Given a quad with vertices $\vec{A}, \vec{B}, \vec{C}, \vec{D}$ in counter–clockwise ordering every point on the quad $\vec{X}$ can be described via bilinear interpolation:

$$\vec{X}(u,v) = (1-u)(1-v)\vec{A} + u(1-v)\vec{B} + uv\vec{C} + (1-u)v\vec{D} \text{ with } u,v \in [0,1]$$

For projecting an arbitrary point $\vec{Y}$ onto the quad, we need to solve the following optimization problem:

$$(u_0, v_0) = \underset{u,v\in[0,1]}{\arg\min} \parallel \vec{Y} - \vec{X}(u,v) \parallel^2$$

This gives the projected point $\vec{Y}_\perp = \vec{X}(u_0, v_0)$. For efficiently solving this problem we can use the Newton scheme: If one wants to minimize $f(x)$ one Newton–iteration has the following form:

1. set initial guess x_0.
2. with known x_k solve $H_f(x_k)\, v_k = -\nabla f(x_k)$ for v_k.
3. set $x_{k+1} = x_k + v_k$.

With

$$f(x) = f(u,v) = \left\langle \vec{Y} - \vec{X}(u,v), \vec{Y} - \vec{X}(u,v) \right\rangle$$

we get...ugly stuff...see matlab

```
Y=sym('Y',[3,1],'real');
x=sym('x',[2,1],'real');
A=sym('A',[3,1],'real');
B=sym('B',[3,1],'real');
C=sym('C',[3,1],'real');
D=sym('D',[3,1],'real');
X(x)=(1-x(1))*(1-x(2))*A+x(1)*(1-x(2))*B+x(1)*x(2)*C+(1-x(1))*x(2)*D;

f(x)=simplify((X-Y)'*(X-Y));
Df(x)=gradient(f);
Hf(x)=hessian(f);
```