RGBN image editing

Thiago Pereira Luiz Velho
Visgraf Laboratory
IMPA - Instituto de Matematica Pura e Aplicada
Rio de Janeiro, Brazil
tpereira@impa.br lvelho@impa.br

***Abstract*—We propose a method to edit RGBNs (images with a color and a normal channel). High resolution RGBNs are easy to obtain using photometric stereo. Free editing will result in normals which do not correspond to any realizable surface. Our normal operators guarantee the integrability of the results.**

Our method can filter normals with any linear kernel allowing high-pass and edge-enhancement filters. We have designed a normal linear combination method for adding details with frequency control. New geometric features can be created with a custom brush that warps deformations along its path. A nonlinear operator is also proposed. Its integrability is controlled with a two-band separation of frequencies: smooth shape and details.

***Keywords*-RGBN, normal map, gradient reconstruction, normal processing, filtering, deformation**

I. INTRODUCTION

This paper proposes new methods to process normal maps that guarantee the integrability of the results. Normal maps evolved from the seminal work of Blinn in bump mapping [1]. Beyond the applications in real-time rendering, normals are extremely important in geometric modelling. In Shape Palletes [2], the authors sketch 3D models specifying only normals.

Normal maps can be obtained in many ways. A high-resolution modeled mesh can be used to extract it and enhance models with fewer polygons [3], [4]. An alternative is capturing real world objects. Capturing normals with 3D scanning requires expensive equipment and lacks resolution. Much cheaper digital cameras can be used to obtain better results with stereophotometric approaches [5] allowing very high resolution normals, as in the digitization of Michelangelo's Pieta [6]. It takes as input multiple images from the same view point, each illuminated with different known light positions. It then solves a least-squares problem to find the normal and albedo (color) of each pixel in the image. In many applications, normals are first obtained from raw data and then undergo an integration process to obtain a mesh with positions. The integration step is unstable, as Nehab et al [7] point out, reconstructing geometry from normals brings low frequency errors. For this reason, we have designed a tool that edits the normals directly and completely avoids reconstruction.

Note that editing normals is semantically different from editing colors. Normals are unit vectors and should always represent a surface. Free editing will result in a normal field which does not correspond to any realizable surface. This is why editing normals is a hard problem. Our main contribution is controlled operations that guarantee integrability, i.e. the existence of a smooth surface with the given normals. Editing general normal maps in an image editing software is difficult, because normals are restricted to the unit sphere. Also due to distortions introduced by parametrizations which are usually hard to comprehend by the user and by the system.

Advances in vision like shape from shading and stereophotometric [5] have made RGBNs an important normal map representation. We restrict our work to RGBNs [8], which are camera pictures containing for each pixel color (RGB) but also geometric information (normal). An RGBN is a projective mapping of a 3D object into the texture space. With this parametrization, we propose simple integrability-preserving linear operators on RGBNs, such as filtering and adding details. We also introduce a nonlinear detail operator which reduces bias towards the camera. The nonlinear operator results can be controlled with a two-band separation of the signal: low frequencies are preserved and only details are edited.

Projective mappings have the advantage that users are well acquainted to projections, as such, editing becomes more intuitive. On the other hand, RGBNs are limited in that they do not allow for change of viewpoint or realistic shadow calculations. Both these limitations are a consequence of having only a local description of the model (normals) instead of a global one (positions). A projective RGBN atlas [9], [6] can counter this limitations. The resulting charts are surfaces mapped by a camera transformation to the plane image. Since they contain normal and colors, each chart is in fact an RGBN and can be edited by our system.

The paper is organized as follows. In section III, we propose a method for filtering normals allowing smoothing, edge enhancement and high pass filters. In section IV we introduce the linear combination method for adding details.

Subsection V defines the creation of features by twisting some known detail along a user-defined path. In section VI we determine when edited normals correspond to a realizable surface. In section VII, we also propose a nonlinear operator to add details.

II. Related Work

Toler-Franklin et al. [8] coined the term RGBN to refer to an array of pixels with associated color and normal channels. They have shown that many NPR rendering algorithms work in RGBNs. These include toon shading, line drawing methods, curvature shading and exaggerated shading. They developed signal processing techniques like low-pass filtering, derivatives and curvature estimation.

ZBrush [10] is a digital sculpting tool. It can create high resolution models using subdivision. They can then be painted/sculpted using 2.5D images of pixols which contain colors and depth. Normal maps can be generated from the final models. For editing normals, they would first have to be converted to depth information outside ZBrush.

In [11], [12] the authors recovered normals from photographs and then developed texture synthesis on the resulting RGBNs. This way the normals are used to guide local distortions in the synthesized texture. In Textureshop [12], the authors also transferred normals between images. Poisson image editing [13] was used to merge normals seamlessly, followed by normalization, a process which does not guarantee a conservative normal field.

Normalpaint [14] proposes a tool for creating normal maps directly, thus avoiding the need to model a high-res mesh in the traditional modeling pipeline. They calculate normals in a way that is equivalent to height extrusion. In their approach only shapes with simple geometry can be created and editing of normals is not supported.

In Gradient Domain Painting [15], the authors propose tools that change the gradient of color images, a GPU-multigrid integrator recovers the new image in real-time. Different gradient blending modes are shown. In our work, the entire pipeline is composed of local operations.

In [15], Taubin defines a specialized laplacian operator for filtering normals on a mesh. He then integrates this normal field to obtain filtered positions. Different smoothing filter for normal maps have been developed for mipmapping. In [16], the author smooths normals and shows how the shortening they introduce can be used to reduce aliasing of specular highlights. In [17], the authors formalize normal map filtering using convolution between normal distribution functions and the BRDF.

III. Filtering

The simple way to filter an RGBN is to consider each channel of a 6D (color + normal) image separately and convolve it with a kernel. Since the resulting normals would not have unit norm, a normalization step would follow. As Toler-Franklin et al [8] pointed out the problem with naive filtering is that due to foreshortening the area of each pixel will be underestimated by $\cos\theta$, θ the angle being between normal and viewing directions. To simplify the analysis the authors assume a constant viewing direction as in the case of a far away viewer. In this case the viewing direction is the z direction and $\cos\theta = n_z$. This analysis means we should replace the normal vector (n_1, n_2, n_3) by $(n_1/n_3, n_2/n_3, 1)$, which we call foreshorten corrected. In this representation, filtering is now a linear operation as long as the third component is preserved. We next show, that we can actually ignore the third component, allowing us to use any linear filter.

We are interested in establishing the equivalence between a filter in a height map representation of a surface and its normal representation. We do not want to obtain a height map explicitly, but it is a good abstraction to develop filters for normals. Assume our surface is given by $z = z(x, y)$. We can write the normal field as a function of its derivatives $z_x(x, y), z_y(x, y)$. Using the surface parametrization $\psi(x, y) = (x, y, z(x, y))$ whose tangent vectors are $\psi_x(x, y) = (1, 0, z_x), \psi_y(x, y) = (0, 1, z_y)$. We obtain the normal vector:

$$N(x,y) = \frac{\psi_x \times \psi_y}{|\psi_x \times \psi_y|} = \frac{(-z_x, -z_y, 1)}{\sqrt{z_x^2 + z_y^2 + 1}}$$

The above formula lets us convert from z_x, z_y to N. In fact the foreshorten correction scheme shown above is the reverse process. Given a unit normal $N = (n_1, n_2, n_3)$:

$$-n_1/n_3 = z_x, -n_2/n_3 = z_y$$

This can be done as long as $n_3 \neq 0$, otherwise there is no height map that represents this surface. What we have shown is a one-to-one mapping between the N and z_x, z_y, as such, we can work with one or the other indiscriminately. Note that we use the terminology z_x, z_y loosely here, since this field might not be the gradient of any height function. We defer a detailed discussion to section VI.

So we are looking for a filtering algorithm that takes the normals of a height map N^z and produces the normals of the filtered height map N^{z*g}. Conceptually we can go from N^z to z_x, z_y and then to z itself. We proceed by convolving z with a kernel g. With this new surface at hand, we can simply differentiate and take the vector product to obtain N^{z*g}, as shown below:

$$\begin{tikzcd}
z \arrow[r, "*g"] & z * g \arrow[d] \\
z_x, z_y \arrow[u] \arrow[r, "(1){,}*g"] & (z*g)_x, (z*g)_y \arrow[d] \\
N^z \arrow[u] \arrow[r, "?"] & N^{z*g}
\end{tikzcd}$$

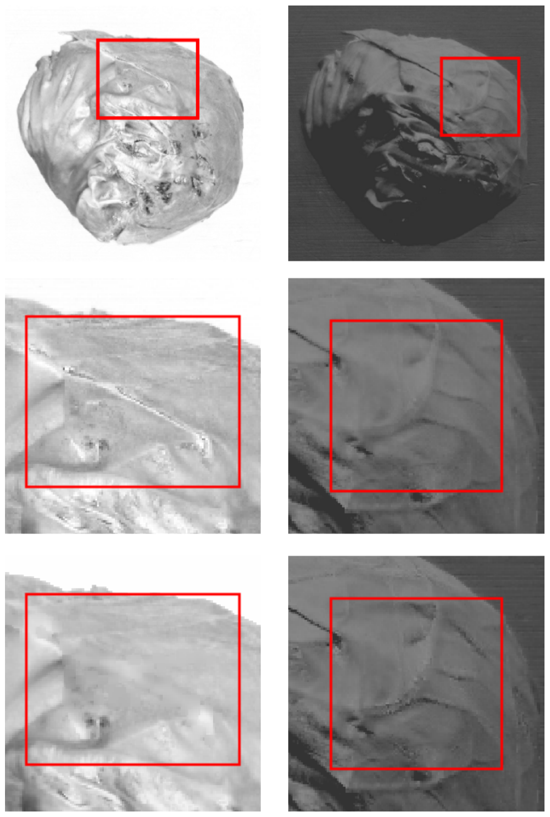
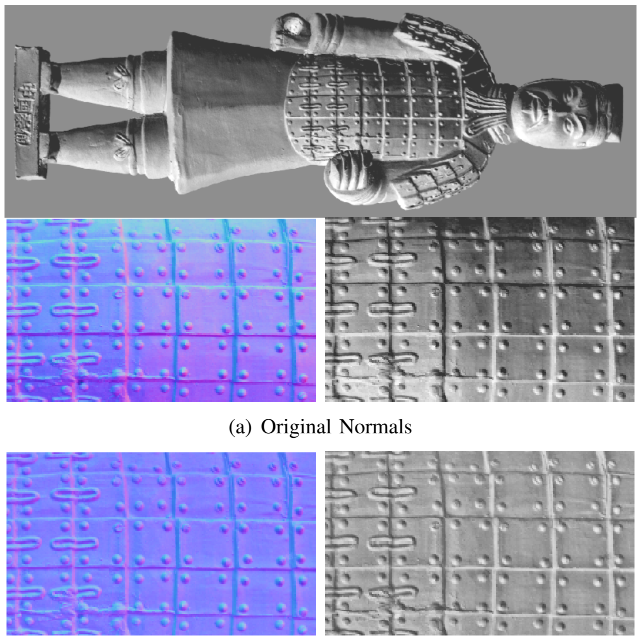

(a) Smoothed Normals (b) Enhanced Normals

Figure 1. Normal filtering can be applied locally as a brush.


(a) Original Normals

(b) High-pass filter result

Figure 2. A high-pass filter was used to remove the shape and retain a flat normal texture. All shaded images in this work are generated with directional lights.

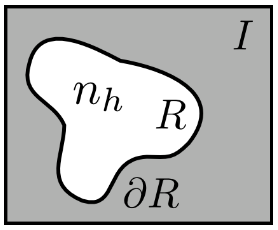


Figure 3. The detail normals n_h defined in R are combined with the image normals.

We want to avoid surface reconstruction. Fortunately, the arrow (1) above will provide a shortcut since derivatives and convolution satisfy the relation $(z * g)_x = (z_x * g)$.

This means when the normal is foreshorten corrected $(-z_x, -z_y, 1)$ we can convolve it with any kernel. Since we are only interested in filtering z_x, z_y, we can simply set the third component to 1, whether the kernel would preserve it or not. After filtering we simply normalize the vector to get a unit normal back. Notice that the resulting field is guaranteed to be conservative, since it is by definition the normals of a height function.

Having developed this filtering framework, we now discuss some applications. Gaussian Filtering, in fact, any low-pass filtering would smooth the normals. We are also interested in Sharpen Filters to enhance detail, i.e, an all-pass filter + high-pass filter. The problem with sharpening is that by enhancing high-frequencies we also enhance noise (Figure 1). A simple solution is using an Edge Enhancement Filter using all-pass + band-pass.

All of the above filters preserve the DC frequency. Filters that do not have this property can be useful for editing normals. For example, we might be interested in extracting a normal texture from an RGBN. In this case, we look for eliminating the low-frequencies related to shape and retaining the high-frequencies related to texture (Figure 2). A Difference of Gaussians and a Laplacian of Gaussian provide simple band-pass filters giving us the texture but also eliminating high-frequency components like noise. Another application is simply scaling the surface represented by the normals generating shallow surfaces.

We have developed a local filtering operator by applying the above filtering procedure in a small neighborhood. The user defines the shape and radius of the region. In Figure 1, we show smoothed and enhanced normals. Noise was also enhanced in this example. The local filter works well in practice, but it can be regarded as a filter with a spatially varying kernel, since it leaves most of the image unaffected. For this reason, the above integrability property may not hold.

IV. Linear Combination

In this section, we investigate a method of adding details to RGBNs. Details could be an applied stamp (Figure 4) or bumps painted by the user (Figure 6).

Given a base normal field n_b defined in the entire image I (Figure 3) and a detail normal field n_h defined in $R \subset I$, the problem of combining normals is generating a new normal field w which agrees with n_b everywhere, but is influenced in R by n_h. We refer to their respective height functions as b and h. We would like combination to be:

1) integrability preserving - lead to normal fields that correspond to a real surface;
2) frequency preserving - respect or replace selective bands.

Notice that simply adding the normal vectors and renormalizing does not satisfy any of the above properties. We propose the linear combination model which is integrability and frequency preserving. This is crucial for editing normal maps, since we usually want to edit mesostructure (normals) without affecting macrostructure possibly encoded in a different representation.

In the linear model we would rather look at the normal fields as derivatives. Just like in the previous section, suppose we could add height functions b and h, respectively base and details. By linearity of the gradient, $w = \nabla(b + h) = \nabla b + \nabla h$. So even if we do not have heights, we can still obtain the new gradient and thus the new normals. By construction the linear model preserves integrability, details in section VI. It is also frequency preserving, if a frequency band is not present in the details, it will be unharmed in w. To replace a given band, we can use a band removal filter in the original RGBN either globally or only in R, thus building the n_b normals which can then be transformed in the ∇b used above. This way we can respect not only low shape frequencies but also high texture frequencies, only changing mesostructure. In section VII a nonlinear combination method is also proposed.

Now that we have a good understanding of combination, we can look at how to specify the details. The first local operator we propose is inserting a small RGBN (stamp) in a neighborhood of a point (Figure 4). The stamp can be an entire RGBN itself or extracted from one. In the last case, we may want to eliminate the lower frequencies through filtering when creating n_h. Notice that if we want to apply any 2D linear transformation A to the stamp, gradients will not be simply copied like colors, instead we should transform them by $(A^{-1})^T$. This allows us to scale, rotate or shear. Translations do not affect normals. In the next section, a more advanced construction of the detail n_h is proposed.

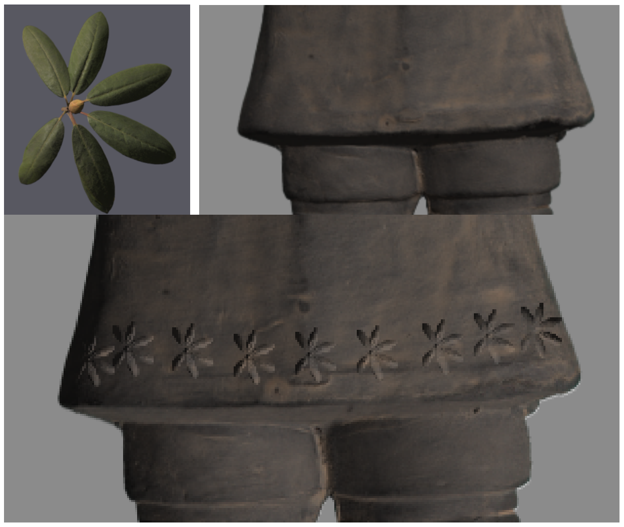


Figure 4. The leaf was used as a stamp and added as detail to the soldier's skirt. Notice how the results of each stamp are different, depending on the base normals.

V. Creating Features

To create line features we use the pen operator. In Figures 5 and 6, custom profiles are used to create bumps, creases or scratches on the surface [1]. A deformation $h(u, v)$ defined in a canonical domain has to be warped along an input path. We call this mapping from $(u, v) = \phi(x, y)$ (Figure 7). We then define the new gradient (or normals):

$$w = \nabla h \cdot D\phi = \begin{pmatrix} \frac{\partial h}{\partial u} & \frac{\partial h}{\partial v} \end{pmatrix} \begin{pmatrix} \frac{\partial u}{\partial x} & \frac{\partial u}{\partial y} \\ \frac{\partial v}{\partial x} & \frac{\partial v}{\partial y} \end{pmatrix}$$

These normals will be used to define the details and the combination proceeds like described in section IV. The above formula simplifies if we only want to transfer a profile, that is, if $h(u, v)$ does not depend on v.

The first step is to establish this mapping in a tubular neighborhood of the path (Figure 7) drawn by the user. The $u(x, y)$ coordinate can be regarded as a radial displacement from the path (distance), while the $v(x, y)$ coordinate as a displacement along the path (ideally arc-length parameterized). In the pen operator the profile is independent of the v coordinate, which simplifies the calculations.

To avoid slow distance field calculations, we use geometrical methods using point to segment distance functions and projections to build $\phi(x, y)$. This might not be the exact distance field because we calculate it locally and online as the user draws. Instead of the distance field approach, one could define the mapping $f(x, y)$ by extracting control points based on the input path and building a spline surface. This mapping can be built C^1 everywhere. After editing the normals, does the resulting normal field correspond to a surface? We analyze this question in the next section.

[1]Matching profiles and deformations is left as an exercise for the reader.

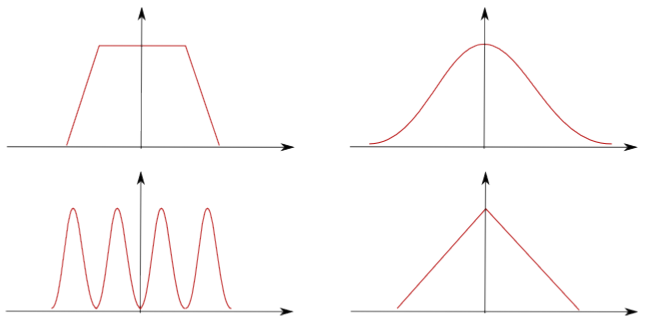


Figure 5. Custom height profiles along a path are transfered to the normals with the pen operator.

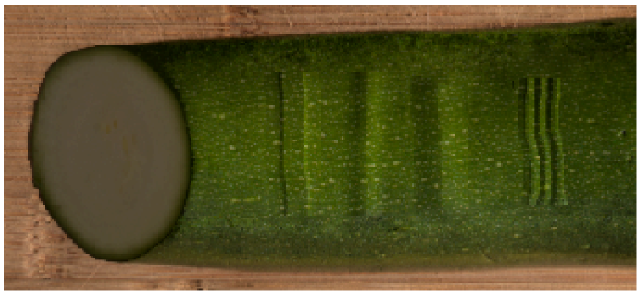


Figure 6. This cucumber was edited using the four profiles above. Height profiles are an intuitive way of specifying deformations.

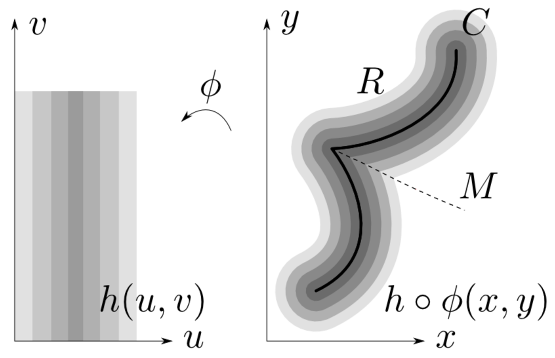


Figure 7. The distance field is used to build a mapping around the curve. It is discontinuous only over the path's medial axis.

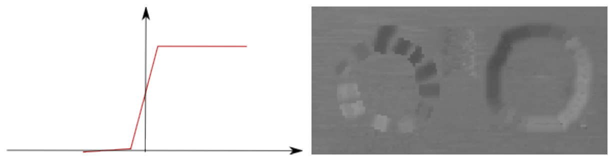


Figure 8. Notice how the shading of an impossible object (left) resembles a rotated version of a bump (right).

VI. Integrability

In the analysis that follows, it pays to look at normal image as 2D vector field w. If this field is the gradient of a C^1 height function, we say it is a conservative vector field. This means our normal image does correspond to a surface. We cay say $w = \nabla h$, even if we cannot build h explicitly.

The linear combination takes as input two conservative fields: base normals ∇h_1 and added detail ∇h_2. It then combines the gradients. Since $w = \nabla h_1 + \nabla h_2 = \nabla(h_1 + h_2)$, the new field is trivially conservative. One hypothesis used above is that the added detail is conservative. This holds for the stamp operator, but requires proof for the pen operator, since the wrapping could destroy this property.

Let's assume the curve C defining the edited region is a C^2 regular curve ($C'(t) \neq 0$). We can always approximate discontinuities with high curvatures. Under this hypothesis, there is a tubular neighborhood T where the mapping defined by ϕ is a diffeomorphism [18]. In fact this holds if T does not intersect C's medial axis M (Figure 7). We have defined the curved profile normals as $w = \nabla h \cdot D\phi = \nabla(h \circ \phi)$, so w is conservative and C^1, assuming h is C^1. Note that we do not have $h \circ \phi$, but we know this function exists. Nowhere in the above demonstration, we used that $h(u, v)$ depends only on u (as is the case for profile editing), so our method works for more patterns like normal textures.

The problem with the above hypothesis is that M can be arbitrarily close to the curve, making R a very thin region. If this does not hold, we can still show equivalency with a continuous surface. If we require $h(u, v)$ to depend only on the distance parameter u, we can show that w is C^1 everywhere, except for the points of M, where w is not defined. In fact, there is a C^0 function h_2 defined in the image I such that w is the gradient of h_2 in $I \backslash M$ where $h_2 = h \circ \phi$. ϕ is C^1 outside M. h_2 is continuous on M because h depends only on $u(x, y)$ which is a continuous function even on M. This discontinuous derivative will lead to discontinuous normals (creases) in the final continuous surface. This characteristic might be desirable or not. If $h(u, v)$ depends also on v, w is the gradient of a height function which may be discontinuous in $I \backslash M$.

Both the stamp and the pen operator are only defined locally, this is equivalent to defining h as being zero outside the edited region R. This requires ∇h to fall smoothly to zero close to ∂R and be equal to 0 outside. For an open curve C, this is not enough. A counter-example based on Escher's infinite stairs is the image below on which many deformations were made. The paths are radial and the height profile was a step function as shown in Figure 8. Each edit is a step up, so we could climb this stair infinitely up. To fix this issue, we ask one more property of w:

$$a, b \notin R \Rightarrow \int_a^b w \; ds = 0$$

This means open curves cannot introduce level changes

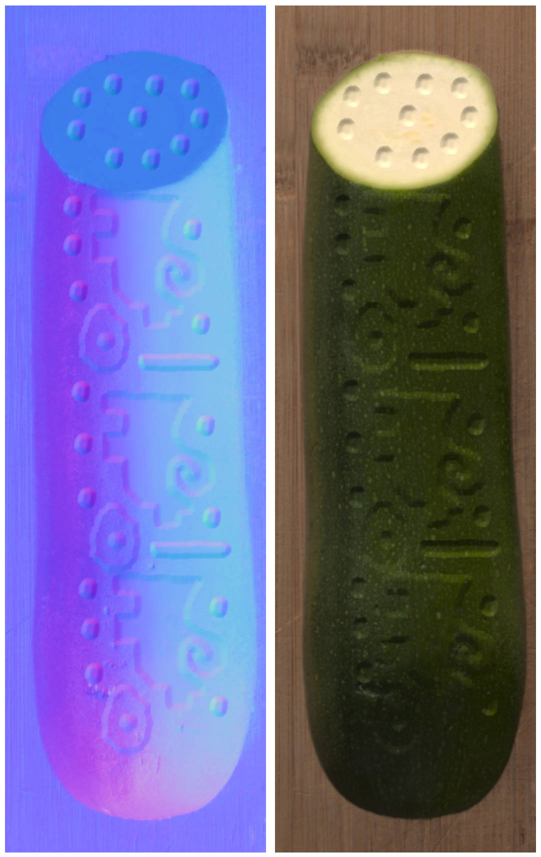


Figure 9. All carving was done with the pen operator. We can raise (lower) a region by tracing its border with a pen operator that introduces a level change.

outside R. For closed curves with no self-intersections the integral restriction above can be made weaker, the integral between the left side (inside of the curve) and the right side (outside) needs only to be a constant, not necessarily 0. Editing with this kind of profile raises (or shallows) the region inside the curve (Figure 9), but w is still guaranteed to be conservative. Notice that in this case $h(u, v)$ should be defined in a domain with the topology of a cylinder. By invariance of line integrals through diffeomorphisms, these properties can be checked in $h(u, v)$. Further care needs to be taken near the medial axis. For the local filtering operator, the integral restriction does not necessarily holds, as such the resulting field may not be conservative.

VII. Nonlinear normal editing

In the linear combination method, deformations are biased towards the camera (Figure 10). This problem is most noticeable near object silhouettes (Figure 12), since in these regions the normal is farther from the camera direction. This same problem is found in mesh deformation methods. We would like a method that wraps the deformations in the direction of the smooth normal. In this section we propose

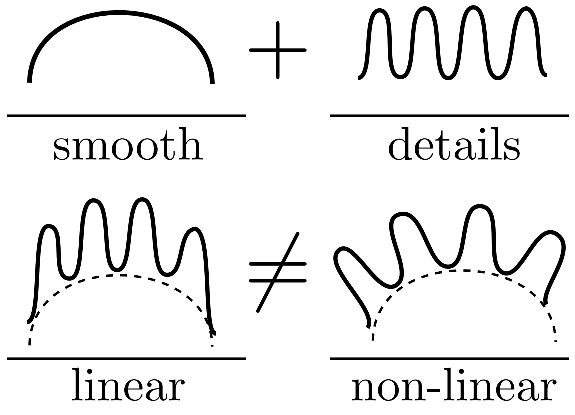


Figure 10. Comparison of the linear and nonlinear combination method.

a nonlinear combination method that works in the local tangent plane, thus eliminating camera bias.

Deformations in the rotation method are in the base normal direction, but since it is a nonlinear method we need a different way to control integrability. We propose a two-band separation scheme. The signal is split in smooth frequencies and details. Our operator will only affect the details and respect the smooth frequencies. For this reason, the rotation model can only be applied to small scale features. This approach leads to good visual results. The extension of this framework for editing normal maps on arbitrary meshes should follow naturally by assuming the base normals to be the fixed mesh normals.

As in Figure 3, n_b denotes the base normal field and n_h a detail normal field defined in $R \subset I$. We want n_b to be smooth. The problem with building n_b with linear filters is that edges are not preserved. Even though pixels near an edge are very close spatially their colors (or normals in our case) are not correlated. The idea of bilateral filtering [19] is to take into account not only domain weights (distance) but also range weights (color similarity). In [8], bilateral filtering was extended to RGBNs, using also normal similarity (see Figure 11). This filter produces great results visually, but it is not know if it is an integrability preserving operation.

In the rotation model, we define a local coordinate system in each point (tangent space) using n_b. We interpret n_h as being in this local system. We compose the two normals with rotations. Our local operations are transformations in a local tangent coordinate system u, v, n defined by the base normal vector n and an orientation vector o in the xy plane. Using rotations, it is easy to work on the tangent plane as well as interpolate operations. Editing the normal starts by specifying a desired normal vector b. We can simply replace the existing normal a with b. Another option is to blend the two vectors. Blending can be very useful for fading smoothly the effect of the new normals as distance from the edited area increases. We can also preserve frequencies if we rotate

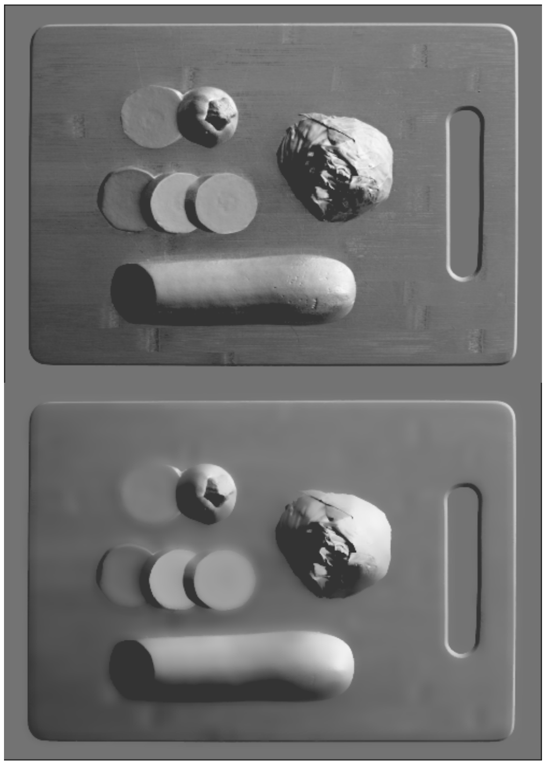


Figure 11. The original normal image is smoothed with bilateral filtering producing the base normals.

the existing normal a (including high frequencies) by the direction and angle specified by b (medium frequencies) in the local tangent system n_b (low frequencies). This strategy fails when b (in global coordinates) has a negative z value. In this case we actually need to replace b by a saturated vector k, defined as the maximum displacement along the great arc which has positive z values larger than a threshold.

VIII. Conclusion

Requiring the deformations to extend to 0 outside R is a limitation of our method, blending techniques should be investigated in the future to enforce this property on any deformation. Since the Jacobian $D\phi$ is used to warp the normals, the method would profit from high quality mappings around the curve. This is a problem only for deformations that depend on the arc-length parameter in high curvature regions.

If real-time feedback is not required, an exact distance transform can be calculated. This would allow curves of topology different from the unit real interval, like a T or H-shaped curve which are currently not supported by our local distance calculation. For profile editing, even curves with self-intersections would be possible. A different approach with the current system is to use the closed line pen operator to trace the border of a thin region, approximating the curve.

As discussed above Toler-Franklin et al. [8] have developed many rendering algorithms for RGBN. In this work we focused on editing geometry, but more visualization methods could be designed, building on our general filtering method.

Another line of work, focusing on appearance, is to work on the relation between reflectance properties, illumination conditions and geometry. We could paint highlights and find the appropriate light positions, or fix lighting and indirectly edit reflectance all on top of an RGBN.

One important limitation of RGBNs (2.5D in general) is that no change of viewpoint is allowed. In future work, a 3D mesh coupled with a projective atlas [9] will allow the extension of the above tools to any 3D model by editing chart by chart. A more general approach is to project an RGBN on demand and reflect the changes back to texture using any surface parametrization. Zbrush allows the user to edit attributes in the viewing plane and project back using the uv texture coordinates. RGBNs fit well in this framework since normals are also available, both for being edited themselves but also to simulate editing of other attributes on the surface. In this point of view, the RGBN encodes the surface's metric per pixel.

It is already possible to simulate normal texture synthesis with the stamp operator (Figure 12). More advanced results would be possible extending the work of Fang et al [20], [12] where the distortions encoded by the normals are taken into account for color synthesis on RGBNs. This would allow more advanced editing of normals and color in large regions. Our combination methods would easily apply. Just like texture synthesis, filtering methods for attributes should be developed taking into account the local metric.

We have shown the need for software specifically designed to edit RGBNs. Our system can filter normals allowing for low-pass, high-pass and edge enhancement. It contains brushes for adding detail and creating new features giving the user great control. Conditions were established that guarantee integrable results.

Acknowledgment

We thank Szymon Rusinkiewicz and the Princeton Graphics Group for the RGBN dataset used in this paper. We thank the anonymous reviewers for their valuable comments. We also thank Emilio Vital Brazil and Adriana Schulz for reviewing this manuscript. This research has been developed in the VISGRAF Laboratory at IMPA. The Laboratory is supported by the agencies CAPES, CNPq, FINEP, and FAPERJ and also by gifts from the companies IBM Brasil and Alias.

References

[1] J. F. Blinn, "Simulation of wrinkled surfaces," *SIGGRAPH Comput. Graph.*, vol. 12, no. 3, pp. 286–292, 1978.

[2] T.-P. Wu, C.-K. Tang, M. S. Brown, and H.-Y. Shum, "Shapepalettes: interactive normal transfer via sketching," *ACM Trans. Graph.*, 2007.

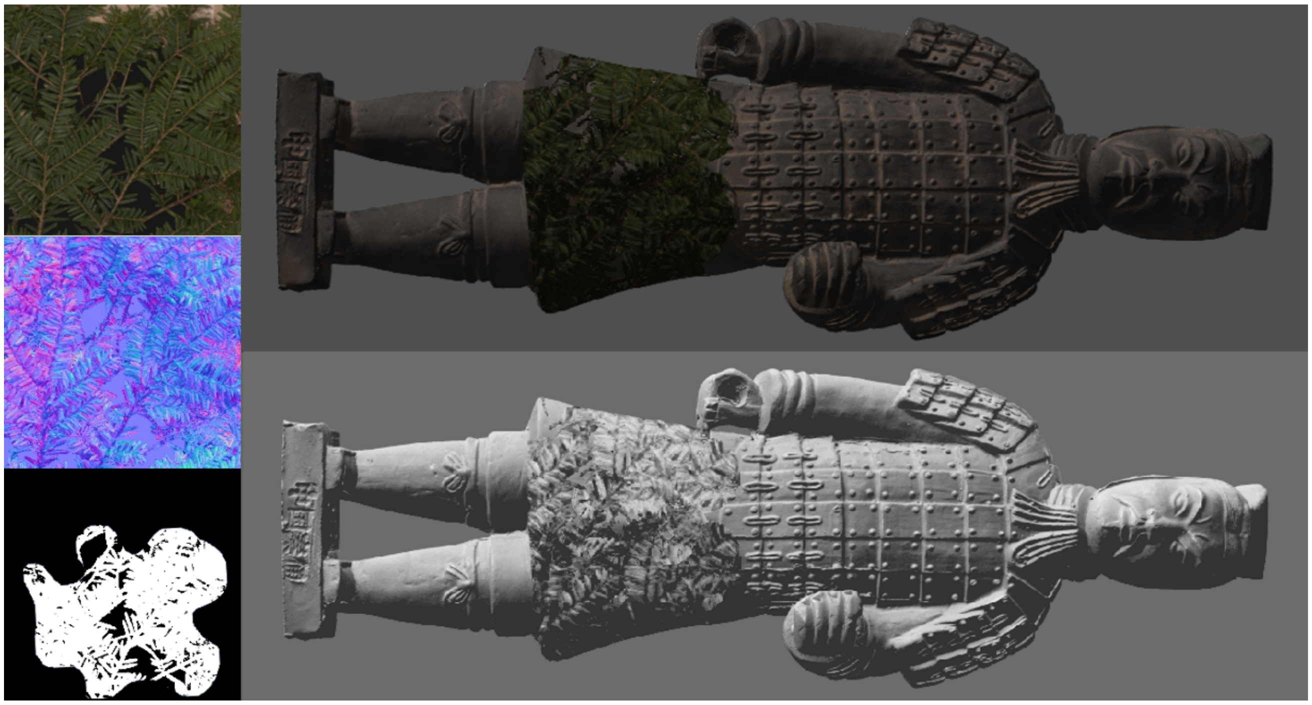


Figure 12. Branches were created on the soldier's skirt. On the left, the RGBN with the branches and a binary mask used to diffuse the regular square borders. The stamp was applied 3 times and segmentation methods were used to restrict editing to the skirt.

[3] J. Cohen, M. Olano, and D. Manocha, "Appearance-preserving simplification," in *SIGGRAPH '98: Proceedings of the 25th annual conference on Computer graphics and interactive techniques*, 1998, pp. 115–122.

[4] P. Cignoni, C. Montani, R. Scopigno, and C. Rocchini, "A general method for preserving attribute values on simplified meshes," in *VIS '98: Proceedings of the conference on Visualization '98*, 1998, pp. 59–66.

[5] R. J. Woodham, "Photometric method for determining surface orientation from multiple images," pp. 513–531, 1989.

[6] F. Bernardini, H. Rushmeier, I. M. Martin, J. Mittleman, and G. Taubin, "Building a digital model of michelangelo's florentine pieta," *IEEE Computer Graphics and Applications*, vol. 22, no. 1, pp. 59–67, 2002.

[7] D. Nehab, S. Rusinkiewicz, J. Davis, and R. Ramamoorthi, "Efficiently combining positions and normals for precise 3d geometry," *ACM Trans. Graph.*, vol. 24, no. 3, pp. 536–543, 2005.

[8] C. Toler-Franklin, A. Finkelstein, and S. Rusinkiewicz, "Illustration of complex real-world objects using images with normals," in *International Symposium on Non-Photorealistic Animation and Rendering (NPAR)*, Aug. 2007.

[9] L. Velho and J. Sossai Jr., "Projective texture atlas construction for 3d photography," *Vis. Comput.*, vol. 23, no. 9, pp. 621–629, 2007.

[10] Pixologic, "Online documentation," *http://www.pixologic.com*.

[11] S. Zelinka, H. Fang, M. Garland, and J. C. Hart, "Interactive material replacement in photographs," in *GI '05: Proceedings of Graphics Interface 2005*, 2005.

[12] H. Fang and J. C. Hart, "Textureshop: texture synthesis as a photograph editing tool," in *SIGGRAPH '04: ACM SIGGRAPH 2004 Papers*, 2004, pp. 354–359.

[13] P. Pérez, M. Gangnet, and A. Blake, "Poisson image editing," *ACM Trans. Graph.*, vol. 22, no. 3, pp. 313–318, 2003.

[14] M. Bammann Gehling, C. Hofsetz, and S. Raupp Musse, "Normalpaint: an interactive tool for painting normal maps," *Vis. Comput.*, vol. 23, no. 9, pp. 897–904, 2007.

[15] J. McCann and N. S. Pollard, "Real-time gradient-domain painting," *ACM Trans. Graph.*, vol. 27, no. 3, pp. 1–7, 2008.

[16] M. Toksvig, "Mipmapping normal maps," *journal of graphics tools*, vol. 10, no. 3, pp. 65–71, 2005.

[17] C. Han, B. Sun, R. Ramamoorthi, and E. Grinspun, "Frequency domain normal map filtering," *ACM Trans. Graph.*, vol. 26, no. 3, p. 28, 2007.

[18] M. D. Carmo, "Differential geometry of curves and surfaces," 1976.

[19] C. Tomasi and R. Manduchi, "Bilateral filtering for gray and color images," in *ICCV '98: Proceedings of the Sixth International Conference on Computer Vision*, 1998.

[20] S. Zelinka and M. Garland, "Jump map-based interactive texture synthesis," *ACM Trans. Graph.*, vol. 23, no. 4, pp. 930–962, 2004.