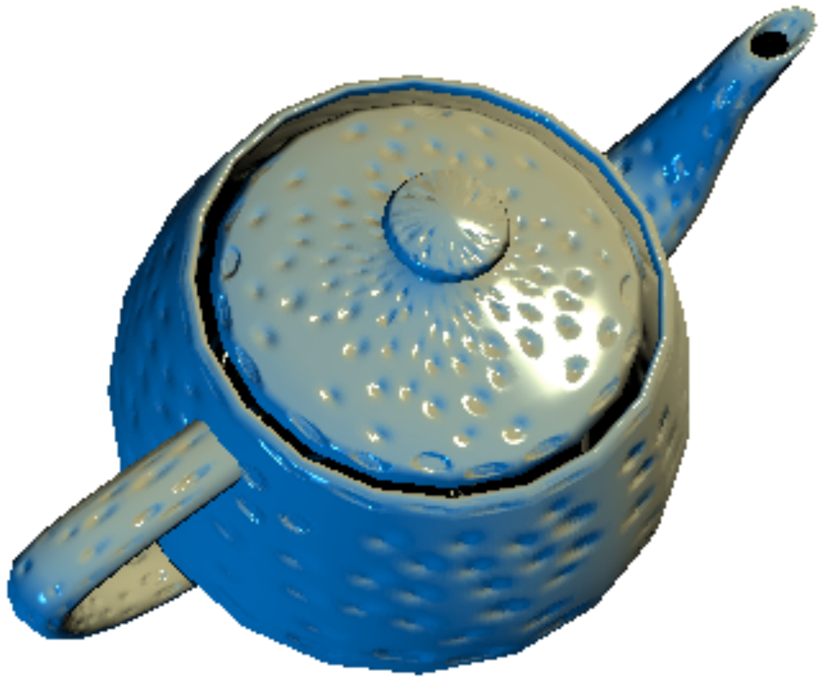
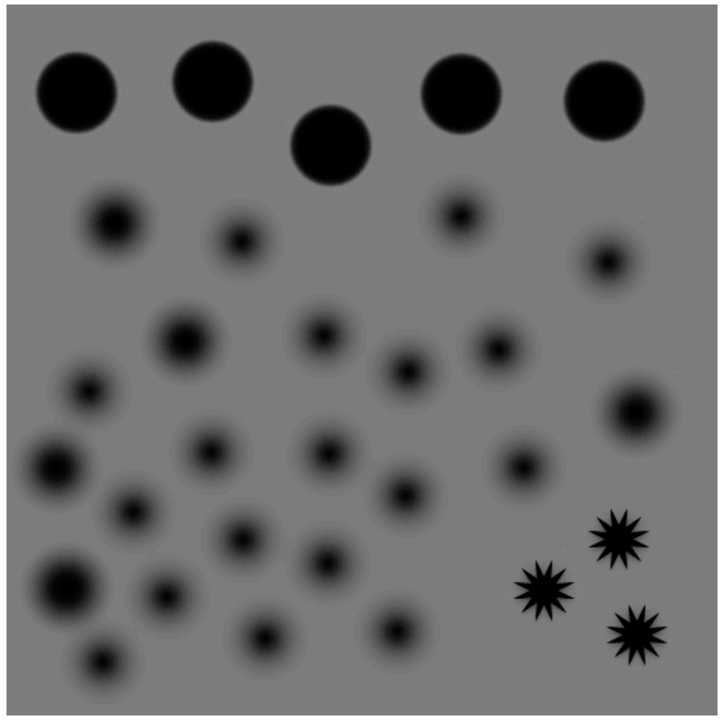
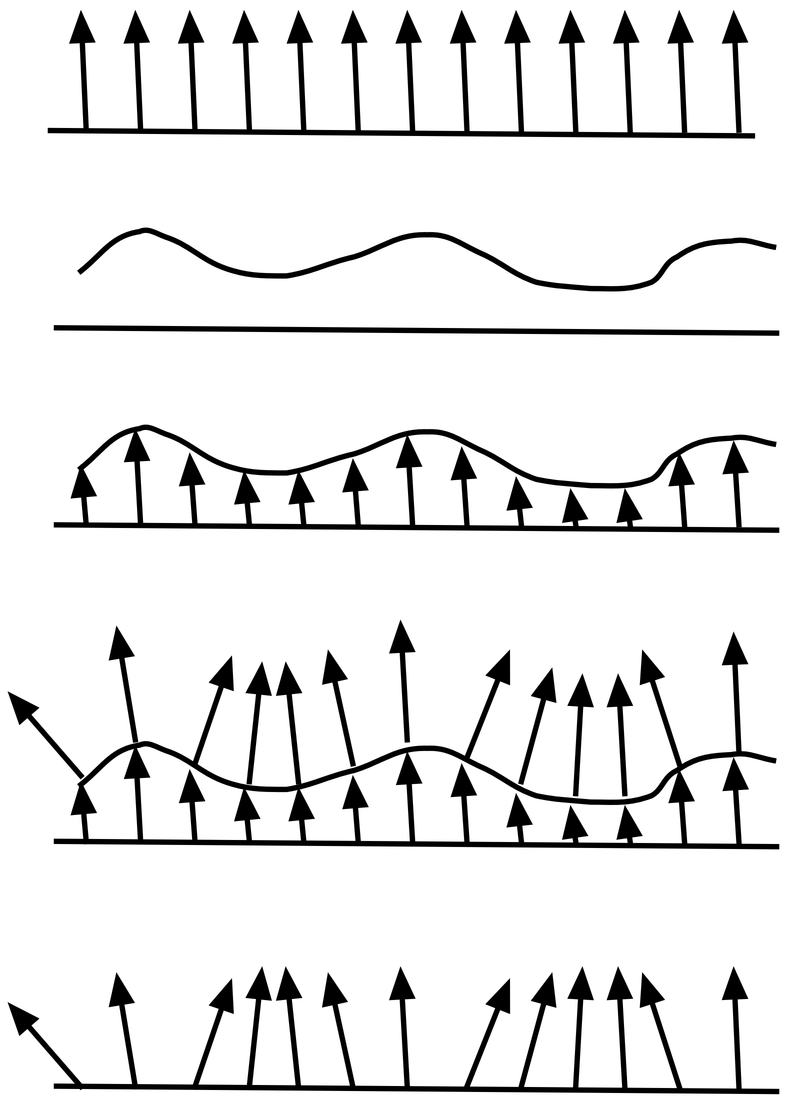
Bump mapping

Simulates surface structure by manipulating the normal vector





Bump mapping - model



Surface with normal vectors

Bump map: scalar function of the texture coordinates

Modulate the surface by the bump function, along normal

Calculate new normals

Resulting normal vectors

Bump mapping - the coordinate systems

Input:
A point **p**, normal vector **n**
Texture coordinates s(**p**), t(**p**)
Directions of texture coordinates **s**, **t**
The bump function b(s,t)

Calculate the partial derivative of the bump function, b_s and b_t

$$\mathbf{n}' = \mathbf{n} + b_t * (\mathbf{s} \times \mathbf{n}) + b_s * (\mathbf{t} \times \mathbf{n})$$

or, if **s**, **t**, **n** are orthogonal

$$\mathbf{n}' = \mathbf{n} + b_s * \mathbf{s} + b_t * \mathbf{t}$$

Texture coordinate system

How do we find the s and t vectors? We have the texture coordinates but no coordinate system!

Cross product with normal vector? With what?

Faking it

Cross product with absolutely anything!

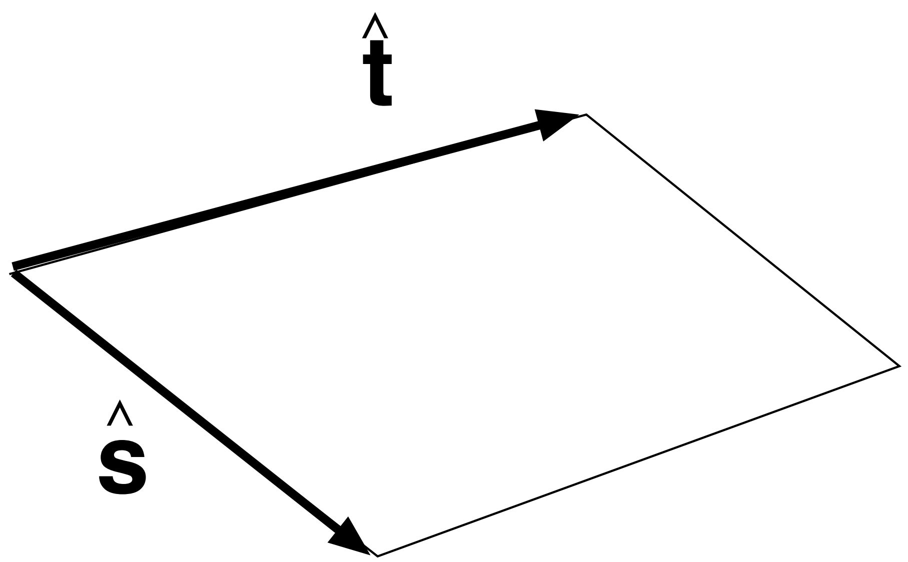
$$s = x \times n / |x \times n|$$
$$t = n \times s$$

Works for some cases. (Noise bump maps in particular.

But we can do better!

Trivial geometry

Very easy for a cube. Comfortable test case.



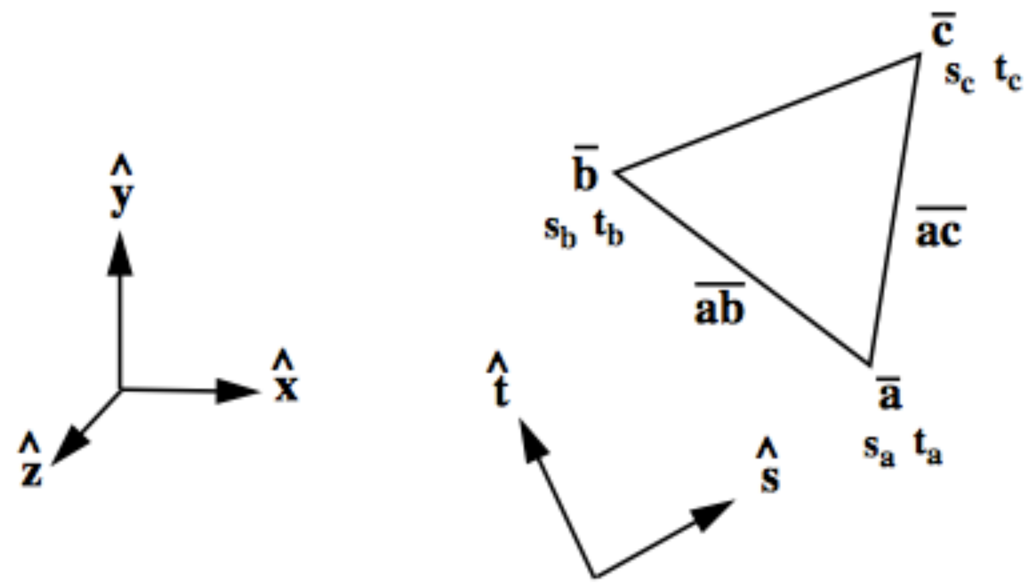
Lengyel's method

Derive through steps by s and t in xyz space

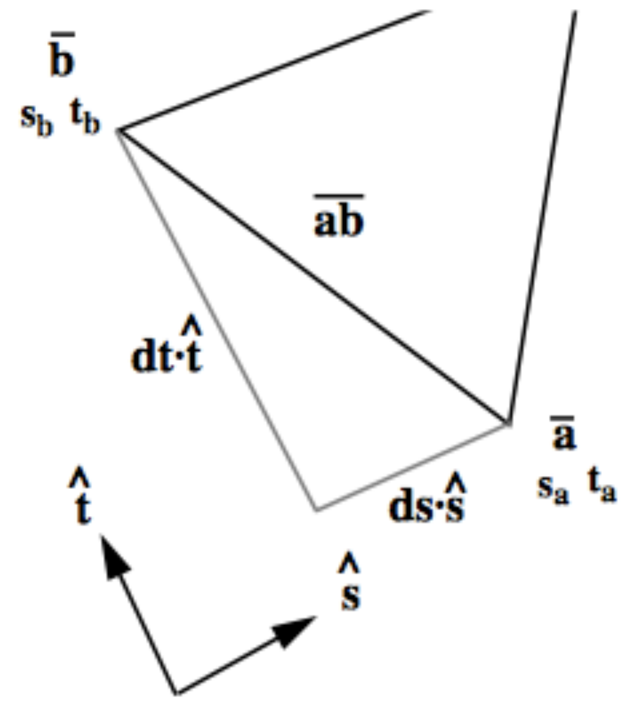
Straight and clean method using matrix algebra

Express two line segments as function of $\hat{s}$ and $\hat{t}$, find the inverse!

Lengyel's method



Given a triangle with texture coordinates, find basis vectors for texture coordinates!



Take edge ab, split to components along s and t. Express as matrix. Find s and t by matrix inverse!

Lengyel's method

in program code - fairly simple!

```
float ds1 = sb - sa; float ds2 = sc - sa;
float dt1 = tb - ta; float dt2 = tc - ta;
vec3 s, t;
float r = 1/(ds1 * dt2 - dt1 * ds2);
s = (ab * dt2 - ac * dt1) * r;
t = (ac * ds1 - ab * ds2) * r;
```

Note! Vector operations!

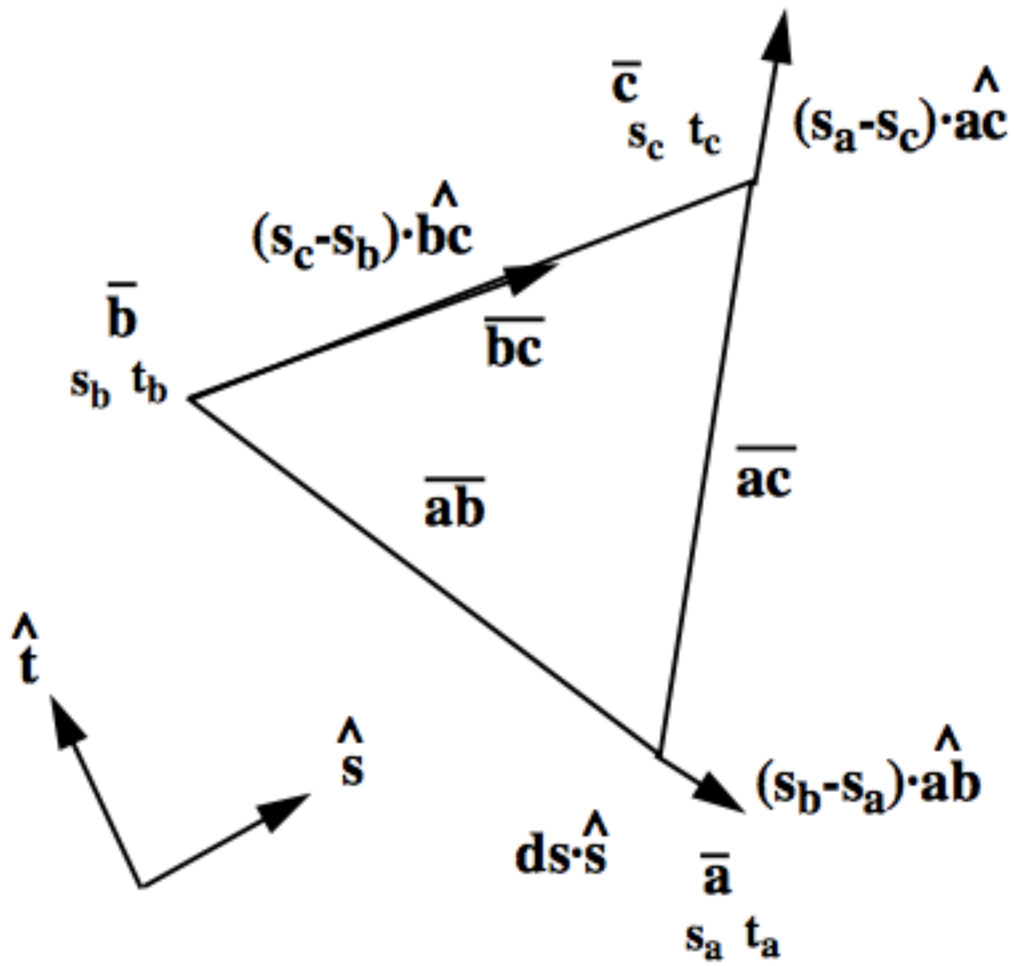
Approximative method

Let each edge of a polygon contribute to s and t depending on their variation in s and t!

Contribution to s from each edge = the edge direction normalized times the variation in s.

Approximative method

$$s_{ab} = \frac{ab}{|ab|}(s_b - s_a)$$



Both methods give good results for complicated models!