Meta-Relief Texture Mapping with Dynamic Texture-Space Ambient Occlusion

Frederico A. Limberger, Victor C. Schetinger, Manuel M. Oliveira
Instituto de Informática – UFRGS
Email: {falimberger, vcschetinger, oliveira}@inf.ufrgs.br

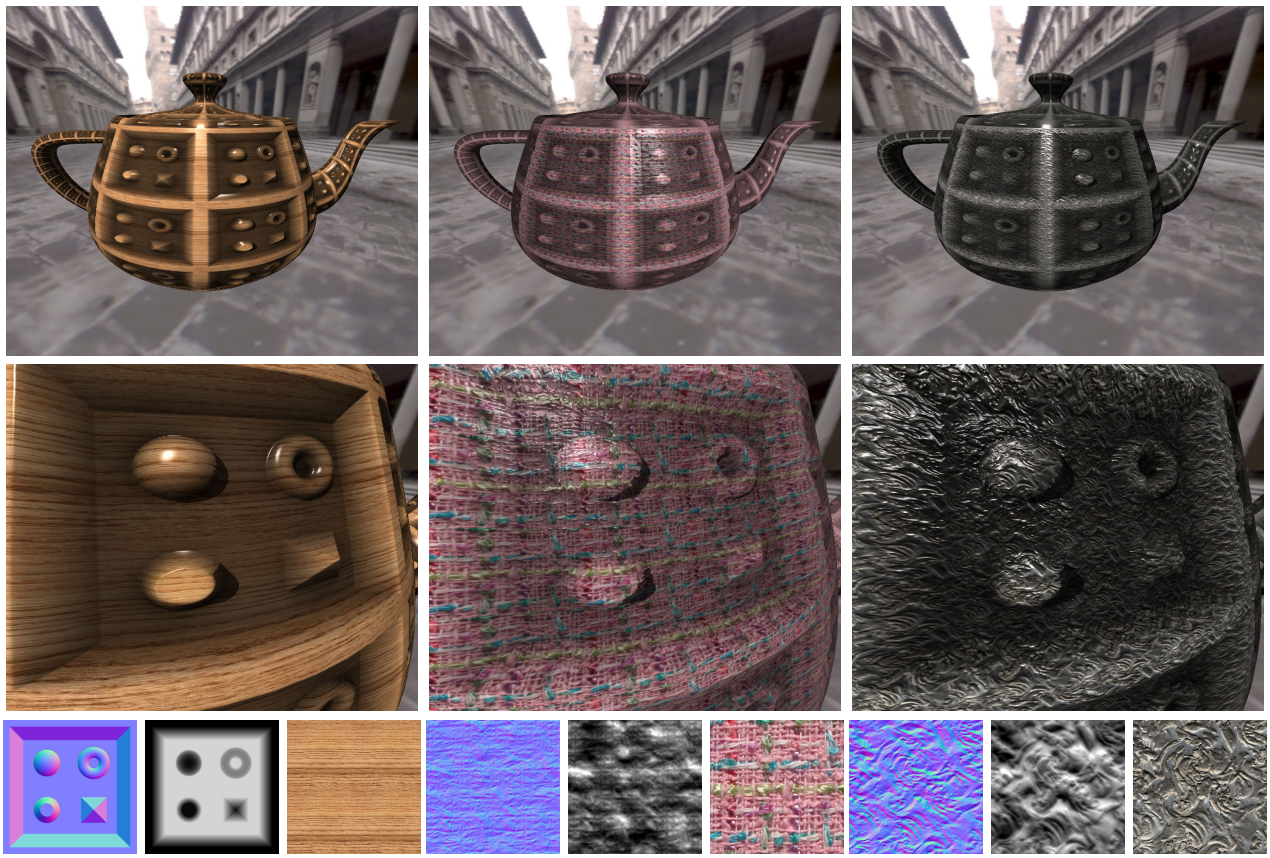

Fig. 1: Comparison between conventional and meta-relief mapping renderings. (a) Relief rendering with ambient occlusion. (b) Meta-relief rendering of a fabric texture on top of the relief shown in (a). Note the finer details introduced by the fabric (middle row). (c) Meta-relief rendering of a metal texture on top of the relief shown in (a). The details of the metal texture follow the base shape (middle row). The last row shows, respectively, the normal, depth, and color maps used for rendering.

***Abstract*—We present an efficient technique for modeling and rendering complex surface details defined at multiple scales. Conceptually, meta-relief texture mapping can be described as recursively mapping finer relief-texture layers on top of coarser ones. Such a factorization has several desirable properties. For instance, it provides a way of simulating highly-complex surface details as a combination of simpler and inexpensive image-based representations. This greatly simplifies the modeling of surface details, enhancing the artists' expressive power. We also introduce a dynamic texture-space ambient-occlusion technique for relief mapping, which greatly improves the quality of relief renderings. We demonstrate the effectiveness of these techniques by creating and rendering a number of meta-relief textures with complex surface details which would have been hard to model directly.**

***Keywords*- Meta-relief mapping; Multiscale surface details; Real-time rendering; Texture-space ambient occlusion;**

I. Introduction

Light interaction with fine surface details produce subtle shading effects, playing an important role in our perception of realism. However, modeling such details using polygonal

representations tend to be cumbersome and often impractical. For this reason, several texture-mapping techniques have been proposed to simulate the existence of fine surface details. This has been achieved, for instance, by changing surface normals [1], modifying the actual geometry [2], performing texture warping prior to mapping [3], or by ray casting in 2-D texture space [4], [5]. While very popular and effective in many practical scenarios, these techniques have their own limitations. For instance, normal mapping [1] cannot handle self-occlusions, self-shadowing, or render object silhouettes. These restrictions can be eliminated with the use of displacement mapping [2], but the technique requires rendering a large number of micropolygons, precluding its use in real-time applications. Current relief-mapping techniques [4], [5], [6], [7] can represent both fine and coarse surface details, but combining both often requires considerable modeling effort.

We introduce efficient extensions to relief texture mapping that greatly simplify the modeling of complex surface details and improve their rendering quality. Conceptually, our technique can be described as *recursively mapping finer relief texture layers on top of coarser ones*. We call our approach *meta-relief texture mapping*. It provides great flexibility, allowing the combination of details at an arbitrary number of scales, and treating each relief-texture scale independent of each other. This allows for their reuse and recombination, lending to diverse and rich texture effects. We also introduce a texture-space ambient-occlusion technique for relief mapping, which dynamically adjusts itself to relief-depth rescaling. By generating soft shadows, it significantly improves the quality of relief-mapping renderings.

Figure 1 illustrates some results obtained with our approach. Figure 1 (a) shows a teapot rendered using conventional relief texture mapping with ambient occlusion. Figure 1 (b) shows the result of applying a fabric relief texture on top of the rendered teapot shown in (a). Note the finer scale details of the fabric. Figure 1 (c) shows another example mapping fine metallic details to the teapot shown in Figure 1 (a).

The **contributions** of our work include:

- An efficient texture-mapping technique for rendering complex surface details defined at multiple scales (Section III). Our approach provides great flexibility and simplifies the modeling task;
- A texture-space ambient-occlusion technique that significantly improves the realism of relief-mapping renderings (Section IV).

II. Related Work

The idea of using textures to render approximate representations of surface details has a long history in computer graphics. Normal mapping [1] simulates the appearance of surface details by changing the surface's normal field, while displacement mapping [2] uses a large number of micropolygons to modify the object's actual geometry. Several techniques have been proposed to accelerate displacement mapping. They are based on variety of strategies, including ray tracing [8], [9], [10], [11], 3-D texture mapping [12], [13], and 3-D inverse image warping [14]. Other techniques use a surface shell layer and volumetric textures to render non-height-field details [15], [16], [17]. Wang et al. [18], [19] use 5-D data structures to store view-dependent samples of a displacement map. In contrast to these techniques, ours is based on relief mapping [4] and can render complex surface details in real time, while having very small memory footprint and supporting close-up views.

BTFs: Bidirectional Texture Functions [20] are 6-dimensional texture representations that account for illumination and viewing-dependent effects. They can be used to realistically render small mesostructure details, but each BTF requires a large number (of the order of thousands) of aligned photographs. The acquisition process often takes several hours, and the resulting dataset can reach over a hundred megabytes, requiring efficient compression schemes. Unlike our technique, BTFs cannot be combined to produce new surface details, or affect the object's silhouette. Müller et al. [21] provides an in-depth discussion of the acquisition, compression, and rendering of BTFs.

Relief Mapping: Relief-mapping techniques [4], [5], [6], [7] implement fragment-driven solutions to relief texture mapping [3]. The information required for rendering surface details on arbitrary polygonal models is stored in RGBα textures containing a *normal map* (RGB channels) and a *depth map* (α channel). Such data can be used with any color texture. For the rendering of each fragment, a ray-height-field intersection is performed in texture space. A local illumination model can be applied and self-shadowing can be computed in a straightforward way. Building upon relief mapping, our approach simplifies the modeling and rendering of complex surface details by combining information from multiple scales.

Ambient Occlusion: Ambient Occlusion (AO) approximates global-illumination effects by estimating, at any surface point, the amount of incoming light from all directions [22], [23]. *Screen-Space Ambient Occlusion* (SSAO) computes an approximation to AO directly in screen space, on a per-frame basis [24], [25]. For this, it uses the depth buffer to evaluate the occlusion of each sample according to its neighboring pixels. Since our ambient-occlusion technique for relief textures performs AO computation based on a depth map, it is similar in spirit to screen-space ambient occlusion. Unlike SSAO, however, our occlusion maps are pre-computed. Nevertheless, our technique supports dynamic changes in visibility resulting from rescaling the relief depth, and handles tileable textures.

Frequency Decomposition: A class of techniques decompose a surface or image into low- and high-frequency components. Such decompositions have been used, for instance, to produce exaggerated shading [26], image-detail enhancement [27], and to extract approximate depth from single images [28]. Contrary to these techniques, ours simplifies the task of modeling and rendering complex surface details, creating photorealistic depictions of 3-D objects in real time.

III. Meta-Relief Texture Mapping

Meta-relief texture mapping (MRTM) simplifies the process of modeling complex surface details using a combination of simpler and inexpensive image-based representations at multiple scales. It supports a number of independent layers (scales) and provides interactive control over the contributions (weights) of the different layers. The independence among layers allows for their reuse and recombination, leading to the creation of diverse and rich texture effects.

MRTM is fully compatible with existing relief mapping techniques [4], [5], [6], [7], which can be instantly modified to support meta-relief texture mapping. As such, MRTM can by applied to arbitrary polygonal models in real time, has low-memory requirements, handles self-occlusions and self-shadowing, and can be completely implemented on GPUs.

A. Meta-Relief Textures (MRTs)

Having finer relief details recursively mapped on top of coarser ones is equivalent to adding the contributions of the individual relief textures at different scales. This surprisingly simple operation can be used to model highly-complex structures. For simplicity, we illustrate the process of combining detail information at two scales, and using the same resolution and tiling factor for both. The generalization for an arbitrary number of scales, possibly using different resolutions and tiling factors, is immediate and given by

$$MRT_{normal} = \frac{\sum_{i=1}^{s} w_{ni} \times normal_i}{\sum_{i=1}^{s} w_{ni}} \quad (1)$$

and

$$MRT_{depth} = \frac{\sum_{i=1}^{s} w_{di} \times depth_i}{\sum_{i=1}^{s} w_{di}}, \quad (2)$$

where MRT_{normal} and MRT_{depth} are the resulting meta-relief texture normal and depth maps, s is the number of scales, w_{ni} and w_{di} are the non-negative weights associated to the i-th normal and depth layers, respectively. Section V shows images rendered using details at more than two scales.

Figures 2 and 3 illustrate the normal and depth maps, as well as the process, used to create the meta-relief textures mapped to the teapots shown in Figures 1 (b) and (c). Figures 2 (left) and 3 (left) are, respectively, the normal and depth maps associated to the regular (single scale) relief texture shown in Figure 1 (a). We will refer to it as the *Base* relief texture, as it represents coarser surface details.

The *Shaped Fabric* meta-relief texture used to render Figure 1 (b) was obtained combining the *Base* + *Fabric* normal and depth maps shown in Figures 2 and 3, respectively. In this case, the numerators of Eqs. 1 and 2 were, respectively, *Shaped Fabric Normals* = $w_{n1}\times$*Base Normals* $+w_{n2}\times$ *Fabric Normals*, and *Shaped Fabric Depth* = $w_{d1}\times$*Base Depth* $+w_{d2}\times$ *Fabric Depth* (Figures 2 and 3). To enforce that the original fabric relief only contributes fine scale details, we used $w_{n1} = w_{d1} = 1.0$ and $w_{n2} = w_{d2} = 0.3$. By (interactively and independently) changing these weights, one can achieve a variety of texture effects. The fabric color texture (Figure 1 (b), bottom row) was used for coloring the obtained meta-relief texture. A similar procedure was used to create the *Shaped Metal* meta-relief texture used for rendering Figure 1 (c). Its color texture is shown in the bottom row of Figure 1 (c). Given a meta-relief texture, it can be rendered using any of the algorithms described in [4], [6], [7].

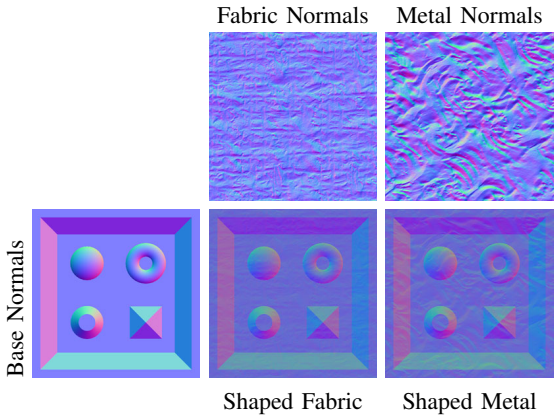

Fig. 2: Examples of normal maps of the two meta-relief textures (*Shaped Fabric* and *Shaped Metal*) used to render the teapots shown in Figures 1 (b) and (c). Their corresponding depth maps are shown in Figure 3.

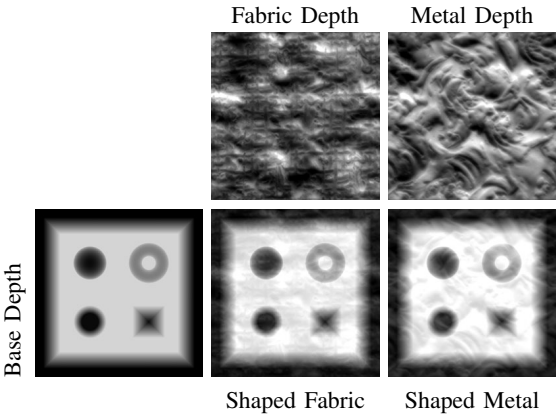

Fig. 3: Examples of depth maps of the two meta-relief textures (*Shaped Fabric* and *Shaped Metal*) used to render the teapots shown in Figures 1 (b) and (c), and whose normal maps are shown in Figure 2.

IV. Ambient Occlusion for Relief Textures

Ambient occlusion (AO) can significantly improve the realism of a scene by emphasizing small-scale details and introducing soft shadows [29]. To improve the visual quality of meta-relief renderings and highlight its benefits, we have developed a *texture-space ambient-occlusion* (TSAO) technique for relief textures. The resulting AO map can be saved as a separate texture, or stored in the blue channel of the (meta-)relief texture. In this case, the R, G, B, and α channels store the X and Y components of the normal, the AO map, and the depth map, respectively. The normal's Z component

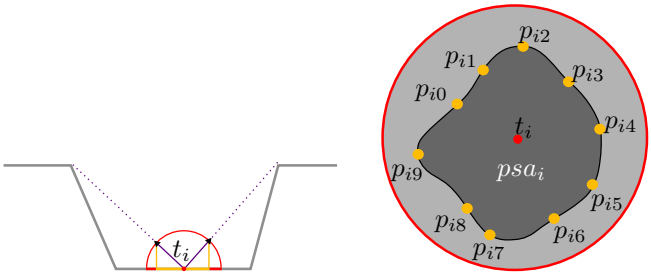

Fig. 4: Computing a discrete version of the projected solid angle corresponding to the visible portion of the unit hemisphere above t_i. (left) A cross-section of the solid angle with vertex at t_i. The solid gray line corresponds to the geometry represented in the depth map. (right) Top view of the discrete projected solid angle on a unit hemisphere. p_{ij} is the intersection of the j-th edge of the generalized cone with the unit hemisphere.

is recovered as $Z = \sqrt{1-(X^2+Y^2)}$. Figure 5 shows an AO map computed for relief textures.

Meta-relief texture mapping naturally enables the use of textures with different resolutions and tiling factors. All textures are loaded on the GPU and sampled according to their resolutions. Baking the textures as a preprocessing would make it more difficult to combine multiple layers with different resolutions and tiling factors. Figure 8 shows examples of meta-relief mapping using textures with different resolutions and tiling factors.

A. Occlusion Map Generation

We obtain AO maps by computing, for each texel t_i in the depth map, the ratio psa_i/π: a discrete version of the projected solid angle corresponding to the visible portion of the hemisphere above t_i. To obtain psa_i, we first compute a discrete version of the generalized cone corresponding to this solid angle and intersect it with a unit hemisphere centered at t_i (Figure 4, left). For creating the discrete version of the generalized cone, we use 50 directions (taken at regular intervals of 7.2 degrees). Along each direction, visibility is computed in a way similar to horizon mapping [30], but using a toroidal topology, since we use tileable textures. Let p_{ij} be the intersection of the j-th generalized cone edge with the unit hemisphere centered at t_i. Such intersection points are orthographically projected onto the (s,t) texture plane defining a closed polygon. The corresponding AO value at t_i is then

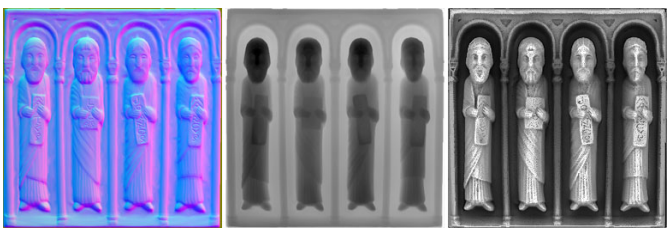

Fig. 5: Ambient-occlusion map for relief textures. (left) Normal map. (center) Depth map. (right) AO map.

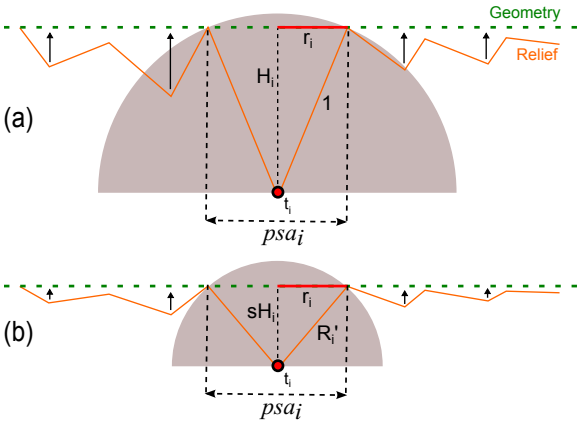

Fig. 6: Updating ambient-occlusion values after depth rescaling. *Geometry* refers to the 3-D model geometry onto which the relief texture (*Relief*) has been mapped. (a) A cross-section of solid angle for a texel t_i, represented using a unit hemisphere. (b) The cross-section of the solid angle at t_i after rescaling the relief depth by a factor s, shown on a hemisphere with radius R_i'.

computed dividing psa_i (the area of the resulting polygon, shown in Figure 4, right) by π (the area of the circle at the base of the unit hemisphere).

The computation of AO maps is performed off-line using a GLSL shader, which produces a discrete map. Like in other ambient-occlusion techniques [31], [32], we apply a low-pass filter at the end of the estimation process to smooth the resulting map. The low-pass filter is only applied at texels with some degree of occlusion, to avoid introducing occlusions where they do not exist. Figure 5 shows an AO map computed for a relief texture using this technique. As expected, AO values are lower at more occluded regions.

AO maps for meta-relief textures are created by combining the AO maps of the individual relief textures. This can be performed in two ways, providing slightly different results. One alternative is to multiply the AO values (visibility) of corresponding texels. The other consists in selecting the smallest value between corresponding texels. The images shown in the paper were rendered using the first alternative.

Since relief textures support dynamic depth scaling [4], the pre-computed AO maps should support it as well. Recall that considering a unit hemisphere (Figure 6(a)), the AO value for a texel t_i is computed as

$$AO(t_i) = \frac{psa_i}{\pi}. \tag{3}$$

One can approximate psa_i by the area of the circle with radius r_i: $psa_i \approx \pi r_i^2$ (Figure 6(a)). Substituting this in Equation 3:

$$r_i = \sqrt{AO(t_i)}. \tag{4}$$

Depth scaling does not affect the size of the texture nor r_i, but only depth values (Figure 6(b)). Thus, after depth scaling, H_i becomes sH_i, and the new hemisphere radius R_i' is given by

$$(R_i')^2 = (sH_i)^2 + r_i^2, \tag{5}$$

where $H_i = \sqrt{1^2 - r_i^2}$, and s is the depth scaling factor. Since the new AO value at t_i is computed dividing psa_i by the area of the circle with radius R'_i, it can be expressed in terms of its original AO value as:

$$AO_{new}(t_i) = \frac{psa_i}{\pi(R'_i)^2} = \frac{AO(t_i)}{(R'_i)^2}. \quad (6)$$

Figure 7 compares relief renderings produced without (left) and with TSAO (right). Note the soft shadows obtained with our ambient-occlusion technique.

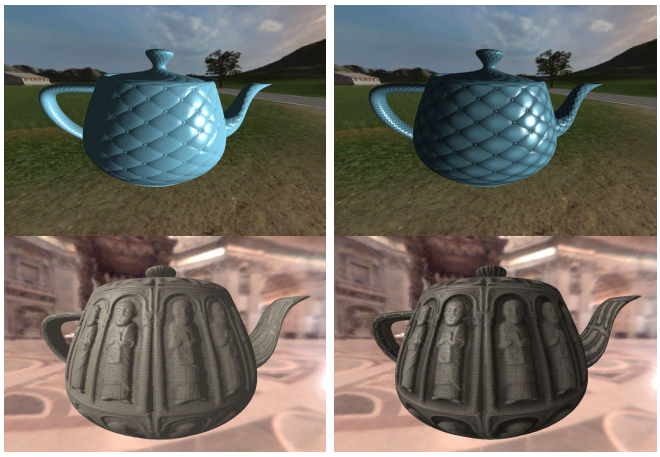

Fig. 7: Texture AO rendering. (left) Teapot rendered using a regular relief texture with shadows. (right) Same teapot rendered using our texture-based ambient occlusion.

V. Results

We have implemented the techniques described in the previous sections on Nvidia's FX Composer, and using Unity [33]. We used them to create a large number of meta-relief renderings with ambient occlusion. For this, we combined many regular relief textures, some of which are shown in Figure 10. Regular relief textures can be conveniently created directly from color maps. For this, one can use any of the available tools for generating normal and depth maps from images. The normal and depth maps for textures *Cushion*, *Mini Bricks*, *Metal*, *Fabric*, and *Stone Bricks* (Figure 10) were obtained directly from their corresponding color textures using the software CrazyBump [34]. The color map for the *Jeans* texture was created using the technique described in [35], and its associated normal and depth maps were created with our own MATLAB script.

Figure 1 compares meta-relief renderings against regular relief renderings (with our texture-based ambient-occlusion technique). Figure 8 illustrates meta-relief renderings containing layers with different resolutions and tilling factors. Both images combine a 1024×1024 *Base* texture with a 512×512 *Metal* detail texture. The tiling factors applied to *Metal* on the left and on the right images are 5.5 and 14.5, respectively.

Figure 9 shows a teapot rendered with a meta-relief texture that combines the *Priests* and *Mini Bricks* relief textures from Figure 10. On the right, we show a close-up view of a portion of the teapot on the left. Note the mini bricks shaped according to the geometry of one priest, as well as the soft shadows due to ambient occlusion.

Figure 11 shows meta-relief renderings of a cube created combining the *Base* relief texture with five others shown in Figure 10: *Mini Bricks*, *Metal*, *Jeans*, *Fabric*, and *Stone Bricks*. Note the rich diversity of effects, ranging from cloth, to metal, to stones, where the overall shape of the *Base* texture has been preserved. Again, the soft shadows are due to ambient occlusion. The second and third rows show close-up views of the simulated geometry, and reveal the surface details obtained from the combined textures.

Figure 12 shows another set of meta-relief renderings for a teapot. In these examples, the meta-relief textures were obtained combining the *Cushion* relief texture with the same group of five textures used to provide the finer details in Figure 11. Note the importance of the soft shadows created by ambient occlusion for the quality of the final renderings. The close-up views in the third row clearly show the contributions of both textures for the resulting surface details.

Figure 13 shows examples of meta-relief textures combining details at three different scales. The image on the left combines the following relief textures shown in Figure 10 (from coarse to fine scales): *Base*, *Cushion*, and *Jeans*. The result shown

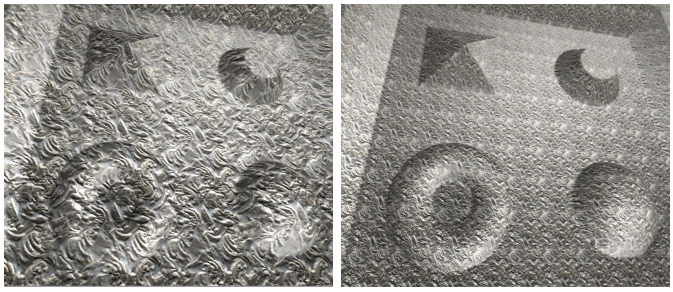

Fig. 8: Example of Meta-relief mapping rendering using different resolutions (*Base* (1024 × 1024) and *Metal* (512 × 512) and fractional tile factors (*Metal* tile factor 5.5 (left) and 14.5 (right)).

Fig. 9: Meta-relief rendering of a teapot with ambient occlusion. The meta-relief combines the *Priests* and *Mini Bricks* textures from Figure 10. (left) Teapot. (right) Close-up showing a combination of meta-relief rendering and ambient occlusion.

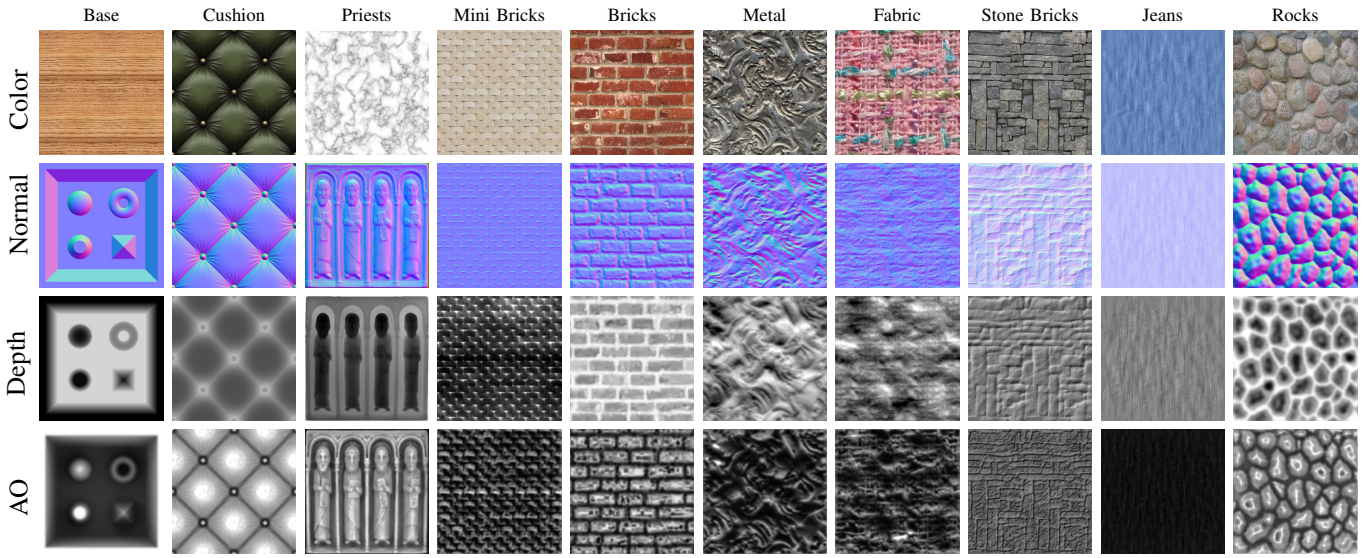

Fig. 10: Various relief textures with corresponding AO maps used to create the meta-relief textures shown in the paper.

in the center combines *Base*, *Mini Bricks*, and *Jeans*, while the image on the right was obtained combining *Base*, *Stone Bricks*, and *Metal*. Note the details at all scales.

A. Discussion

High-quality renderings can be obtained even with very small meta-relief textures, as illustrated in Figure 14. The teapots shown in (a) were rendered using full 256×256 meta-relief textures (MRTs), *i.e.*, color, normal, depth, and ambient-occlusion maps all have only 256×256 texels. The images shown in (b) were rendered using considerably higher-resolution: full 1024×1024 MRTs (top), and full 600×600 MRTs (bottom). These images are visually indistinguishable from their corresponding ones in (a). Column (c) shows close-up views of the two teapots rendered with full 256×256 MRTs. Column (d) shows close-ups of the same objects using MRTs with 256×256 normal, depth, and AO maps, but using higher resolution color maps (1024×1024 for the top row, and 600×600 for the bottom row). Column (e) presents close-ups of these objects using full 1024×1024 MRTs (top), and full 600×600 MRTs (bottom).

At the extreme zoom levels used in Figures 14 (c)-(e), a direct comparison of images in columns (c) and (e) reveals that highlights are properly positioned in (c). Such a comparison also reveals that the images in (c) are slightly blurrier than the ones in (e), while the fabric image in (c) looks slightly brighter. The blurring and extra brightness result from the use of lower-resolution color and ambient-occlusion maps, respectively. A good compromise to handle extreme close-up views is to use low-resolution normal, depth, and ambient-occlusion maps, while using high-resolution color maps. This situation, which eliminates the blurriness although does not fix the additional brightness, is illustrated in column (d). It can be improved further using a higher-resolution ambient-occlusion map.

For the combinations of scale sizes and tilling factors that allow for the meta-relief texture to be baked into a single texture, the rendering space and time complexity become exactly the same as for the rendering of regular relief textures.

VI. Conclusion

We have introduced extensions to relief texture mapping that greatly simplify the modeling of complex surface details and improve their rendering quality. Meta-relief texture mapping has several desirable properties. It provides a way of simulating highly-complex surface details by combining simpler and inexpensive image-based representations. By treating layers independently, it allows for their reuse and recombination. This gives artists significant expressive power and is fully compatible with previous relief-mapping techniques. We have also presented a dynamic texture-space ambient-occlusion technique that significantly improves the quality of relief-mapping renderings.

We have demonstrated the effectiveness of these techniques by modeling and rendering a variety of complex surface details, which would have been hard to model otherwise. We have shown that meta-relief renderings properly combine details at various scales, while the use of ambient occlusion greatly improves the realism of 3-D scenes.

Acknowledgments

This work was sponsored by CNPq (grants 131002/2012-0, 482271/2012-4 and 306196/2014-0) and CAPES.

References

[1] J. F. Blinn, "Simulation of wrinkled surfaces," in *Proc. SIGGRAPH 1978*, 1978, pp. 286–292.

[2] R. L. Cook, "Shade trees," in *Proceedings of SIGGRAPH*. ACM Press, 1984, pp. 223–231.

[3] M. M. Oliveira, G. Bishop, and D. McAllister, "Relief texture mapping," in *Proc. SIGGRAPH 2000*, pp. 359–368.

[4] F. Policarpo, M. M. Oliveira, and J. Comba, "Real-time relief mapping on arbitrary polygonal surfaces," in *Proc. ACM Symposium on Interactive 3D Graphics and Games*, 2005, pp. 155–162.

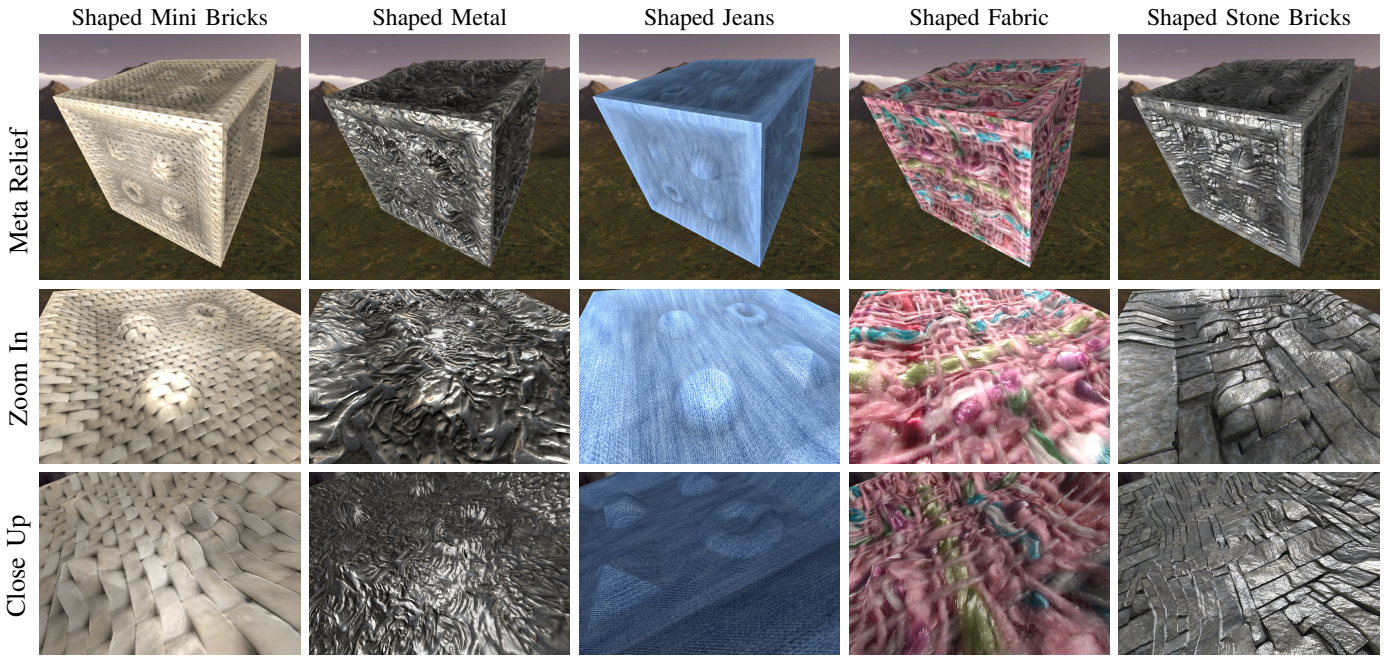

Fig. 11: Meta-relief renderings created by combining the *Base* relief texture with five others shown in Figure 10: *Mini Bricks*, *Metal*, *Jeans*, *Fabric*, and *Stone Bricks*. (second row) Closer view of one of the faces of the cubes shown on top. (third row) Close-up of the geometric elements represented in the *Base* texture.

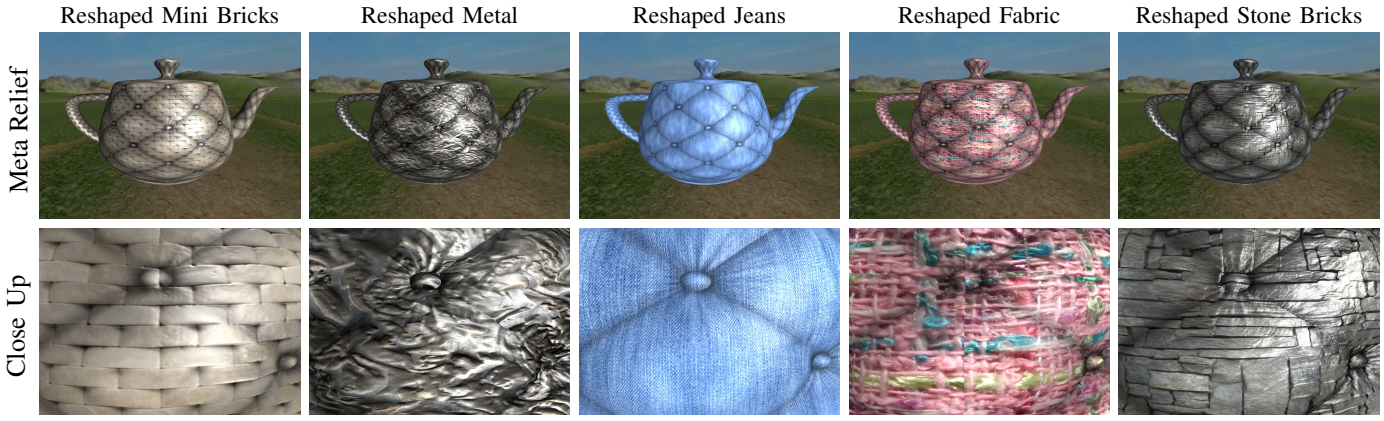

Fig. 12: Meta-relief renderings of a teapot obtained by combining the *Cushion* relief texture with five others shown in Figure 10: *Mini Bricks*, *Metal*, *Jeans*, *Fabric*, and *Stone Bricks*. (second row) Close-up showing that the renderings of the meta-relief textures clearly exhibit surface details from the combined textures. Soft shadows created with our ambient occlusion technique.

[5] N. Tatarchuk, "Dynamic parallax occlusion mapping with approximate soft shadows," in *Proc. ACM Symposium on Interactive 3D Graphics and Games*, 2006, pp. 63–69.

[6] M. M. Oliveira and F. Policarpo, "An efficient representation for surface details," Federal University of Rio Grande do Sul - UFRGS, http://www.inf.ufrgs.br/∼oliveira/pubs_files/Oliveira_ Policarpo_RP-351_Jan_2005.pdf, Tech. Rep. RP-351, January 2005.

[7] F. Policarpo and M. M. Oliveira, "Relief mapping of non-height-field surface details," in *Proc. ACM Symposium on Interactive 3D Graphics and Games*, 2006, pp. 55–62.

[8] J. Patterson, S. Hoggar, and J. Logie, "Inverse displacement mapping," *Comp. Graphics Forum*, vol. 10, no. 2, pp. 129–139, 1991.

[9] M. Pharr and P. Hanrahan, "Geometry caching for ray-tracing displacement maps," in *Eurographics Rendering Workshop*, 1996, pp. 31–40.

[10] W. Heidrich and H.-P. Seidel, "Ray-tracing procedural displacement shaders," in *Graphics Interface*, 1998, pp. 8–16.

[11] B. E. Smits, P. Shirley, and M. M. Stark, "Direct ray tracing of displacement mapped triangles," in *Proc. Eurographics Workshop on Rendering Techniques 2000*, 2000, pp. 307–318.

[12] A. Meyer and F. Neyret, "Interactive volumetric textures," in *Eurographics Rendering Workshop 1998*, 1998, pp. 157–168.

[13] J. Kautz and H.-P. Seidel, "Hardware accelerated displacement mapping for image based rendering," in *Graphics Interface*, 2001, pp. 61–70.

[14] G. Schaufler and M. Priglinger, "Efficient displacement mapping by image warping," in *EG Rendering Workshop 1998*, pp. 175–186.

[15] Y. Chen, X. Tong, J. Wang, S. Lin, B. Guo, and H.-Y. Shum, "Shell texture functions," *ACM TOG*, vol. 23, no. 3, pp. 343–352, August 2004.

[16] J. Peng, D. Kristjansson, and D. Zorin, "Interactive modeling of topologically complex geometric detail," *ACM Trans. Graphics*, vol. 23, no. 3, pp. 635–643, August 2004.

[17] S. Porumbescu, B. Budge, L. Feng, and K. Joy, "Shell maps," *ACM TOG*, vol. 24, no. 3, pp. 626–633, 2005.

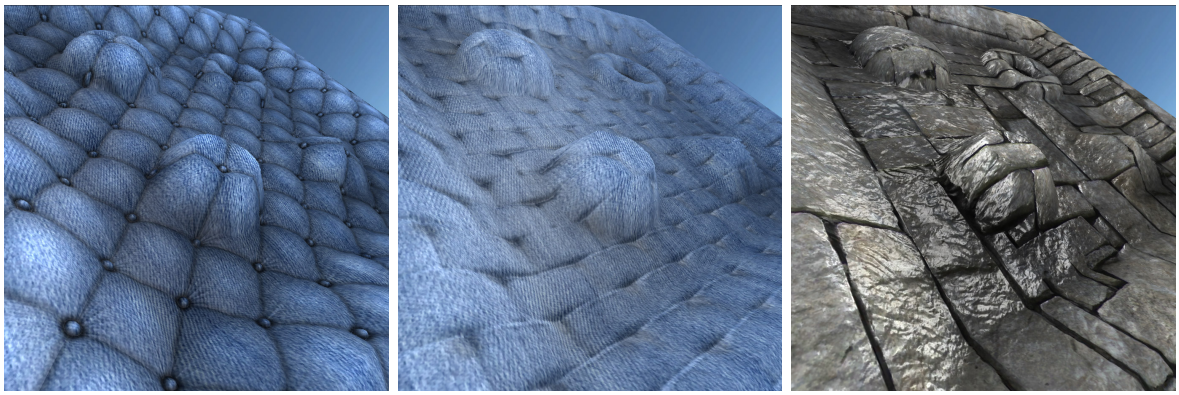

Fig. 13: Examples of meta-relief renderings containing details at three scales. (left) Base + Cushion + Jeans. (center) Base + Mini Bricks + Jeans. (right) Base + Stone Bricks + Metal. Note how the details at the various scales are clearly visible.

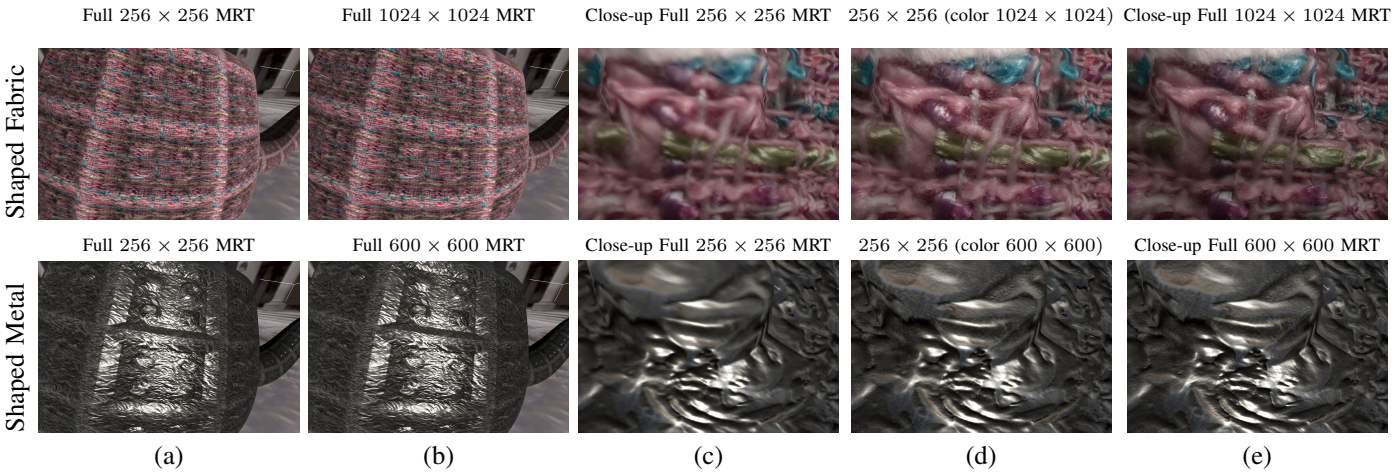

Fig. 14: Rendering quality obtained using meta-relief textures (MRTs) with different resolutions. (a) Images rendered using full 256×256 MRTs (i.e., color, normal, depth, and AO maps all have 256 texels). (b) Images rendered using higher-resolution MRTs: full 1024×1024 MRTs (top) and full 600×600 MRTs (bottom). The corresponding images in (a) and (b) are visually indistinguishable. (c) Close-ups using full 256×256 MRTs. (d) Close-ups using MRTs with 256×256 normal, depth, and AO maps, but higher resolution color maps. (e) Close-ups using full 1024×1024 MRTs (top), and full 600×600 MRTs (bottom).

[18] L. Wang, X. Wang, X. Tong, S. Lin, S. Hu, B. Guo, and H.-Y. Shum, "View-dependent displacement mapping," *ACM Trans. Graph.*, vol. 22, no. 3, pp. 334–339, 2003.

[19] X. Wang, X. Tong, S. Lin, S. Hu, B. Guo, and H.-Y. Shum, "Generalized displacement maps," in *Proc. EGSR*, 2004, pp. 227–233.

[20] K. J. Dana, B. van Ginneken, S. K. Nayar, and J. J. Koenderink, "Reflectance and texture of real-world surfaces," *ACM Trans. Graph.*, vol. 18, no. 1, pp. 1–34, Jan. 1999.

[21] G. Müller, J. Meseth, M. Sattler, R. Sarlette, and R. Klein, "Acquisition, synthesis and rendering of bidirectional texture functions," in *Eurographics 2004, State of the Art Reports*, Sep. 2004, pp. 69–94.

[22] S. Zhukov, A. Inoes, and G. Kronin, "An ambient light illumination model," in *Rendering Techniques '98*, 1998, pp. 45–56.

[23] H. Landis, "Production-ready global illumination. Course 16 Notes. SIGGRAPH 2002."

[24] T. Luft, C. Colditz, and O. Deussen, "Image enhancement by unsharp masking the depth buffer," *ACM Trans. Graph.*, vol. 25, no. 3, pp. 1206–1213, Jul. 2006.

[25] V. Kajalin, "Screen space ambient occlusion," in *ShaderX7*. Charles River Media, March 2009, pp. 413–424.

[26] S. Rusinkiewicz, M. Burns, and D. DeCarlo, "Exaggerated shading for depicting shape and detail," *ACM Trans. Graph.*, vol. 25, no. 3, pp. 1199–1205, 2006.

[27] R. Fattal, "Edge-avoiding wavelets and their applications," *ACM Trans. Graph.*, vol. 28, no. 3, pp. 22:1–22:10, 2009.

[28] J. Lopez-Moreno, J. Jimenez, S. Hadap, E. Reinhard, K. Anjyo, and D. Gutierrez, "Stylized depiction of images based on depth perception," in *Proc. of NPAR '10*, 2010, pp. 109–118.

[29] J. Hoberock and Y. Jia, "High-quality ambient occlusion," in *GPU Gems 3*. Addison Wesley, March 2008, pp. 257–274.

[30] N. Max, "Horizon mapping," *The Visual Computer*, vol. 4, pp. 109–117, 1988.

[31] J. Kontkanen and S. Laine, "Ambient occlusion fields," in *Proc. ACM Symposium on Interactive 3D graphics and Games*, 2005, pp. 41–48.

[32] L. Szirmay-Kalos, T. Umenhoffer, B. Toth, L. Szecsi, and M. Casasayas, "Volumetric ambient occlusion," *Computer Graphics and Applications, IEEE*, vol. PP, no. 99, p. 1, 2009.

[33] Unity, "Unity - game engine," 2015. [Online]. Available: https://unity3d.com/

[34] CrazyBump, "Demo version 1.2," Oct. 2013. [Online]. Available: http://www.crazybump.com/

[35] B. Galerne, A. Lagae, S. Lefebvre, and G. Drettakis, "Gabor noise by example," *ACM TOG*, vol. 31, no. 4, pp. 73:1–73:9, Jul. 2012.