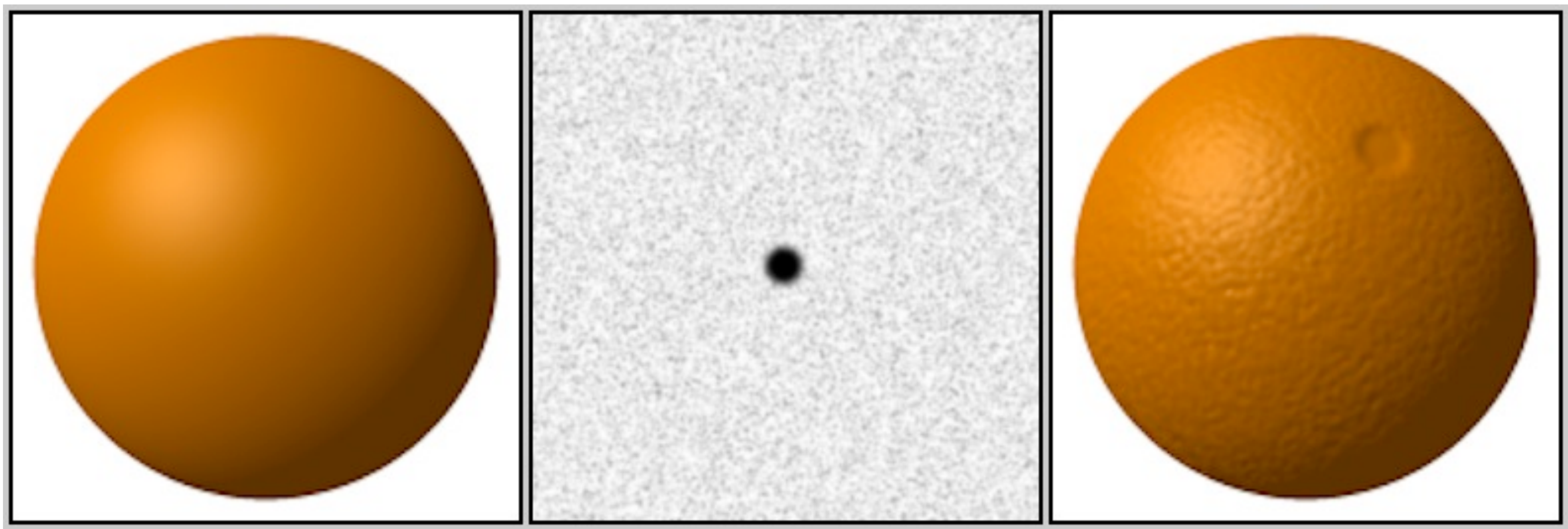
Rendering Rough Surfaces

Prof. Vladlen Koltun
Computer Science Department
Stanford University

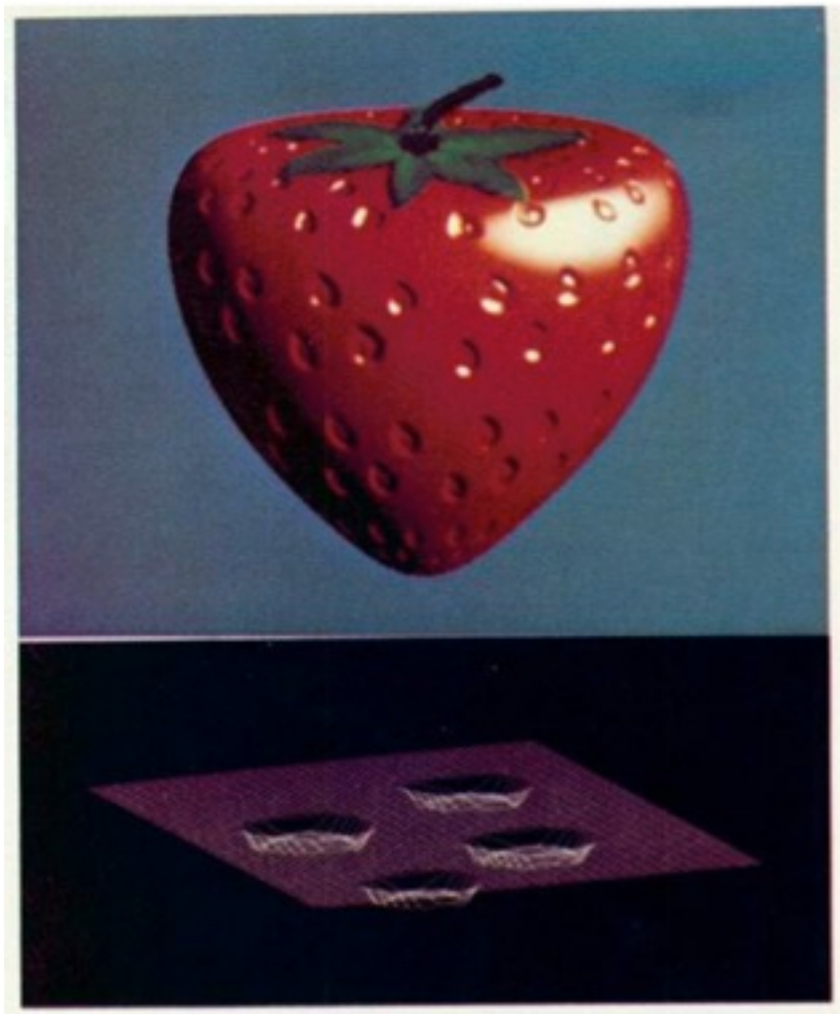
Bump mapping

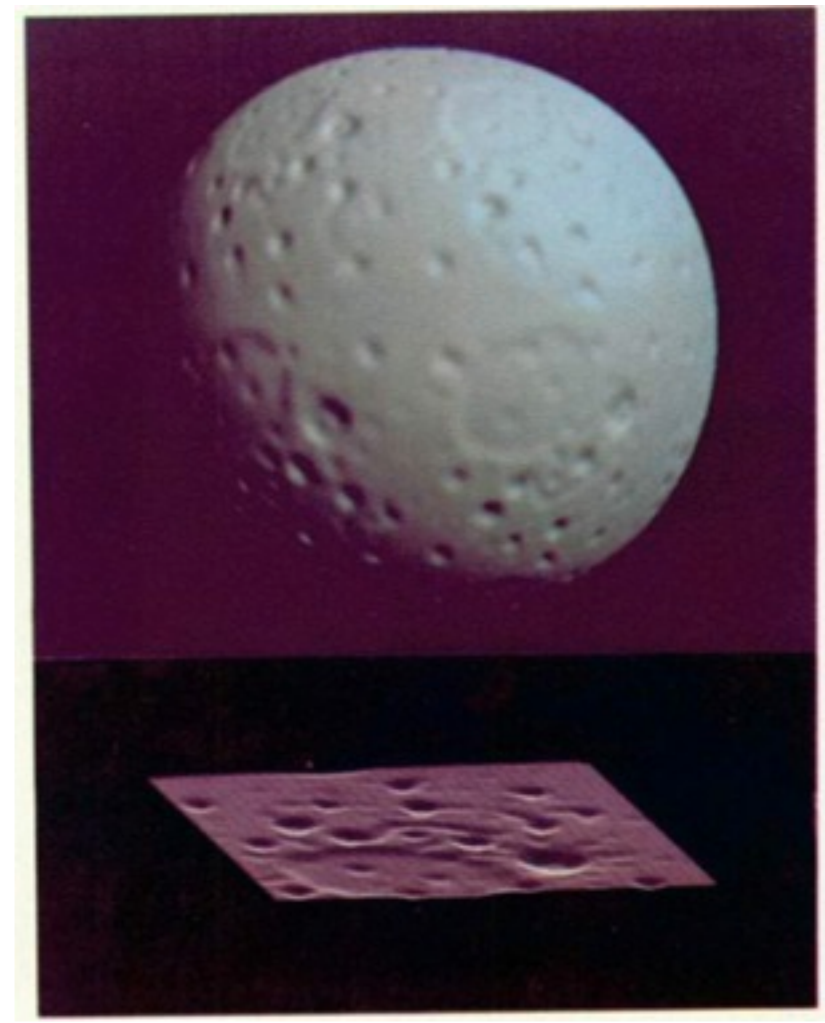


- Simulates roughness ("bumpiness") of a surface without adding geometry
- Uses a two-dimensional height field (bump map) to perturb the normal during per-fragment shading calculations
- Limitations: the mapping of texture onto the surface is unaffected; silhouette is also unaffected.

GDallimore (Wikimedia Commons)

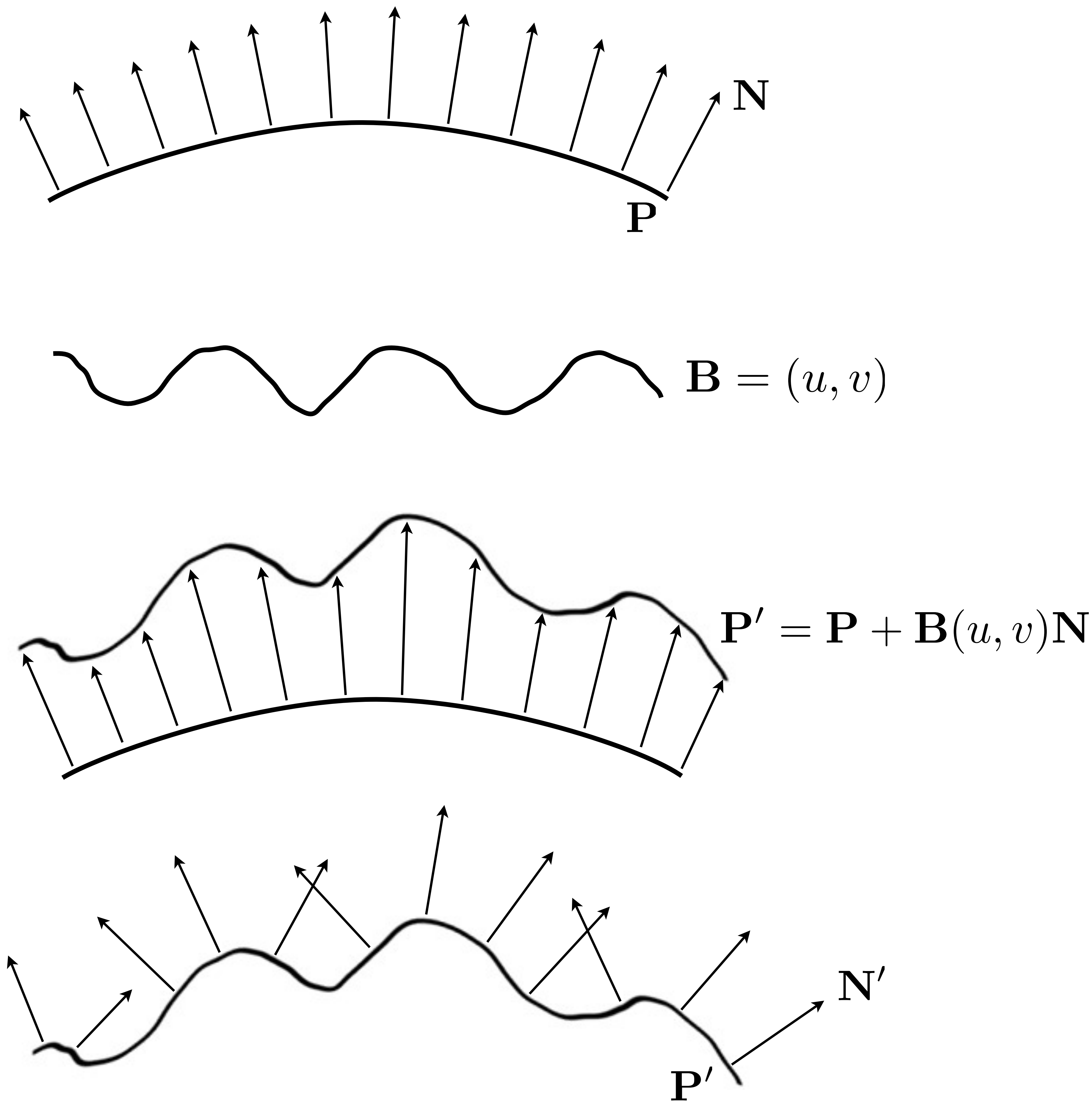
Bump mapping



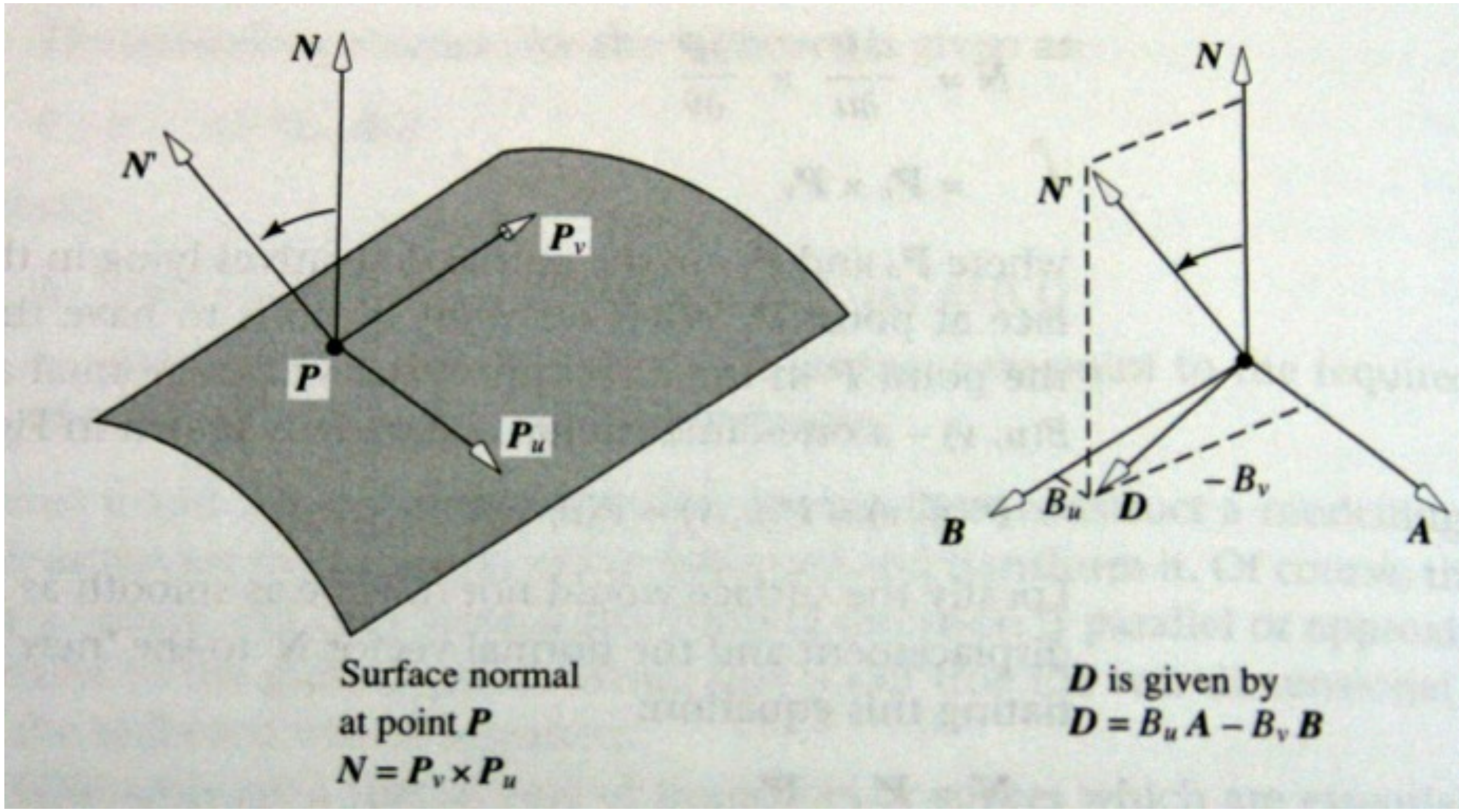


- Simulates roughness (“bumpiness”) of a surface without adding geometry
- Uses a two-dimensional height field (bump map) to perturb the normal during per-fragment shading calculations
- Limitations: the mapping of texture onto the surface is unaffected; silhouette is also unaffected.

Blinn, SIGGRAPH 1978


N
P
$\mathbf{B} = (u, v)$
$\mathbf{P}' = \mathbf{P} + \mathbf{B}(u, v)\mathbf{N}$
N′
P′

Bump mapping



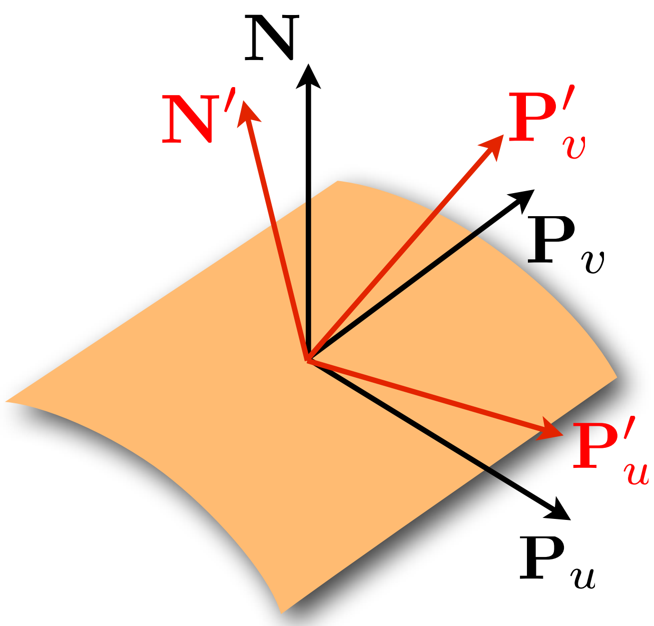
$$\mathbf{T} = \mathbf{P}_u = \left(\frac{\partial x}{\partial u}, \frac{\partial y}{\partial u}, \frac{\partial z}{\partial u}\right)$$

$$\mathbf{B} = \mathbf{P}_v = \left(\frac{\partial x}{\partial v}, \frac{\partial y}{\partial v}, \frac{\partial z}{\partial v}\right)$$

$$\mathbf{N} = \mathbf{P}_u \times \mathbf{P}_v$$

Watt, 3D Computer Graphics

Bump mapping derivation



$$\mathbf{N} = \mathbf{P}_u \times \mathbf{P}_v$$
$$\mathbf{N}' = \mathbf{P}'_u \times \mathbf{P}'_v$$

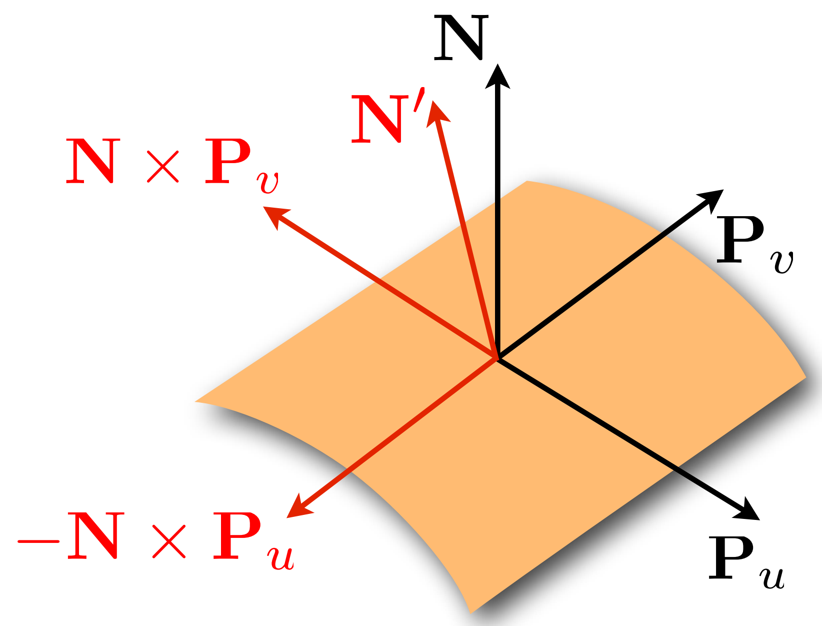
The normals are always appropriately normalized, but we omit that for simplicity

$$\begin{aligned} \mathbf{P}'_u &= \frac{\partial \mathbf{P}'}{\partial u} \\ &= \frac{\partial}{\partial u}(\mathbf{P} + \mathbf{B}(u,v)\mathbf{N}) \\ &= \mathbf{P}_u + \frac{\partial \mathbf{B}}{\partial u}\mathbf{N} + \mathbf{B}(u,v)\frac{\partial \mathbf{N}}{\partial u} \\ &\approx \mathbf{P}_u + \frac{\partial \mathbf{B}}{\partial u}\mathbf{N} \end{aligned}$$

$$\begin{aligned} \mathbf{P}'_v &= \frac{\partial \mathbf{P}'}{\partial v} \\ &= \frac{\partial}{\partial v}(\mathbf{P} + \mathbf{B}(u,v)\mathbf{N}) \\ &= \mathbf{P}_v + \frac{\partial \mathbf{B}}{\partial v}\mathbf{N} + \mathbf{B}(u,v)\frac{\partial \mathbf{N}}{\partial v} \\ &\approx \mathbf{P}_v + \frac{\partial \mathbf{B}}{\partial v}\mathbf{N} \end{aligned}$$

These are not really partial derivatives, but we follow Blinn's original notation

Bump mapping derivation

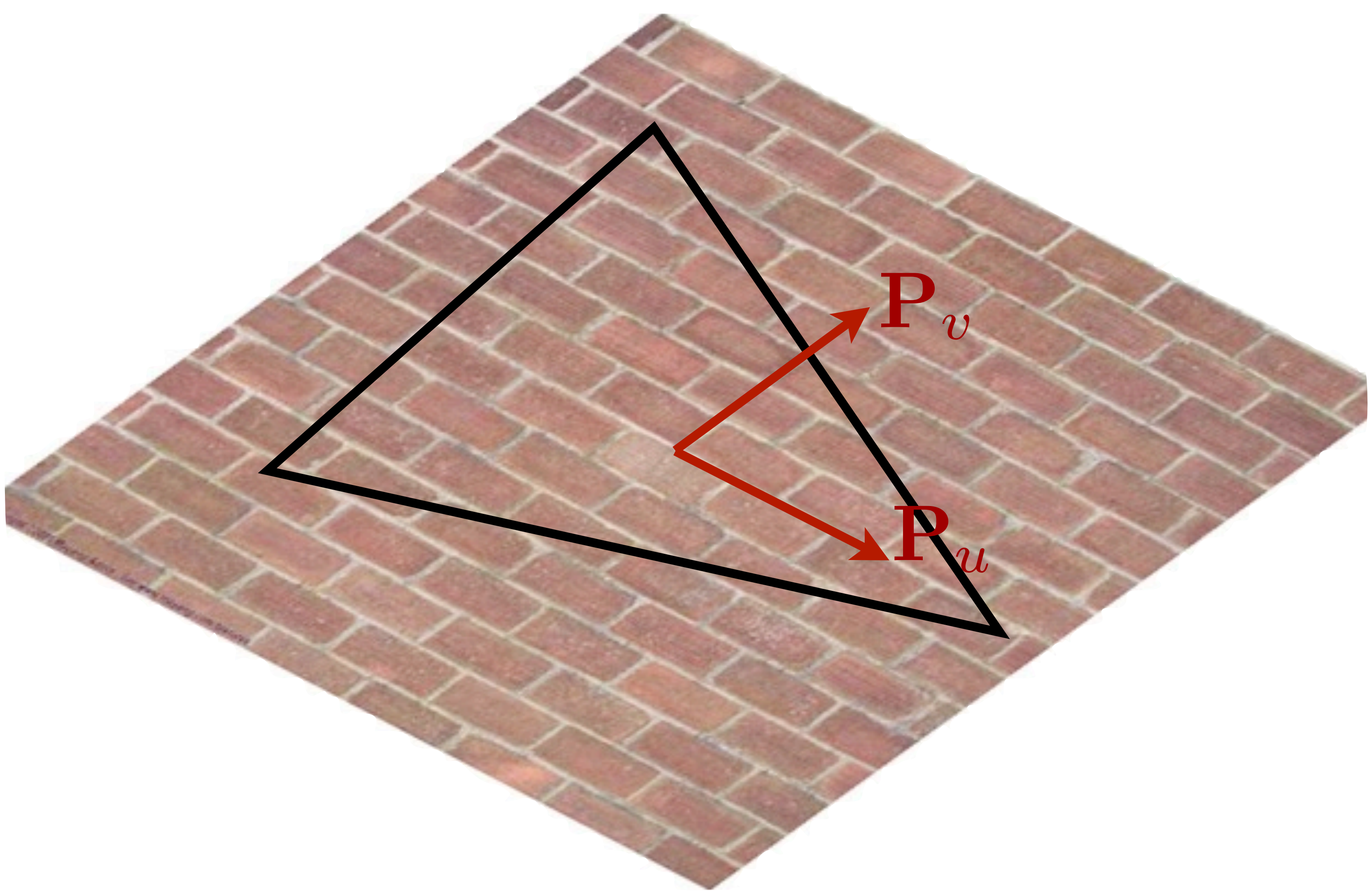


$$
\begin{aligned}
\mathbf{N}' &= \mathbf{P}'_u \times \mathbf{P}'_v \\
&= \left(\mathbf{P}_u + \frac{\partial \mathbf{B}}{\partial u}\mathbf{N}\right) \times \left(\mathbf{P}_v + \frac{\partial \mathbf{B}}{\partial v}\mathbf{N}\right) \\
&= \mathbf{N} + \frac{\partial \mathbf{B}}{\partial u}\mathbf{N} \times \mathbf{P}_v + \frac{\partial \mathbf{B}}{\partial v}\mathbf{P}_u \times \mathbf{N} \\
&= \mathbf{N} + \frac{\partial \mathbf{B}}{\partial u}\mathbf{N} \times \mathbf{P}_v - \frac{\partial \mathbf{B}}{\partial v}\mathbf{N} \times \mathbf{P}_u
\end{aligned}
$$

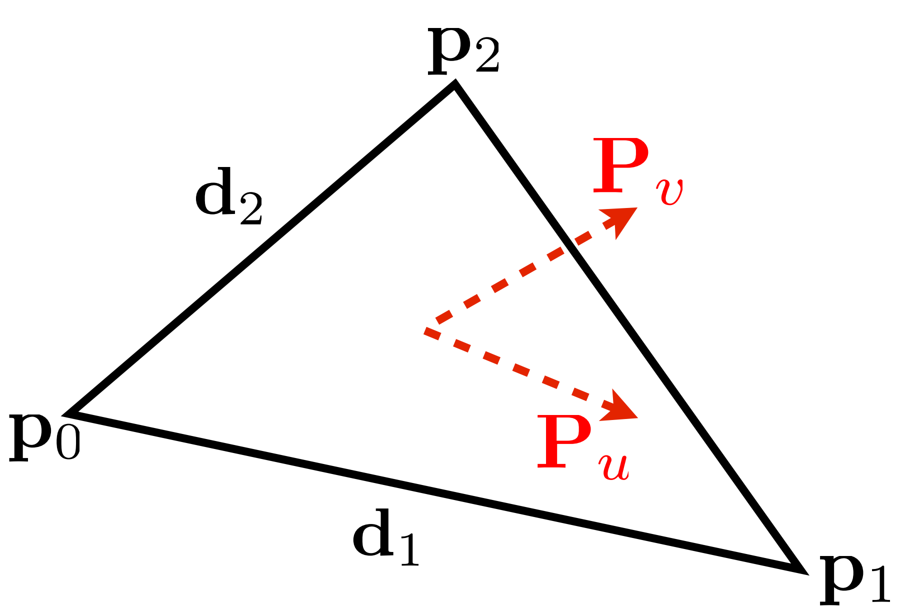
The "partial derivatives" are just differences between adjacent pixel values in the bump map

Computing tangent space basis vectors

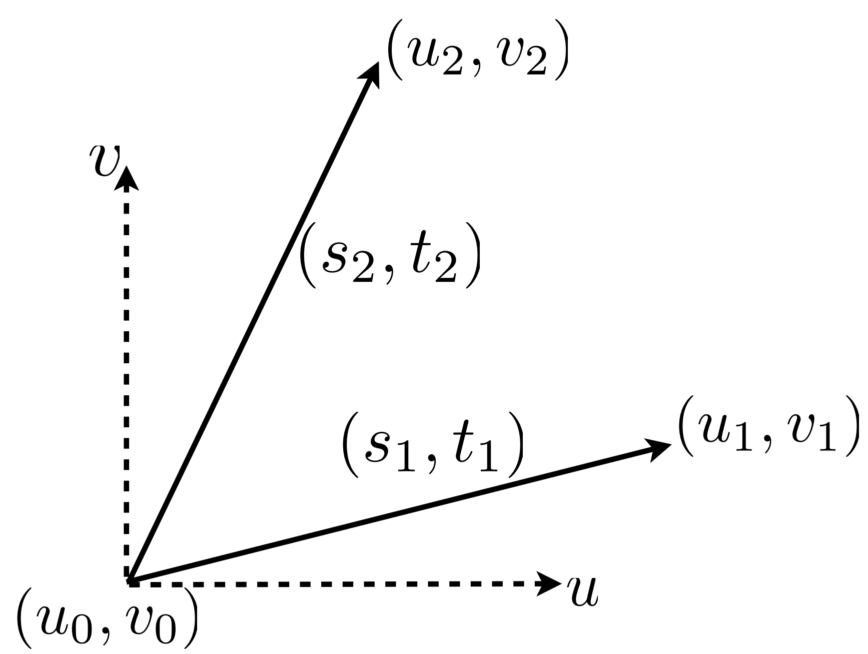
We now need to compute the vectors $\mathbf{P}_u$ and $\mathbf{P}_v$. These are the directions on the surface that correspond to zero change in the *v* parameter and the *u* parameter, respectively.



Computing tangent space basis vectors



object space



texture space

$$\mathbf{d}_1 = \mathbf{p}_1 - \mathbf{p}_0$$

$$\mathbf{d}_2 = \mathbf{p}_2 - \mathbf{p}_0$$

$$(s_1, t_1) = (u_1 - u_0, v_1 - v_0)$$

$$(s_2, t_2) = (u_2 - u_0, v_2 - v_0)$$

$$\mathbf{d}_1 = s_1\mathbf{P}_u + t_1\mathbf{P}_v$$

$$\mathbf{d}_2 = s_2\mathbf{P}_u + t_2\mathbf{P}_v$$

Computing tangent space basis vectors

$$\mathbf{d}_1 = s_1 \mathbf{P}_u + t_1 \mathbf{P}_v$$
$$\mathbf{d}_2 = s_2 \mathbf{P}_u + t_2 \mathbf{P}_v$$

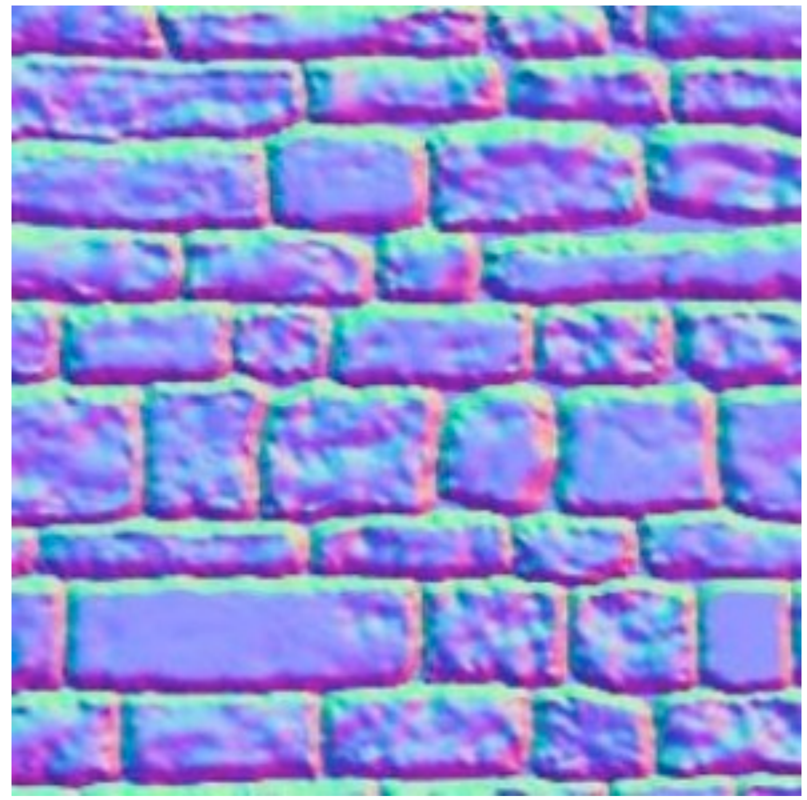
$$\begin{pmatrix} \mathbf{d}_1^\top \\ \mathbf{d}_2^\top \end{pmatrix} = \begin{pmatrix} s_1 & t_1 \\ s_2 & t_2 \end{pmatrix} \begin{pmatrix} \mathbf{P}_u^\top \\ \mathbf{P}_v^\top \end{pmatrix}$$

$$\begin{pmatrix} \mathbf{P}_u^\top \\ \mathbf{P}_v^\top \end{pmatrix} = \frac{1}{s_1 t_2 - s_2 t_1} \begin{pmatrix} t_2 & -t_1 \\ s_2 & s_1 \end{pmatrix} \begin{pmatrix} \mathbf{d}_1^\top \\ \mathbf{d}_2^\top \end{pmatrix}$$

We can now normalize these vectors and use them for bump mapping.

Normal mapping

original wall



normal map

shaded wall

- Store the displaced normals directly. Reduces runtime overhead, at the expense of memory requirements
- (x,y,z) values in the tangent space are stored in the RGB channels. To compute the normal at a fragment, we simply multiply the (interpolated) tangent space basis by (x,y,z)

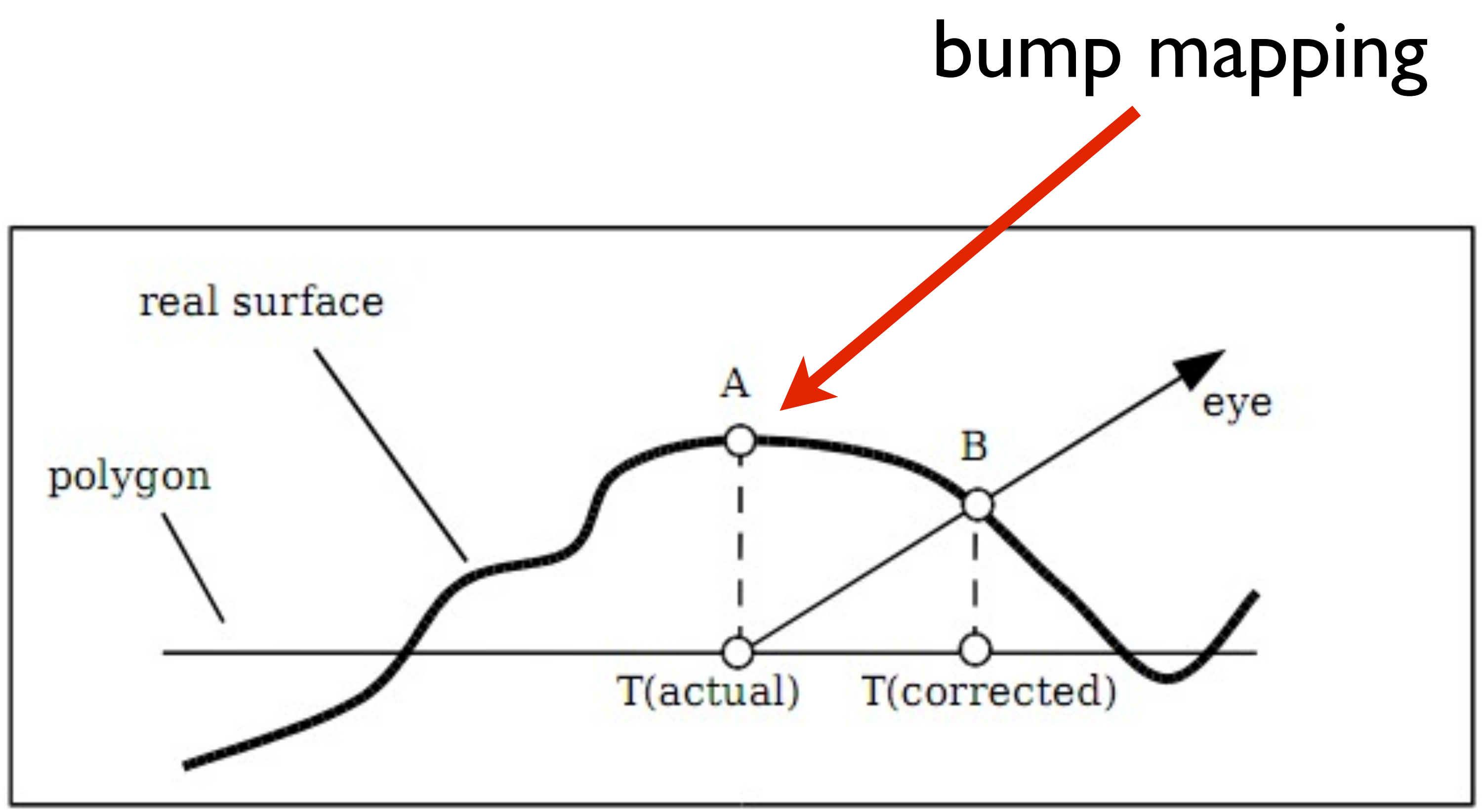
http://www.blacksmith-studios.dk/projects/downloads/bumpmapping_using_cg.php

Bump mapping limitations

- The application of texture onto the surface is not affected by the simulated medium-scale geometry. Only shading is affected.
- The “bumps” are illusory, which is apparent at grazing angles: the texture appears “flattened”.

Image from Natalya Tatarchuk

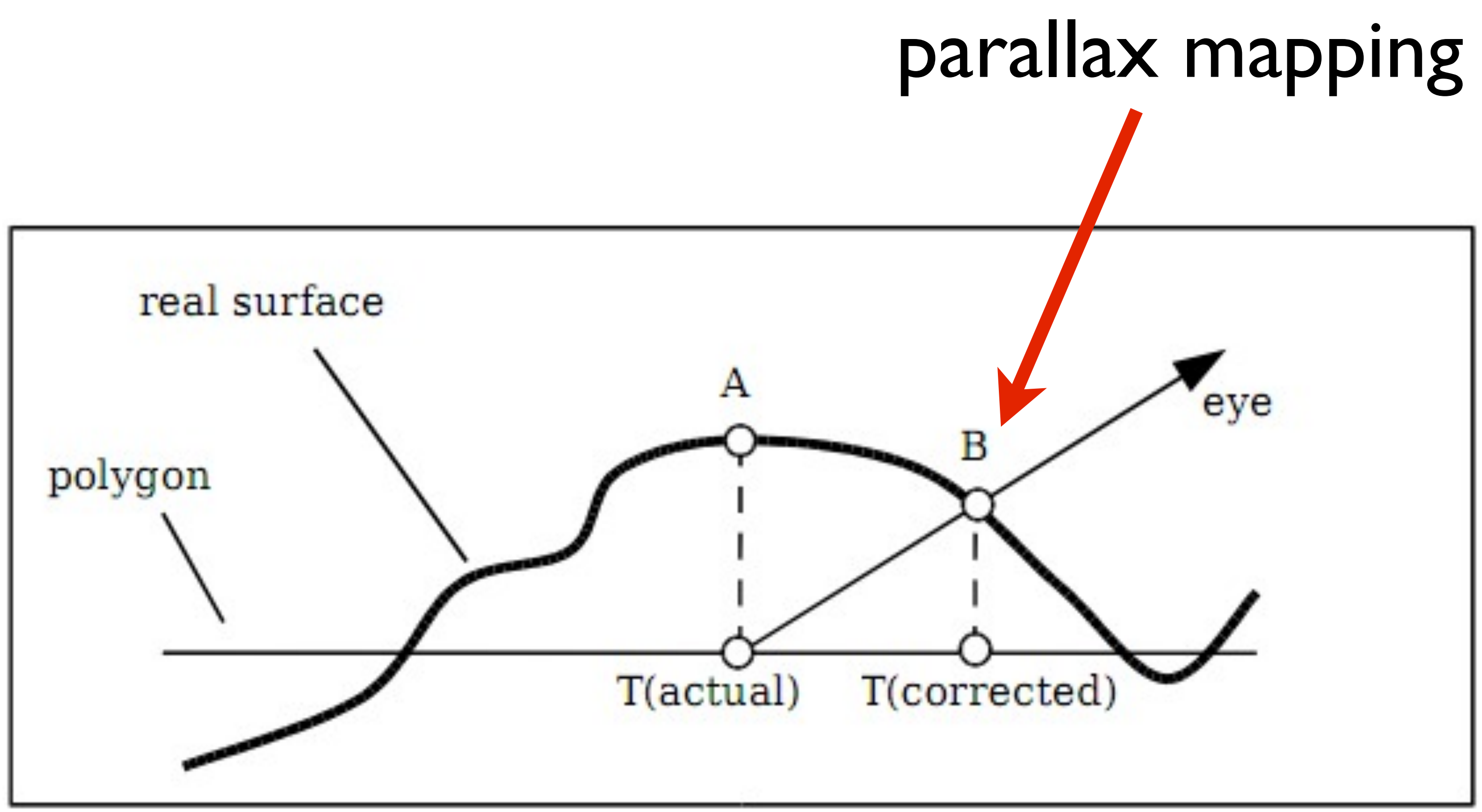
Parallax mapping



- With bump mapping or normal mapping, the texture parameters are mapped onto the surface without consideration of the medium-scale geometry modeled by the bump map.

Image from Terry Welsh

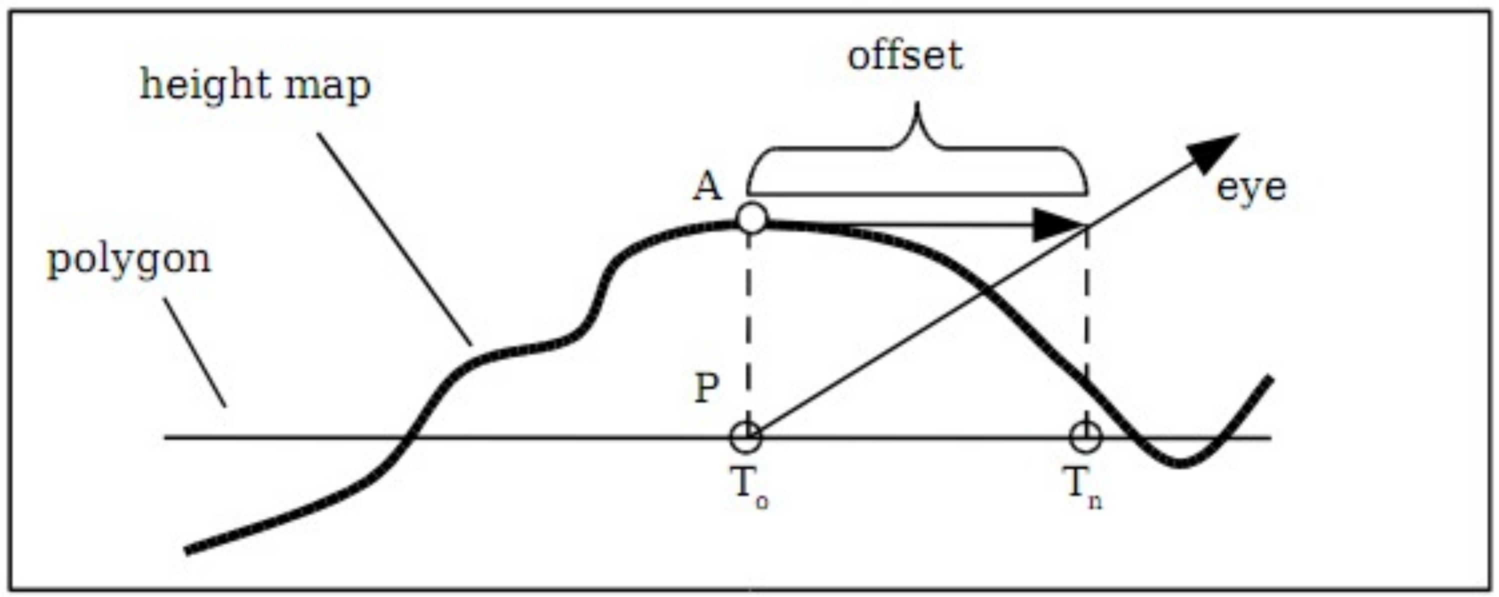
Parallax mapping



- We would like to find the texture parameters that correspond to the "actual" point on the perturbed surface that is hit by the view vector

Image from Terry Welsh

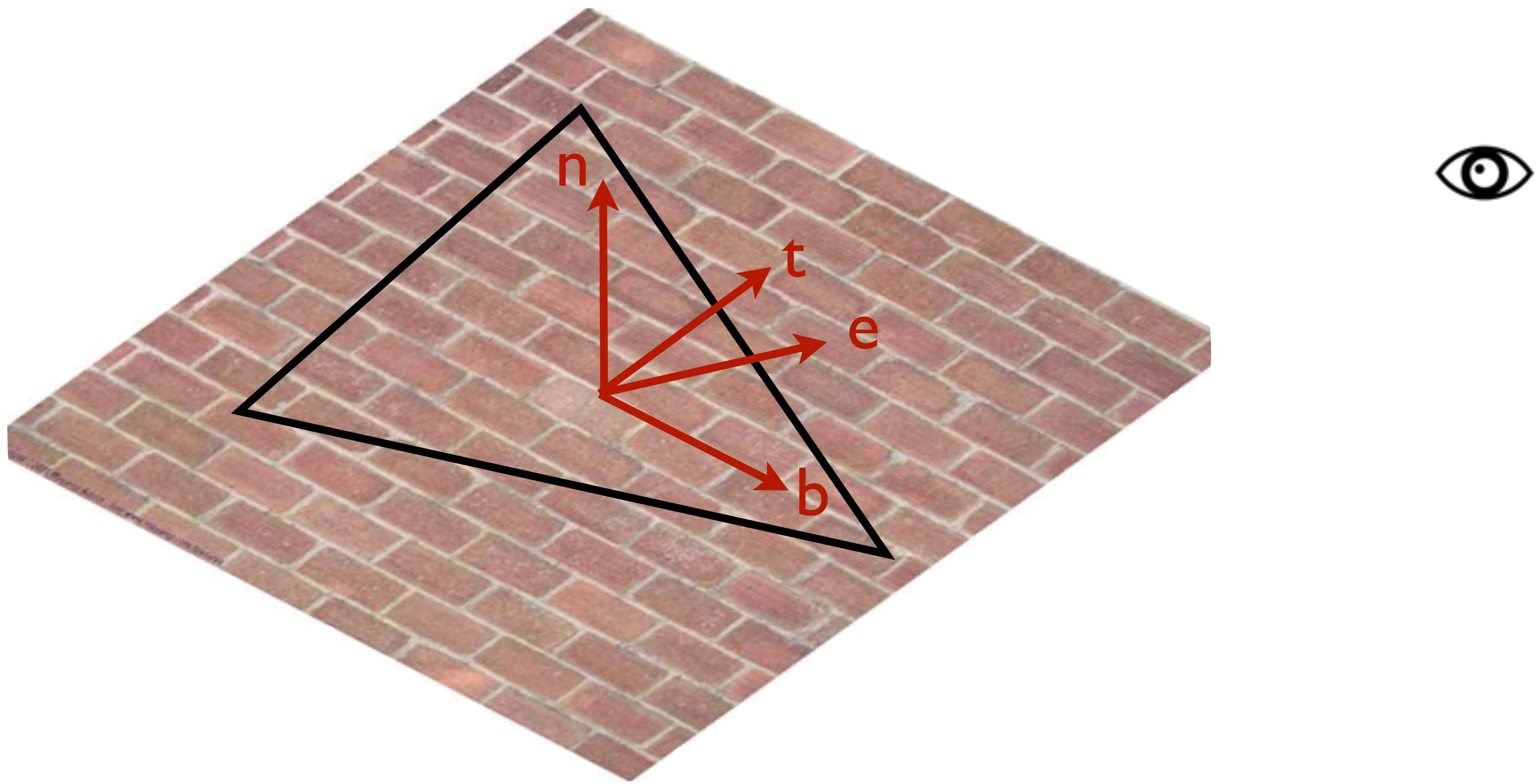
Parallax mapping



- In parallax mapping, we approximate this by offsetting the texture coordinates by an offset determined by projecting the view vector onto the tangent plane.

Image from Terry Welsh

Parallax mapping



Let e be the normalized eye vector

We can compute its projection onto each axis of the tangent frame: e_n, e_t, e_b . (Simply take the dot product.) Let $e_{tb} = e_t + e_b$

Parallax mapping

Let h be the height of the bump map at the current point. Assuming this height is locally constant, the offset is given by

$$h\frac{e_{tb}}{e_n}$$

For grazing angles, the offset can be very large, resulting in uncontrolled sampling of the texture. We can use

$$he_{tb}$$

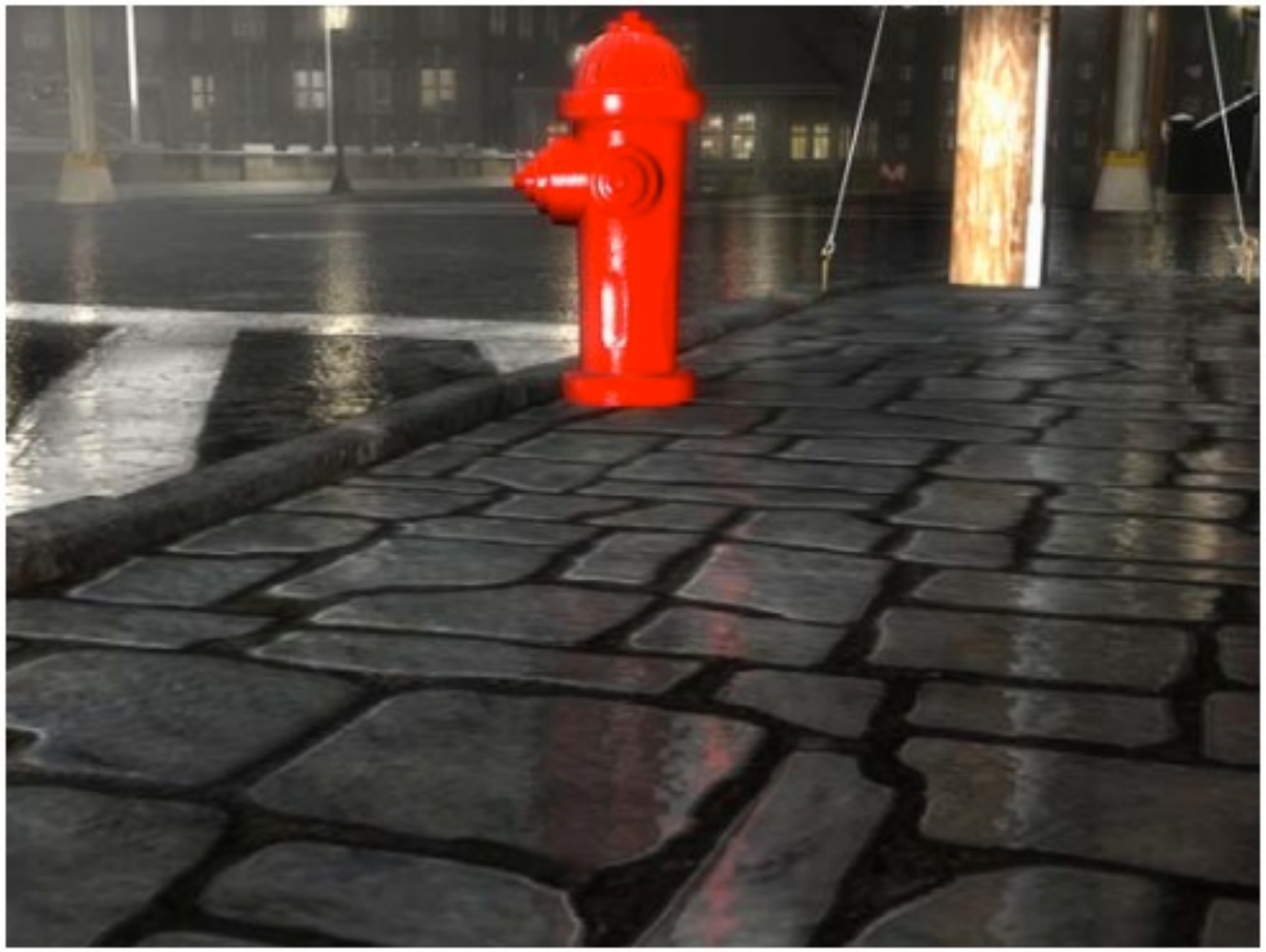
instead, effectively limiting the offset by the local displacement in height.

In texture coordinates, this corresponds to an offset of (he_t, he_b), scaled in each dimension according to the relationship between (t,b) and (u,v)

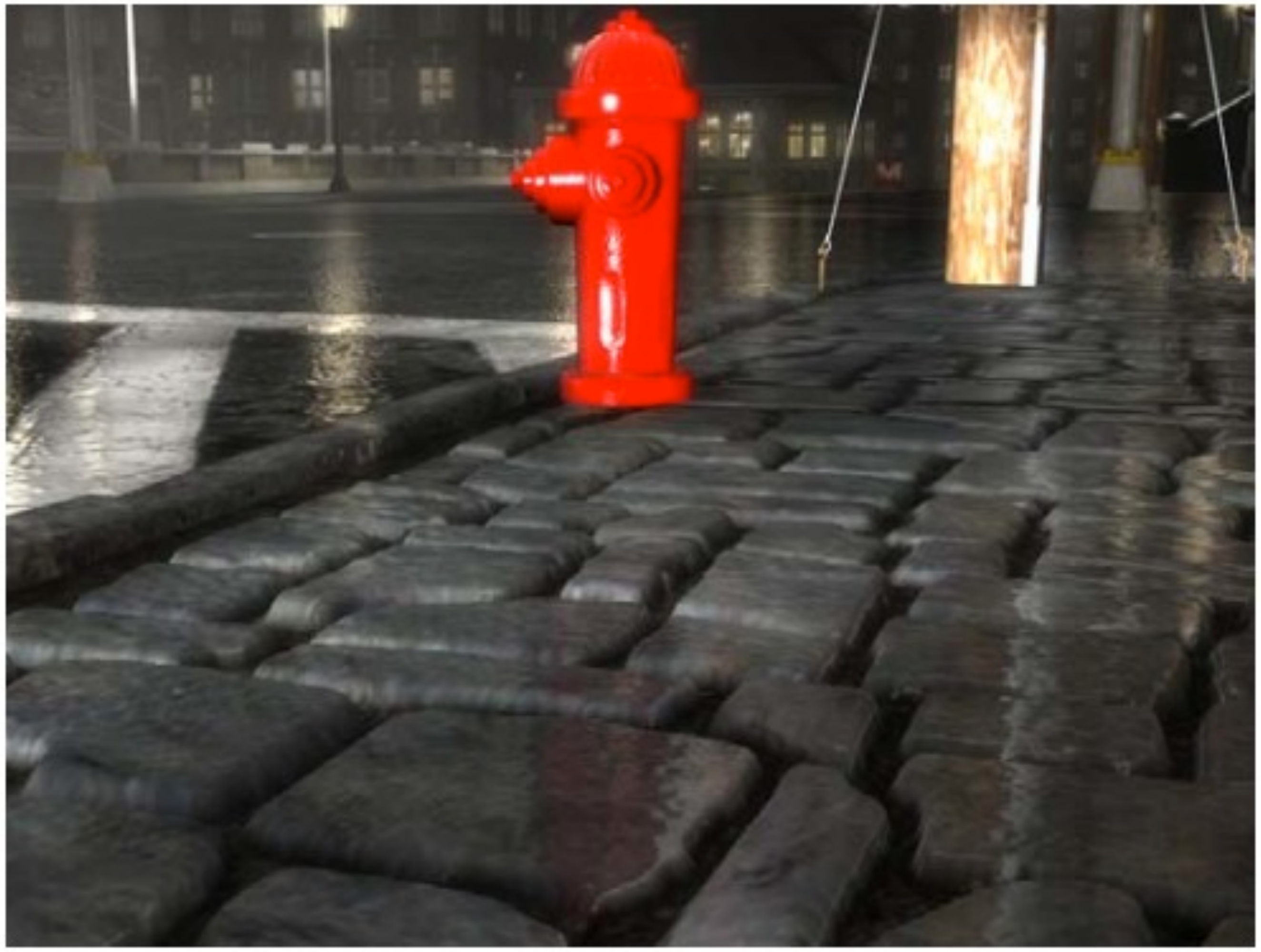
Parallax mapping

Image from Terry Welsh

Relief mapping



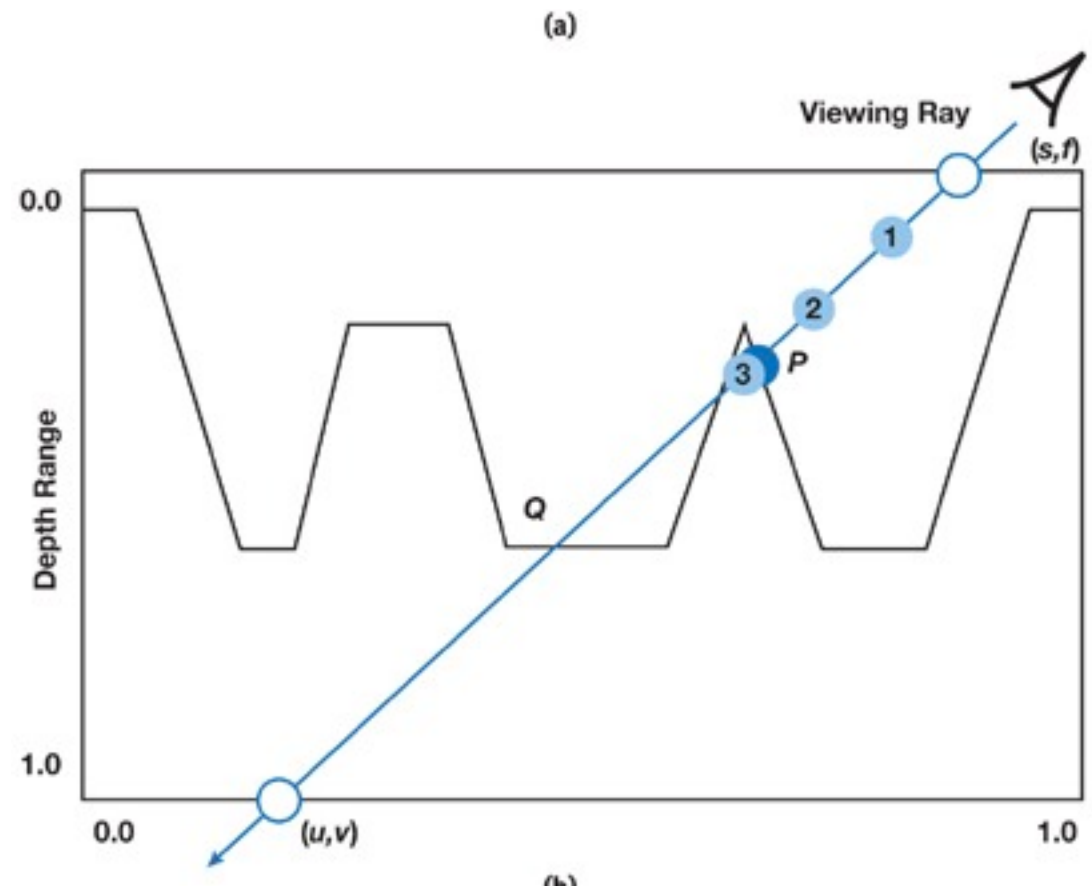
normal mapping



relief mapping

Image from Natalya Tatarchuk

Relief mapping



- Trace the eye ray into the bump map. A simple implementation can rasterize the projection of the ray onto the tangent plane, stepping along (e_t, e_b) and adjusting the height by a factor proportional to e_h.

Image from Policarpo and Oliveira (2008)

Relief mapping

Image from Policarpo et al. (2005)