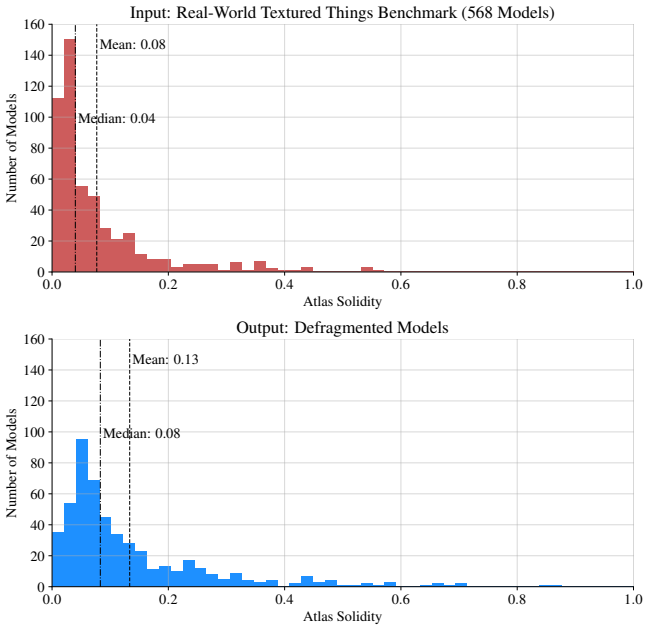


Figure 9: Distribution of Atlas Solidity values before and after processing the benchmark.

We batch process the dataset, successfully producing a result from every input. Examples of processed models are reported in Figure 16.

We use the following settings: we allow alignment errors up to twice the seam length, and forbid operations that do not reduce the perimeter by 20% for at least one of the charts; the ARAP energy thresholds are 0.5 for the local optimization area, and 0.025 for the energy of the entire atlas (the latter compensates for the relaxed matching threshold used).

Processing time The average processing time is around 5 minutes on a PC equipped with an Intel Core i7 8750H and 32 GBs of RAM.

Defragmentation efficacy We empirically assess the efficacy by statistically examining a number of quantitative measurements and practical impacts in the produced output.

We measure the *Atlas Solidity* [MPCT20] value of each model before and after our defragmentation, and plot the distributions in Figure 9. Atlas Solidity is an aggregate measure characterizing the overall amount of seams in an atlas, defined as the inverse of the ratio between the total length of texture seams and the length of the minimal perimeter of a 2D shape encircling the same texture area, with a Solidity of 1 being the best and theoretical maximum. Other measures comprise the reduction in the total number of charts, and UV border length (Figure 10).

A direct practical benefit of defragmenting the UV-map is the re-

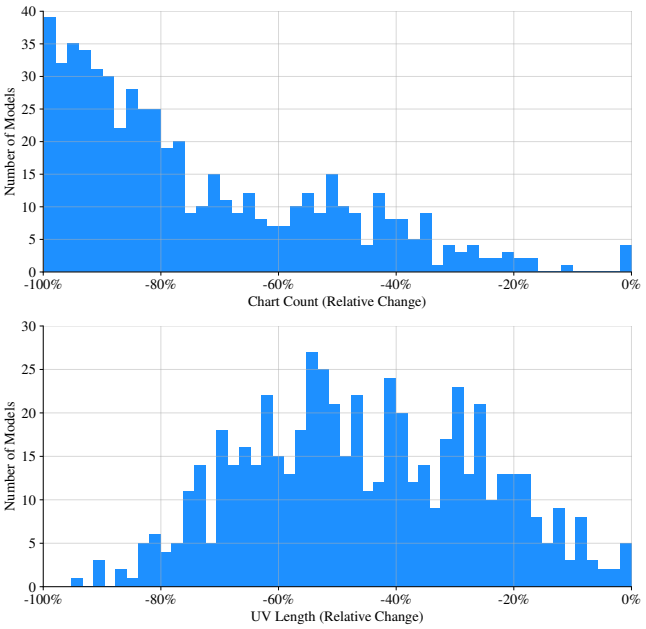


Figure 10: Aggregate chart count and UV border length reductions achieved by our method.

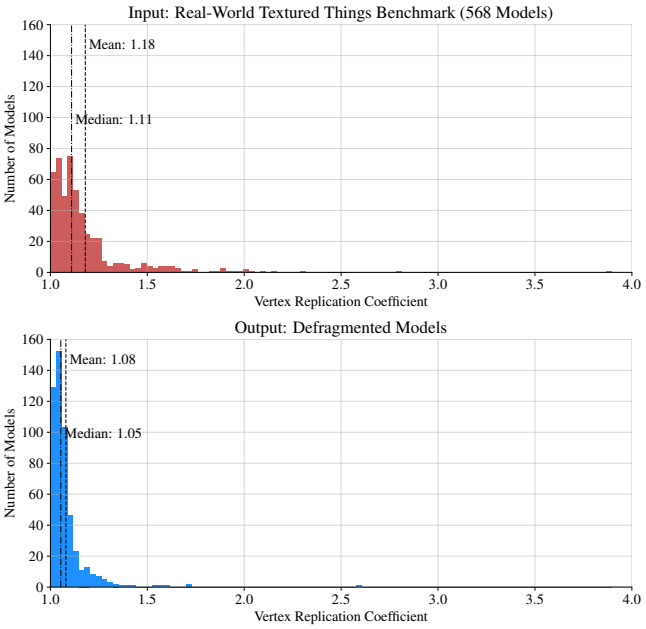


Figure 11: Vertex replication factors before and after processing the dataset.

duction of vertex replications that are necessary to encode seams, which impacts the memory required to store and render the model (e.g. the GPU buffers). Distributions of the vertex replication coefficients, i.e. the ratios of the number of 2D vertices to 3D vertices, before and after optimizing the models with our method is reported in Figure 11. Another direct effect is the reduction of the overall

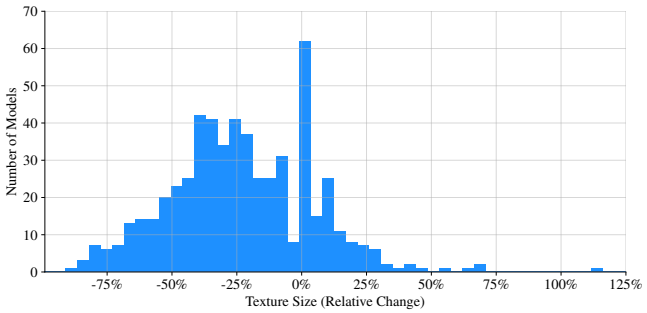


Figure 12: Efficacy of our method in reducing the size of texture images required to render the models.

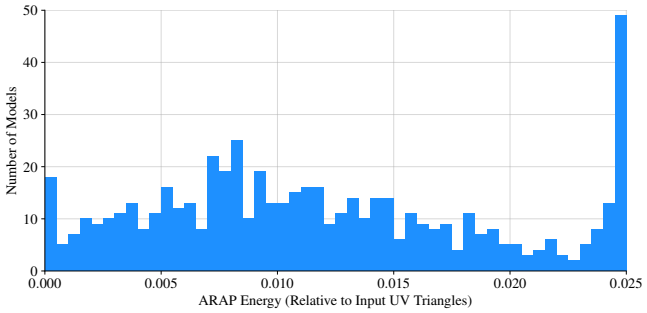


Figure 13: Distribution of ARAP energies after optimizing the texture atlases. Note the very low global cut-off at 0.025, and also how in most cases the energy values remain well below this threshold.

number of texture samples while preserving the input texel density and mapping resolution, reported in Figure 12.

The different measures concur in showing a clear reduction of the fragmentation and of the associated practical problems, both on average and in each instance. For example, chart count decreases up to 99% and 72% on average, vertex duplication decreases up to 43% and 6.8% on average, and atlas solidity increases by up to 20 times and 21% on average.

The one exception is overall texture size. While the average reduction is 23.3%, there are a few cases where the the output texture is actually larger than the input texture. In our analysis, two factors contribute to this result. First, we always pack our textures using a padding size of 4 texels (meaning charts are at least 8 texels apart), which ensures correct filtering when MIP-mapping is used up to level two, while the several of the original images accommodate for no padding at all. Second, the larger produced charts can be harder to pack; if deemed necessary, this could be countered with strategies such as [LVS18, LFY*19], which reintroduce straight cuts for the sake of packing efficiency. We omitted 7 models from this analysis, as they are anomalous in the context of photogrammetric reconstructions, in that they contain either instanced objects or faces with random texture coordinates; these configurations mislead our system into trying to unnecessarily allocate texture space for these areas.

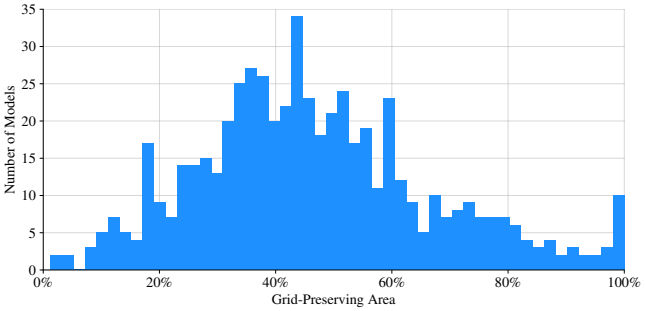


Figure 14: Distribution of fractional object areas (computed in 3D space) that do not require resampling the input texture and where the signal reconstruction exactly matches the input data.

Introduced distortions As stated, our second objective is to keeping the mapping distortion faithful to the input UV-map low, for the reasons explained in Sec. 1.1. We assess the fulfillment of this goal by measuring the ARAP energy (which, as discussed, measures the loss of isometry with respect to the original UV layout,) as done in Figure 13.

A closely connected objective, as stated, is to avoid any resampling in areas of the original model, which is one main characteristic setting our method apart from any predecessor (to the best of our knowledge). We report statistics of the fulfillment of this objective in Figure 14. On average, 46.6% of the surface area was *not* resampled.

5.1. Ablation study

To validate our design choices, we have performed an ablation study measuring the individual impacts of the local optimization phase (for various values of the expansion parameter α), the greedy ordering of the operations (against a random order), and of each component of the Appeal function (5) used to order them.

The results, reported in Table 1, confirm that each stage of our algorithm yields a significant benefit, and that our choice of parameters strike good trade-off.

In particular: larger values of α, expanding the UV re-optimized area, improve the results in terms of atlas Solidity, but increase the need of texture resampling and the running time considerably; the opposite is true for smaller values. The ordering of the operations is proved important, as adopting a random order produces consistently worse results and drastically increases the running time; also, dropping any component of the Appeal function clearly degrades the performance, with the possible exception of discarding SizeBonus, which leads to mixed results according to the other metrics. Finally, a designated experiment confirms that reattempting previously failed operations is unquestionably beneficial.

5.2. Comparison against computing an entirely new UV-map

In support of the premise of our work, we compare against traditional approaches which ignore the existing UV-map and rebuild

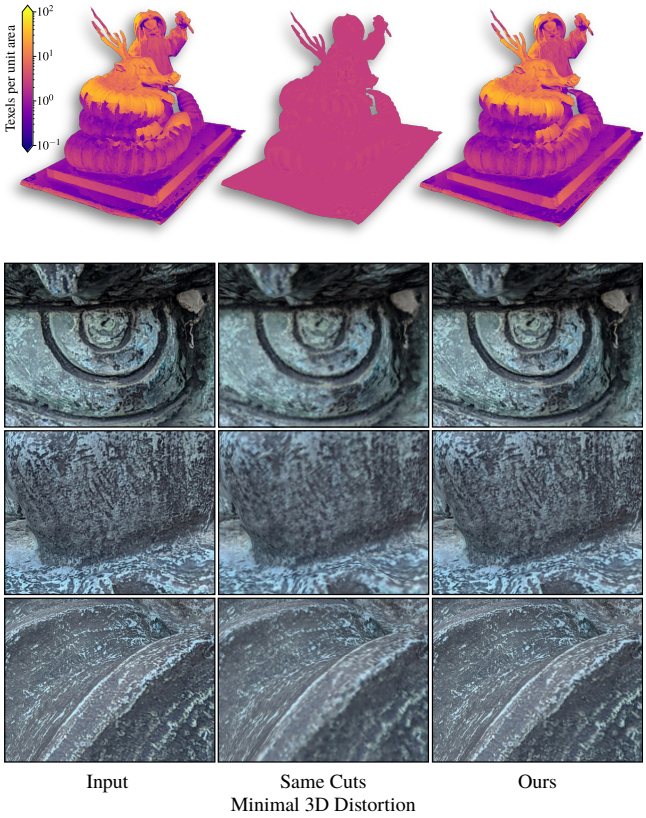


Figure 15: Comparison between generating a UV-map *ex-novo* and our method on a model exhibiting uneven texel distribution across the surface. Top: traditional parametrization approaches minimize distortion w.r.t. the 3D surface and are unable to preserve the input texel distribution. Bottom: close-ups of areas mapped at high-resolution reveal how this results in loss of high-frequency detail, while our approach produces a noticeably sharper texture.

it from scratch; an obstacle is that, as mentioned, previous experiments [MPCT20] already establish that a direct application of off-the-shelf software suites or existing implementations of automatic UV-mappers fails to produce satisfactory results in a majority of cases (due to various inconsistencies on the input models or their excessive resolution). We consider this as an argument in favor of our approach.

A central point of our strategy is that we strive to preserve the shape of the existing UV-map triangles, rather than minimizing the distortions of the 3D-to-2D mapping, which can make a significant difference when the texture samples distribution and UV-map resolution over the object surface are not uniform (Figure 15, top). To validate this choice, we visually compare our method against the results obtained by applying a state-of-the-art UV-optimization algorithm to the original UV-map, reducing its distortion but without affecting the cut layout of the mesh. The produced UV-map is representative of the results that can be obtained by recomputing a UV-map ex-novo, at least away from cuts. For this experiment, we minimize the symmetric Dirichlet energy [SS15] (minimal for iso-

		Average relative change over Dataset					Average relative change over Baseline					
	Time (s)	Resampled Area	Vertex Replication	Texture Size	Solidity	# Charts	Time (s)	Resampled Area	Vertex Replication	Texture Size	Solidity	# Charts
Baseline	322	+53.4 %	−6.8 %	−23.3 %	+121.0 %	−72.3 %						
$\alpha = 2.5$	295	+39.0 %	−6.1 %	−23.6 %	+89.5 %	−68.5 %	−9.5 %	−26.0 %	+0.7 %	0.0 %	−10.9 %	+25.4 %
$\alpha = 10$	600	+64.6 %	−7.0 %	−23.2 %	+133.2 %	−73.5 %	+130.1 %	+27.8 %	−0.2 %	+0.4 %	+4.3 %	−7.1 %
NoRetry	319	+53.0 %	−6.7 %	−23.4 %	+117.2 %	−71.6 %	−0.1 %	−0.8 %	+0.1 %	−0.1 %	−1.0 %	+6.1 %
NoBC	406	+53.6 %	−6.7 %	−23.4 %	+117.7 %	−72.0 %	+28.2 %	+1.5 %	+0.1 %	+0.4 %	−1.3 %	+2.8 %
NoSB	363	+53.8 %	−6.8 %	−23.6 %	+122.6 %	−72.2 %	+14.7 %	+1.0 %	−0.1 %	−0.2 %	+0.8 %	+1.3 %
Random	607	+55.9 %	−6.5 %	−23.3 %	+115.5 %	−70.7 %	+83.7 %	+7.0 %	+0.3 %	+0.8 %	−0.4 %	+37.7 %

Table 1: Results of the ablation study comparing baseline results of our approach against variations of our algorithm reducing and increasing the UV optimization area expansion α (compared to our baseline value of 5), and altering the greedy sequence of merge operations. *NoRetry*: do not reattempt failed operations; *NoBC*: drop the Benefit/Cost ratio from the Appeal function (5), *NoSB*: drop the SizeBonus term from Appeal function (5); *Random*: rank the operations randomly.

metric mappings) with Composite Majorization [SPSH*17], using scaffolding[JSP17] to ensure the new texture atlas is overlap-free.

A visual comparison is offered in Figure 15. Our approach is visibly better at preserving highly detailed areas and high frequencies of the input texture image.

6. Conclusions

We proposed a new automatic approach designed to attenuate texture fragmentation of an existing texture model, which is a defect systematically arising with photogrammetry-based 3D reconstruction techniques. The method is proved to be effective at reducing the fragmentation and the consequent drawbacks, making the models more usable by downstream applications. Compared to alternative approaches, including a global re-parametrization of the model, the main benefits are reliability, and fidelity to the input image quality, thanks to the ability to preserve the original texel distribution and UV-map resolution. A re-sampling of the texture is avoided on approximately half of the object surface on average. The method was extensively tested and evaluated by batch processing a dataset consisting of 568 real world 3D models created using the most common photogrammetry tools.

Future work This work is, to the best of our knowledge, the first approach to target specifically the goals set in Sec. 2; we think it is very likely that a more effective or more elegant strategy can be employed to reduce the fragmentation or the resampled areas more aggressively. For example, the local optimization itself could be designed to pursue the objective to avoid re-sampling inside each optimized triangle. This objective is "all-or-nothing" in nature, making its optimization difficult; recently, however, advanced numerical methods have been successfully employed over objectives posing a similar challenge [PTH*17, LKK*18].

Limitations Our approach is not well suited for situations where there is a vast redundancy of input photographs, and where super-resolution approaches such as [LTH*17] could be exploited to increase the texture quality. Another scenario is when the texture resolution must also be downsized, which can be the case for example when very high-resolution photographs are available. In both cases, the benefit of inheriting the input texel distribution is diminished (although the reliability and robustness of the proposed method would still be an advantage).

Acknowledgments

This work has been partially supported by the ARIADNEplus project under grant agreement 2019-2023 H2020-INFRAIA-2018-1-823914. We also thanks Alessandro Muntoni for porting the code inside the MeshLab framework.

We wish to dedicate this contribution to our dear colleague and friend Matteo Dellepiane, who initially sketched the research direction at the basis of this work. Your voice is sorely missed.

References

[ACK13] ATTENE M., CAMPEN M., KOBBELT L.: Polygon mesh repairing: An application perspective. *ACM Comput. Surv. 45*, 2 (2013).

[BL08] BURLEY B., LACEWELL D.: Ptex: Per-face texture mapping for production rendering. In *Computer Graphics Forum* (2008), vol. 27, pp. 1155–1164.

[BTB02] BALMELLI L., TAUBIN G., BERNARDINI F.: Space-optimized texture maps. In *Computer Graphics Forum* (2002), vol. 21, pp. 411–420.

[DMA02] DESBRUN M., MEYER M., ALLIEZ P.: Intrinsic parameterizations of surface meshes. *Computer Graphics Forum 21*, 3 (2002), 209–218.

[EDD*95] ECK M., DEROSE T., DUCHAMP T., HOPPE H., LOUNSBERY M., STUETZLE W.: Multiresolution analysis of arbitrary meshes. In *Proceedings of the 22nd Annual Conference on Computer Graphics and Interactive Techniques* (1995), SIGGRAPH '95, pp. 173–182.

[EHP04] ERICKSON J., HAR-PELED S.: Optimally cutting a surface into a disk. *Discrete & Computational Geometry 31*, 1 (2004), 37–59.

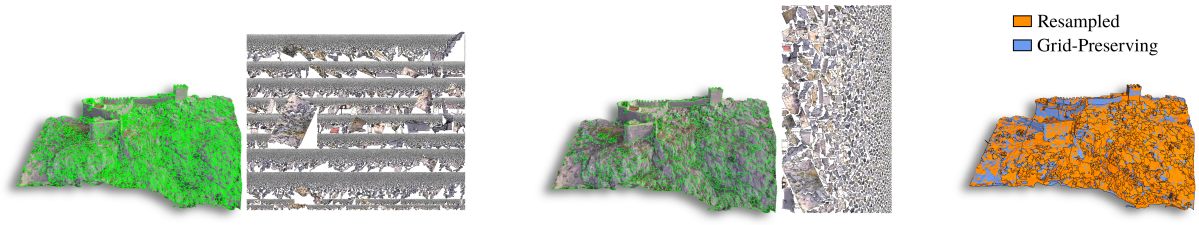
[FH05] FLOATER M. S., HORMANN K.: Surface parameterization: a tutorial and survey. In *Advances in Multiresolution for Geometric Modelling* (Berlin, Heidelberg, 2005), Dodgson N. A., Floater M. S., Sabin M. A., (Eds.), Springer Berlin Heidelberg, pp. 157–186.

[Flo97] FLOATER M. S.: Parametrization and smooth approximation of surface triangulations. *Computer Aided Geometric Design 14*, 3 (1997), 231–250.

[GGH02] GU X., GORTLER S. J., HOPPE H.: Geometry images. *ACM Trans. Graph. 21*, 3 (2002), 355–361.

[GGSC96] GORTLER S. J., GRZESZCZUK R., SZELISKI R., COHEN M. F.: The lumigraph. In *Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques* (1996), SIGGRAPH '96, pp. 43–54.

[GWO*10] GAL R., WEXLER Y., OFEK E., HOPPE H., COHEN-OR D.: Seamless montage for texturing models. *Computer Graphics Forum 29*, 2 (2010), 479–486.



Input
Atlas Solidity 0.004
72574 Charts, 380K Replicated Verts

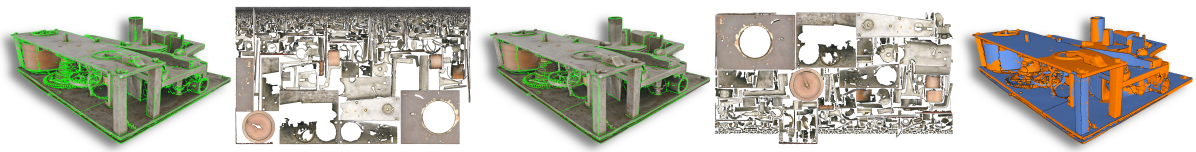
Output
Atlas Solidity 0.015
1462 Charts, 76K Replicated Verts

Output
Grid-Preserving Area: 28%

Input
Atlas Solidity 0.068
646 Charts, 13K Replicated Verts

Output
Atlas Solidity 0.15
87 Charts, 6K Replicated Verts

Output
Grid-Preserving Area: 34%



Input
Atlas Solidity 0.013
16010 Charts, 132K Replicated Verts

Output
Atlas Solidity 0.034
545 Charts, 43K Replicated Verts

Output
Grid-Preserving Area: 51%

Input
Atlas Solidity 0.033
1479 Charts, 30K Replicated Verts

Output
Atlas Solidity 0.063
296 Charts, 15K Replicated Verts

Output
Grid-Preserving Area: 49%

Input
Atlas Solidity 0.009
12050 Charts, 67K Replicated Verts

Output
Atlas Solidity 0.040
154 Charts, 14K Replicated Verts

Output
Grid-Preserving Area: 20%

Figure 16: A gallery of results obtained with our algorithm.

[HG00] HORMANN K., GREINER G.: MIPS: An efficient global parametrization method. In *Curve and Surface Design: Saint-Malo 1999*, Laurent P.-J., Sablonnière P., Schumaker L. L., (Eds.), Innovations in Applied Mathematics. Vanderbilt University Press, Nashville, TN, 2000, pp. 153–162.

[HPS08] HORMANN K., POLTHIER K., SHEFFER A.: Mesh parameterization: Theory and practice. In *ACM SIGGRAPH ASIA 2008 Courses* (2008), SIGGRAPH Asia '08.

[HZ04] HARTLEY R., ZISSERMAN A.: *Multiple View Geometry in Computer Vision*, 2 ed. Cambridge University Press, 2004.

[JSP17] JIANG Z., SCHAEFER S., PANOZZO D.: Simplicial complex augmentation framework for bijective maps. *ACM Trans. Graph. 36*, 6 (2017).

[KGL16] KOVALSKY S. Z., GALUN M., LIPMAN Y.: Accelerated quadratic proxy for geometric optimization. *ACM Trans. Graph. 35*, 4 (2016).

[LFJG17] LIU S., FERGUSON Z., JACOBSON A., GINGOLD Y.: Seamless: Seam erasure and seam-aware decoupling of shape from mesh resolution. *ACM Trans. Graph. 36*, 6 (2017).

[LFY*19] LIU H.-Y., FU X.-M., YE C., CHAI S., LIU L.: Atlas refinement with bounded packing efficiency. *ACM Trans. Graph. 38*, 4 (2019).

[LH06] LEFEBVRE S., HOPPE H.: Perfect spatial hashing. *ACM Trans. Graph. 25*, 3 (2006), 579–588.

[LI07] LEMPITSKY V., IVANOV D.: Seamless mosaicing of image-based texture maps. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2007), IEEE Computer Society, pp. 1–6.

[LKK*18] LI M., KAUFMAN D. M., KIM V. G., SOLOMON J., SHEFFER A.: Optcuts: Joint optimization of surface cuts and parameterization. *ACM Trans. Graph. 37*, 6 (2018).

[LPRM02] LÉVY B., PETITJEAN S., RAY N., MAILLOT J.: Least squares conformal maps for automatic texture atlas generation. *ACM Trans. Graph. 21*, 3 (2002), 362–371.

[LTH*17] LEDIG C., THEIS L., HUSZAR F., CABALLERO J., CUNNINGHAM A., ACOSTA A., AITKEN A., TEJANI A., TOTZ J., WANG Z., SHI W.: Photo-realistic single image super-resolution using a generative adversarial network. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (July 2017), pp. 105–114.

[LVS18] LIMPER M., VINING N., SHEFFER A.: Box cutter: Atlas refinement for efficient packing via void elimination. *ACM Trans. Graph. 37*, 4 (2018).

[LZX*08] LIU L., ZHANG L., XU Y., GOTSMAN C., GORTLER S. J.: A local/global approach to mesh parameterization. *Computer Graphics Forum 27*, 5 (2008), 1495–1504.

[MPCT20] MAGGIORDOMO A., PONCHIO F., CIGNONI P., TARINI M.: Real-world textured things: A repository of textured models generated with modern photo-reconstruction tools. *Computer Aided Geometric Design 83* (2020).

[NS11] NÖLL T., STRIEKER D.: Efficient packing of arbitrary shaped charts for automatic texture atlas generation. *Computer Graphics Forum 30*, 4 (2011), 1309–1317.

[PTH*17] PORANNE R., TARINI M., HUBER S., PANOZZO D., SORKINE-HORNUNG O.: Autocuts: Simultaneous distortion and cut optimization for uv mapping. *ACM Trans. Graph. 36*, 6 (2017).

[RNLL10] RAY N., NIVOLIERS V., LEFEBVRE S., LÉVY B.: Invisible seams. *Computer Graphics Forum 29*, 4 (2010), 1489–1496.

[RPPSH17] RABINOVICH M., PORANNE R., PANOZZO D., SORKINE-HORNUNG O.: Scalable locally injective mappings. *ACM Trans. Graph. 36*, 2 (2017).

[RSN*14] REMONDINO F., SPERA M. G., NOCERINO E., MENNA F., NEX F.: State of the art in high density image matching. *The Photogrammetric Record 29*, 146 (2014), 144–166.

[SA07] SORKINE O., ALEXA M.: As-rigid-as-possible surface modeling. In *Proceedings of the Fifth Eurographics Symposium on Geometry Processing* (2007), SGP '07, pp. 109–116.

[SC17] SAWHNEY R., CRANE K.: Boundary first flattening. *ACM Trans. Graph. 37*, 1 (2017).

[SCD*06] SEITZ S. M., CURLESS B., DIEBEL J., SCHARSTEIN D., SZELISKI R.: A comparison and evaluation of multi-view stereo reconstruction algorithms. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2006), IEEE Computer Society, pp. 519–528.

[SCOGL02] SORKINE O., COHEN-OR D., GOLDENTHAL R., LISCHINSKI D.: Bounded-distortion piecewise mesh parameterization. In *Proceedings of IEEE Visualization* (2002), IEEE Computer Society, pp. 355–362.

[SGSH02] SANDER P. V., GORTLER S. J., SNYDER J., HOPPE H.: Signal-specialized parametrization. In *Proceedings of the 13th Eurographics Workshop on Rendering* (2002), EGRW '02, Eurographics Association, pp. 87–98.

[SH02] SHEFFER A., HART J. C.: Seamster: inconspicuous low-distortion texture seam layout. In *Proceedings of IEEE Visualization* (2002), IEEE Computer Society, pp. 291–298.

[SHR16] SORKINE-HORNUNG O., RABINOVICH M.: Least-squares rigid motion using SVD, 2016. Technical note.

[SPR*07] SHEFFER A., PRAUN E., ROSE K., ET AL.: Mesh parameterization methods and their applications. *Foundations and Trends® in Computer Graphics and Vision 2*, 2 (2007), 105–171.

[SPSH*17] SHTENGEL A., PORANNE R., SORKINE-HORNUNG O., KOVALSKY S. Z., LIPMAN Y.: Geometric optimization via composite majorization. *ACM Trans. Graph. 36*, 4 (2017).

[SS15] SMITH J., SCHAEFER S.: Bijective parameterization with free boundaries. *ACM Trans. Graph. 34*, 4 (2015).

[SSGH01] SANDER P. V., SNYDER J., GORTLER S. J., HOPPE H.: Texture mapping progressive meshes. In *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques* (2001), SIGGRAPH '01, Association for Computing Machinery, pp. 409–416.

[SWB98] SLOAN P.-P. J., WEINSTEIN D. M., BREDERSON J.: Importance driven texture coordinate optimization. *Computer Graphics Forum 17*, 3 (1998), 97–104.

[SWG*03] SANDER P. V., WOOD Z. J., GORTLER S. J., SNYDER J., HOPPE H.: Multi-chart geometry images. In *Proceedings of the 2003 Eurographics/ACM SIGGRAPH Symposium on Geometry Processing* (2003), SGP '03, Eurographics Association, p. 146âĂŞ155.

[Tar16] TARINI M.: Volume-encoded uv-maps. *ACM Trans. Graph. 35*, 4 (2016).

[TCS03] TARINI M., CIGNONI P., SCOPIGNO R.: Visibility based methods and assessment for detail-recovery. In *Proceedings of IEEE Visualization* (2003), IEEE Computer Society, pp. 457–464.

[Tut63] TUTTE W. T.: How to draw a graph. *Proceedings of the London Mathematical Society 3*, 1 (1963), 743–767.

[YLT19] YUKSEL C., LEFEBVRE S., TARINI M.: Rethinking texture mapping. *Computer Graphics Forum 38*, 2 (2019), 535–551.

[Yuk17] YUKSEL C.: Mesh color textures. In *Proceedings of High Performance Graphics* (2017), HPG '17.

[ZMT05] ZHANG E., MISCHAIKOW K., TURK G.: Feature-based surface parameterization and texture mapping. *ACM Trans. Graph. 24*, 1 (2005), 1–27.