Geometry into Shading

Christophe Hery* Michael Kass† Junyi Ling‡
Technical Memo 14-04
Pixar Animation Studios

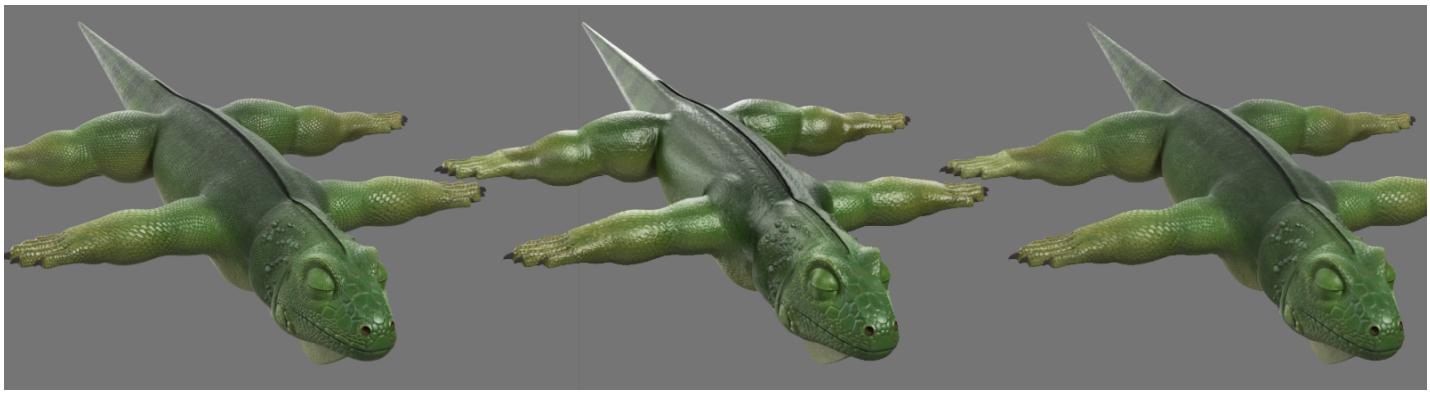


Figure 1: *the left-most lizard is rendered with a shading rate of 0.01, or 100 samples in a single pixel, to simulate the overall effect of displacement to illumination. This simulates the ground truth. The middle lizard is rendered with the shading rate of 1, this is a more "normal" render for production, but note all the fine details provided by displacement is lost. The right-most lizard is also rendered with the shading rate of 1, with our "bump roughness" method, which renders quickly. Yet it also preserves the visual interest of the ground truth render.*

Abstract

Sometimes, when 3D CG objects are viewed from a distance, a great deal of geometry can end up inside a single pixel. When this happens, full-sized facets, or pixels in a displacement map should play the role of traditional micro-facets. Here we describe how to alter traditional shading calculations to take this into account.

1 Introduction

Microfacet shading models have been widely used in computer graphics ever since Cook and Torrance [1982] introduced the approach. The idea of the microfacet approach is to characterize the portion of a surface visible at a sample by a distribution of surface orientations, and use the distribution to calculate an appropriate specular reflectance.

When objects with highly detailed shading are viewed from a distance, considerable portions of a displacement map may project to a single pixel. When this happens, common practice is simply to shade using a smoothed displacement map. Unfortunately, this practice yields surfaces which are too smooth, and which therefore tend to look like plastic.

The problem with the traditional practice is that it ignores the degree of bump-map variation within a pixel. In effect, when bump map samples become much smaller than pixels, they become micro-facets themselves, and should contribute to surface roughness. Here we develop some techniques for efficiently calculating the effect of bump-maps on surface roughness. Previous work in this area [Han et al. 2007] has used spherical harmonics and spherical expectation maximization to get at this issue using far more computation storage expense and complexity than the approach we present here.

Similar to [Dupuy et al. 2013] and [Olano and Baker 2010], we also use two mipmapped textures to approximate microfracet-based BRDFs. This report presents work that was independently derived from the basic definition of a

*e-mail:chery@pixar.com
†e-mail:kass@pixar.com
‡e-mail:jling@pixar.com

microfacet-based specular BRDF. It is also different and more modular for our pipeline. We hope that this report would prove useful in a shading and rendering environment similar to ours.

2 Displacement Maps

One popular and physically-based microfacet approach is the Beckman distribution [Beckmann and Spizzichino 1968]. In this model, the specular reflectance is given by

$$K_{spec} = \frac{\exp(-\tan^2(\alpha)/m^2)}{\pi m^2 \cos^4(\alpha)} \tag{1}$$

where $\alpha = \arccos(N \cdot H)$ and H is the half-angle direction midway between the light vector L and the viewpoint vector V. Here, m is $\sqrt{2}$ times the root-mean square of the microfacet slopes, or equivalently, the standard deviation of the slope samples.

Now, we consider a displacement-mapped parametric surface, where the displacements are all in the normal direction. Then a point on the surface is given by

$$P(u,v) = R(u,v) + \frac{N(u,v)}{|N(u,v)|} * w(u,v) \tag{2}$$

where R is the undisplaced surface, N is the normal, and w is a scalar displacement field.

Often, displacement maps are used to provide high spatial frequency details to much smoother geometry. In this case, we can consider the undisplaced surface normal N to be slowly varying compared to the displacement.

In that case, the effective slope of a displaced microfacet is the sum of its micro-slope and the slope $(\partial w/\partial u, \partial w/\partial v)$ of the displacement map. If the microfacet distribution and the displacement map are statistically independent, then their energies will add. So if m_{orig} is the roughness term of the microfacet distribution and m_{disp} is $\sqrt{2}$ times the RMS slope of displacement then m_{eff}, the effective value of m is given by

$$m_{eff} = \sqrt{(m_{orig}^2 + 2 * \sigma^2)} = \sqrt{(m_{orig}^2 + m_{disp}^2)} \tag{3}$$

With traditional shading, we already know m_{orig}. It remains to calculate the additional contribution to the roughness due to the displacement map. To do that, we need the standard deviation of the displacement map slope relative to the normal plane evaluated in the area A corresponding to the sample. In the u direction, the effective displacement map slope is given by

$$s_u = \frac{\partial w}{\partial u}\left(\frac{\partial R}{\partial u}\right)^{1/2} \tag{4}$$

Similarly, in the v direction, the effective displacement map slope is

$$s_v = \frac{\partial w}{\partial v}\left(\frac{\partial R}{\partial v}\right)^{1/2} \tag{5}$$

And we seek standard deviation m_{disp} of $\sqrt{(s_u^2 + s_v^2)}$ over the area A.

Once again, we make use of the fact that, in general, the undisplaced geometry $R(u,v)$ is slowly varying relative to the displacement. When that holds, we can ignore changes in arc-length scaling over the area A, and the slope standard deviation can be written:

$$m_{disp} \approx \left[(\partial R/\partial u)^{-2}\sigma(\partial w/\partial u)^2 + (\partial R/\partial v)^{-2}\sigma(\partial w/\partial v)^2\right]^{1/2} \tag{6}$$

If the displacement $w(u,v)$ is a simple texture, we can evaluate the necessary standard deviations by using the machinery of ordinary texture filtering. Let $f_A(x)$ denote a standard texture filter corresponding to the area A. Then, if we want to evaluate the standard deviation of some quantity x over that area, it is given by

$$\sigma(x) = \sqrt{f_A(x^2) - f_A(x)^2} \tag{7}$$

In our case, we have two standard deviations to compute: $\partial w/\partial u$ and $\partial w/\partial v$. To do this efficiently at render time, we pre-compute images for the u derivative $w_u(u,v)$ and the v derivative $w_v(u,v)$ by simple finite differences.

This can be done easily, for example, in Nuke, see figure 2. (note that one must take care to make sure that no gamma correction or other tone mapping occurs when saving these out as textures). We also pre-compute images for $w_u^2(u,v)$ and $w_v^2(u,v)$. These are simply pixel by pixel squares of the other two pre-computed images. Then using texture map filtering, we can get a good estimate of m_{disp} as follows

$$m_{disp} \approx \left([f(w_u^2) - f(w_u)^2](\partial R/\partial u)^{-2} + [f(w_v^2) - f(w_v)^2](\partial R/\partial v)^{-2}\right)^{1/2} \tag{8}$$

2.1 Anisotropy

More generally, this analysis can be extended to the Beckmann Anisotropic BRDF. In this section, we convert the previous derivation to cover two seperate specular roughness values in two different directions. We note that a microfacet distribution along u,v tangent space can be expressed as a covariance matrix $\mathbf{A_{X,Y}}$ in 2-d

$$\mathbf{A_{X,Y}} = \begin{bmatrix} \mathbf{A_{XX}} & \mathbf{A_{XY}} \\ \mathbf{A_{YX}} & \mathbf{A_{YY}} \end{bmatrix} \tag{9}$$

where $\mathbf{A_{XX}}$ = var(X), $\mathbf{A_{YY}}$ = var(Y), and $\mathbf{A_{XY}}$ = cov(X,Y). By definition, a covariance matrix is positive definite and symmetric, such that all its eigenvalues are positive and that $\mathbf{A_{XY}}$=$\mathbf{A_{YX}}$. Microfacet variance in the u, v directions can be defined in a similar way to the isotropic case. Also recall that m = $\sqrt{2}\sigma$ and var() = σ^2. Hense $var() = 1/2m^2$, and we apply scalar of $1/2$ to all variance terms. We can now define the terms of the covariance matrix of the microfacet distribution explicitly.

$$\begin{aligned} \mathbf{A_{XX}} &= \frac{1}{2}\ [f(w_u^2) - f(w_u)^2](\partial R/\partial u)^{-2} \\ \mathbf{A_{YY}} &= \frac{1}{2}\ [f(w_v^2) - f(w_v)^2](\partial R/\partial v)^{-2} \\ \mathbf{A_{XY}} &= \frac{1}{2}\ \frac{[f(w_u.w_v) - f(w_u)f(w_v)]}{(\partial R/\partial u)(\partial R/\partial v)} \end{aligned} \tag{10}$$

This is a valid statistical representation of the microfacet slope variance in 2-d. For ease of use, we need to convert this representation into a format our Beckmann Anisotropic Specular BRDF consumes [Hery and Villemin 2013]. For that, we compute the two eigenvalues and their associated eigenvectors from the covariance matrix $\mathbf{A_{X,Y}}$, such that

$$\begin{aligned} \mathbf{A_{X,Y}}.\mathbf{V_1} &= l_1.\mathbf{V_1} \\ \mathbf{A_{X,Y}}.\mathbf{V_2} &= l_2.\mathbf{V_2} \end{aligned} \tag{11}$$

This is a standard eigen-decomposition. We provide example renderman shading code in the following sections for reference. And finally, the anisotropic effective roughness in each direction can be expressed as

$$\begin{aligned} m_{eff1} &= \sqrt{m_{orig1}^2 + l_1} \\ m_{eff2} &= \sqrt{m_{orig2}^2 + l_2} \end{aligned} \tag{12}$$

For each of the principle directions, $\mathbf{V_1}$ and $\mathbf{V_2}$ respectively. We note that the covariance matrix can be extended into 3-d as well, with a final projection step that projects the 3-d anisotropic microfacet variance onto a projected tangent space. This would allow surface parameterization-independent roughness calculations.

2.2 Illumination

Our final implementation note is that from this point, the parameters m_{eff1}, m_{eff2} and $\mathbf{V_1}$, $\mathbf{V_2}$ are plugged directly into Beckmann Anisotropic BRDFs. Pixar's physically-based GI rendering system will adjust lighting and BRDF samples accordingly. Dome, Point, Area, Image-based lights, as well direct and indirect reflections behave correctly without additional overhead.

3 Results

The verification of our process is conducted with a control group, which is shaded with traditional displacement at a shading rate of 0.01. This simulates the ground truth by rendering images resolved with brute force pixel samples. We then compare traditional displacement shaded geometry vs. bump roughness renders. Both of these test scenarios are rendered with the standard shading rate of 1. To see visual comparison, see Figures Table 1.

Three-sets for tests are run under different lighting environments. The first row of tests are lit with a point light source to show that bump roughness preserving anisotric nature of the concentric grooves. The additional images are rendered with Grace Cathedral IBL. Note that even though the bump roughness renders are slightly different from the ground truth control renders, the nature of the displaced surfaces are fully preserved with bump roughness and the look is vastly superior to simple displacement renders. We also provide videos for additional examples accompanying this report to illustrate these findings.

4 Conclusion

In this report, we have presented a simple and versatile method to preserve geometric details, that have been filtered away in rendering, by statistically converting them to roughness of a Beckmann BRDF. We have shown that it preserves important visual details that are lost with traditional bump and displacement methods. It is cheap to evaluate in modern rendering systems. It plugs directly into a BRDF, such that no modification is needed past the material-level description and has no effect on how the BRDF is sampled or light transport is integrated.

References

BECKMANN, P., AND SPIZZICHINO, A. 1968. *The Scattering of Electromagnetic Waves from Rough Surfaces*.

COOK, R. L., AND TORRANCE, K. E. 1982. A reflectance model for computer graphics. *ACM Trans. Graph. 1*, 1 (Jan.), 7–24.

DUPUY, J., HEITZ, E., AND OSTROMOUKHOV, V. 2013. To appear in acm tog 32(6). linear efficient antialiased displacement and reflectance mapping.

HAN, C., SUN, B., RAMAMOORTHI, R., AND GRINSPUN, E. 2007. Frequency domain normal map filtering. *ACM Transactions on Graphics (Proceedings of SIGGRAPH 2007) 26*, 3, 28:1–28:12.

HERY, C., AND VILLEMIN, R. 2013. Siggraph 2013. physically based lighting at pixar, physically based shading course.

OLANO, M., AND BAKER, D. 2010. Lean mapping. *In Proc. Symp. on Interactive 3D Graphics and Games, ACM*, 181–199.

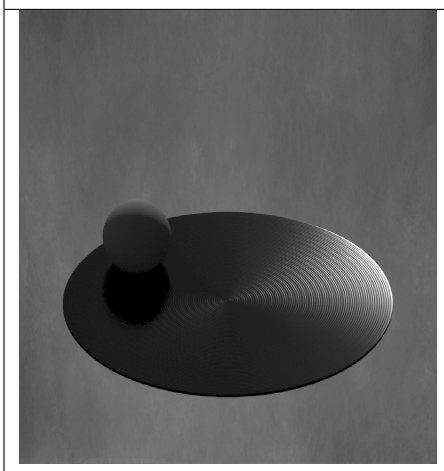
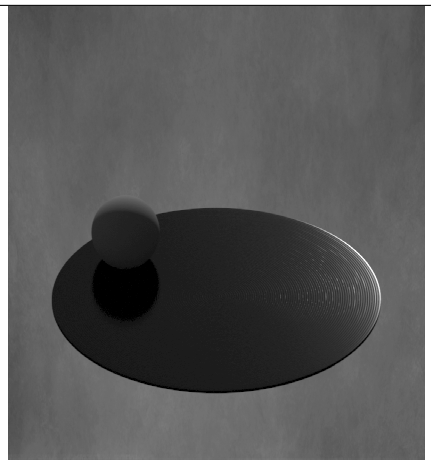
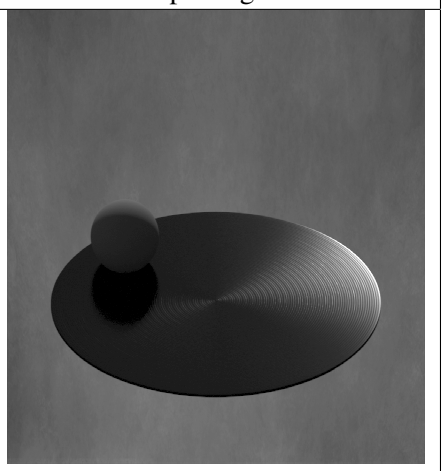
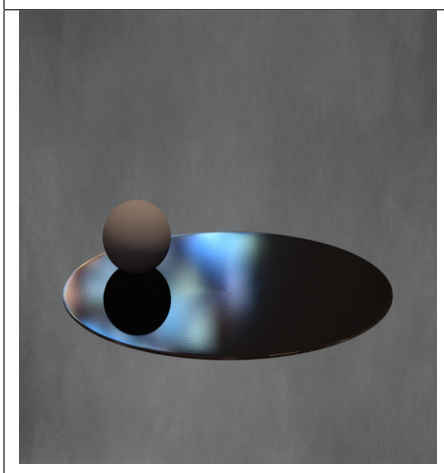
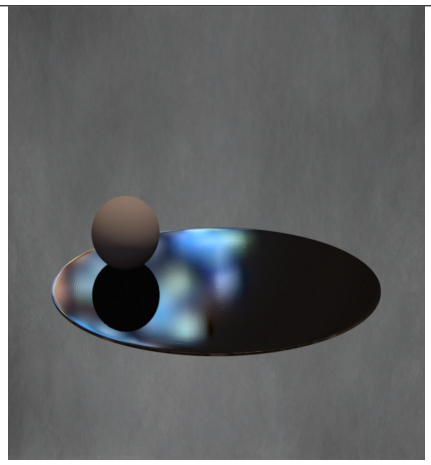
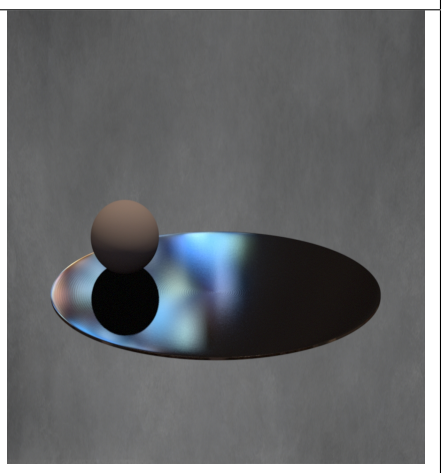
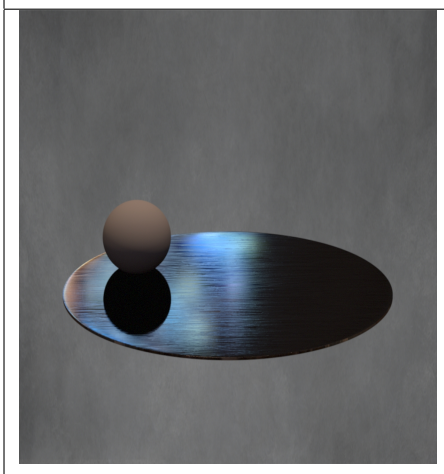
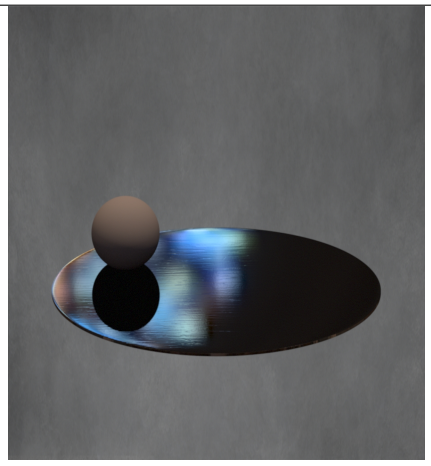
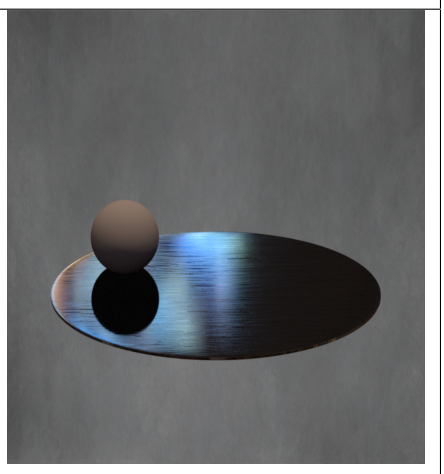
Control	Traditional	Bump Roughness
		
		
		

Table 1: *Left column renders are with low shading rates, Mid column renders are normal shading rate renders without bump roughness converted, the right column has bump maps converted to roughness. Note how our method recovers visual details lost in normal bump mapping*

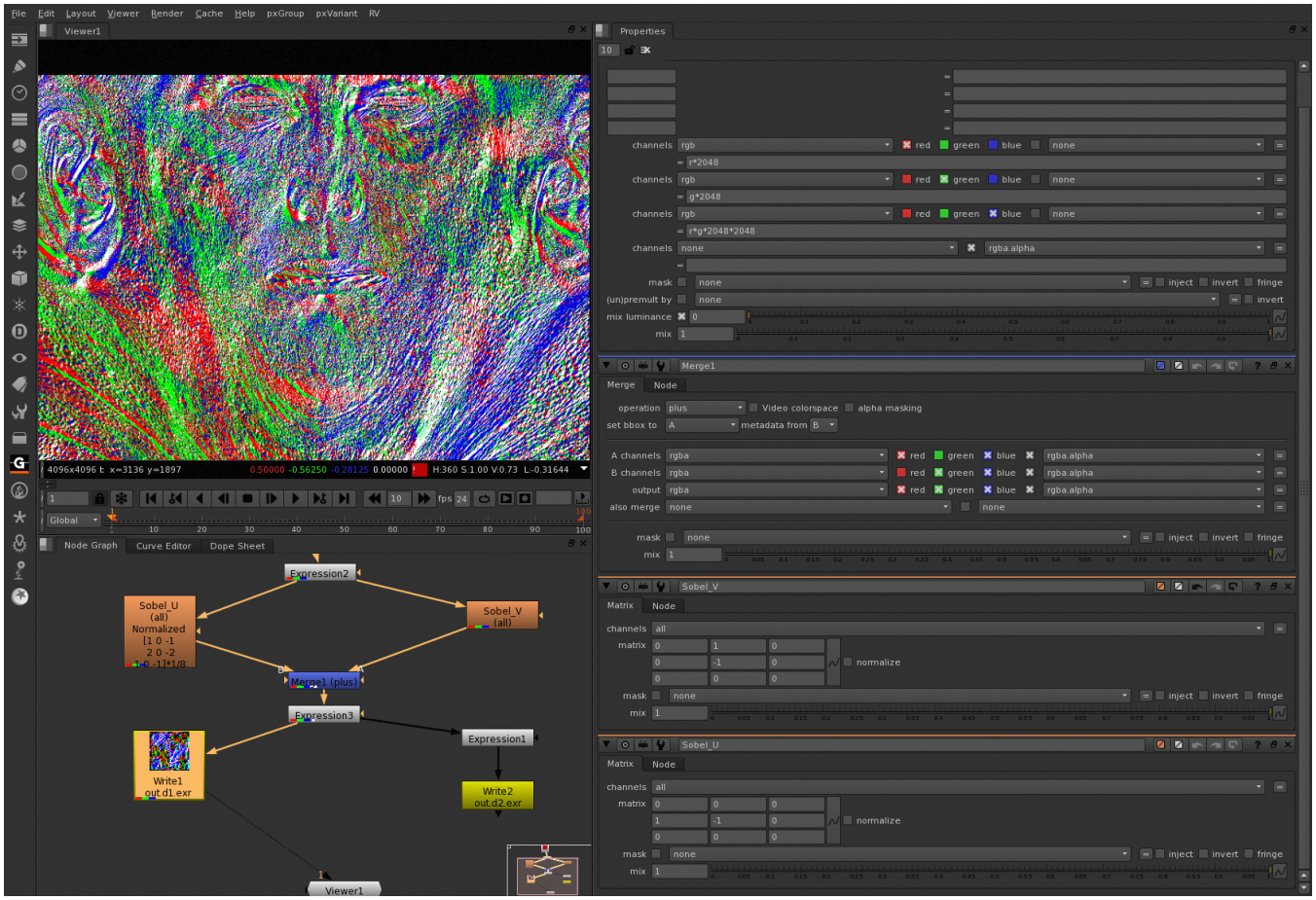


Figure 2: *nuke filter node*

5 Sample Renderman Code

```
void PartialDeriv(  varying point Q;
                    varying float a;
                    varying float b;
                    output vector dQda;
                    output vector dQdb;) {

    varying vector dQdu = Du(Q);
    varying vector dQdv = Dv(Q);

    varying float A = Du(a);
    varying float B = Du(b);
    varying float C = Dv(a);
    varying float D = Dv(b);

    varying float det = (A*D - B*C);

    dQda = (dQdu*D - dQdv*B)/det;
    dQdb = (dQdv*A - dQdu*C)/det;
}
```

Above sample RSL calculates partial derivative of point Q with respect to a surface parameterization a and b; Note that Q can also be a float-type variable, this would make dQda and dQdb output float type variables as well.

```
void SymetricEigenVecVal2D
    (
    varying float a;
    varying float b;
    varying float c;
    output varying float l1;
    output varying float l2;
    output varying vector v1;
    output varying vector v2;
    ){

    //del is the discriminate
    float del = sqrt(a*a+4*b*b-2*a*c+c*c);

    //solve for eigenvalues
    l1 = .5*(a+c+del);
    l2 = .5*(a+c-del);

    //two eigenvectors v1, v2
    v1=vector(0);
    v1[0] = 1;
    v1[1] = (l1-a)/b;
    v1 = normalize(v1);

    v2=vector(0);
    v2[0] = 1;
    v2[1] = (l2-a)/b;
    v2 = normalize(v2);
}
```

Above sample RSL converts a 2-d covariance matrix to two sets of eigenvalues and eigenvectors.

```
void calcBumpRoughness(
    //surface parameterization
    varying float u;
    varying float v;

    //input deriv maps precalucated
    //from displacement.
    //note this is read in as textures
    //so the inputs are filtered f(x)

    varying color d1;
    varying color d2;
    output varying float l1, l2;
```

```
    output varying vector v1, v2;
    )
{
    float axx=.5*(d2[0]-d1[0]*d1[0]);
    float axy=.5*(d2[2]-d1[0]*d1[1]);
    float ayy=.5*(d2[1]-d1[1]*d1[1]);

    vector pRpu, pRpv;
    PartialDeriv(P, u, v, pRpu, pRpv);

    SymetricEigenVecVal2D(
        axx,axy,ayy,l1,l2,v1,v2);

    float d=length(pRpv)*length(pRpu);

    l1=l1/d;
    l2=l2/d;
    v1=pRpv*v1[0]+pRpu*v1[1];
    v2=pRpv*v2[0]+pRpu*v2[1];
}
```

Above sample RSL calculates variances l1,l2 and principle directions v1 and v2 from the deriv maps d1, d2, which were read from offline textures. Note that the eigenvectors are projected back to ”current” space with the partial derivatives of P with respect to the surface parameterization of u and v.