Direct Surface Reconstruction for Structured Light

Morten Hannemose, Janus Nørtoft Jensen,
Anders Bjorholm Dahl, Jakob Wilm, Jeppe Revall Frisvad

Section for Image Analysis & Computer Graphics, Technical University of Denmark

Motivation

Traditional methods for surface reconstruction are based on computing a lot of points that lie on the object, and fitting a surface to these.

This has the following disadvantages:

- Information lost because surface fitted to points, not original data
- Tradeoff between smoothing and fitting well to points
- Difficult to get confidence intervals

Summary

We are working on a method to reconstruct a 3d mesh of an object, using images from a structured light scanner directly.

We do this by starting out with an initial guess of the geometry, which enables us to compute what the projected images should look like to the cameras.

With the real image and the rendering, we can compute a residual image per pattern and camera.

We then let an optimization algorithm to update the vertex positions of the mesh, with the goal of minimizing the sum of squared residuals.

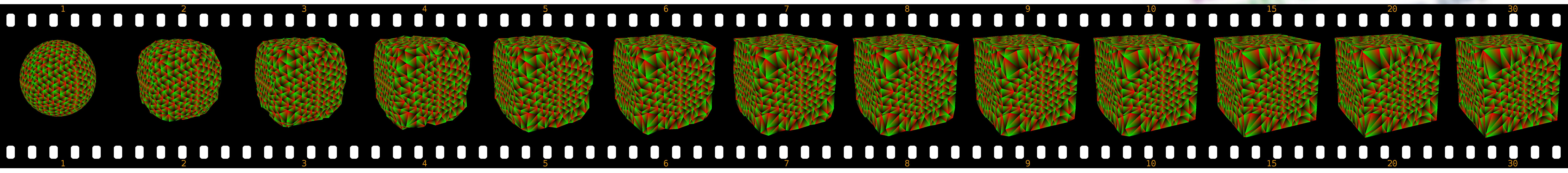

Figure 1. Sphere observed by fifty cameras is optimized to look like cube. Only loss is pixel differences. Iteration numbers in orange.

Method

The entire problem is formulated as a nonlinear least squares problem in terms of the vertex positions in the mesh, v.

We compute the difference between the rendering in the i'th pixel of the c'th camera, $f_{c,i}(v)$, and the value of the corresponding ground truth pixel.

Using closed form solutions for the analytical derivatives of $f_{c,i}(v)$, we are able to do fast gradient evaluations.

We have solved our problem using Levenberg–Marquardt, which gives much faster convergence than line search, even for our simple test cases.

$$OptimalMesh = \underset{v}{\operatorname{argmin}} \sum_{c=1}^{\#images} \sum_{i=1}^{\#pixels} \left(f_{c,i}(v) - GroundTruth_{c,i}\right)^2$$

Results

As can be seen in Figures 1-3, 5, our method is able to converge fully to the desired mesh in a simple case. The rendering of all fifty cameras, and the solving for a step size step is done on the GPU at a rate of nine iterations per second, which results in interactive performance.

The Sphere to Bunny experiment uses around fifty iterations to converge, but still has thin strands connecting the ears, which means this case needs remeshing in order to converge completely to the correct result.

Area penalty

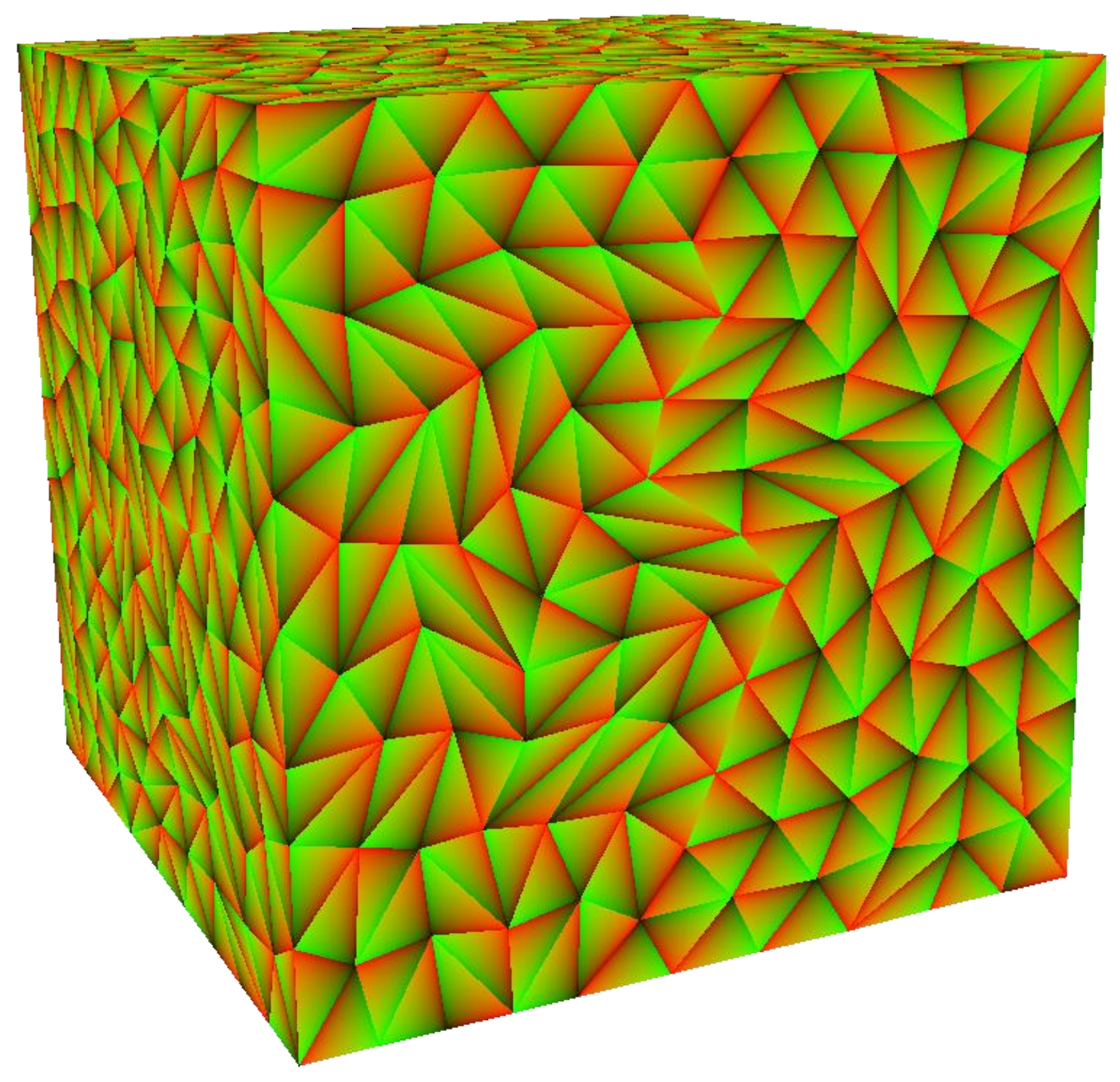

Figure 2. Sphere to cube with triangle area penalty

Edge length penalty

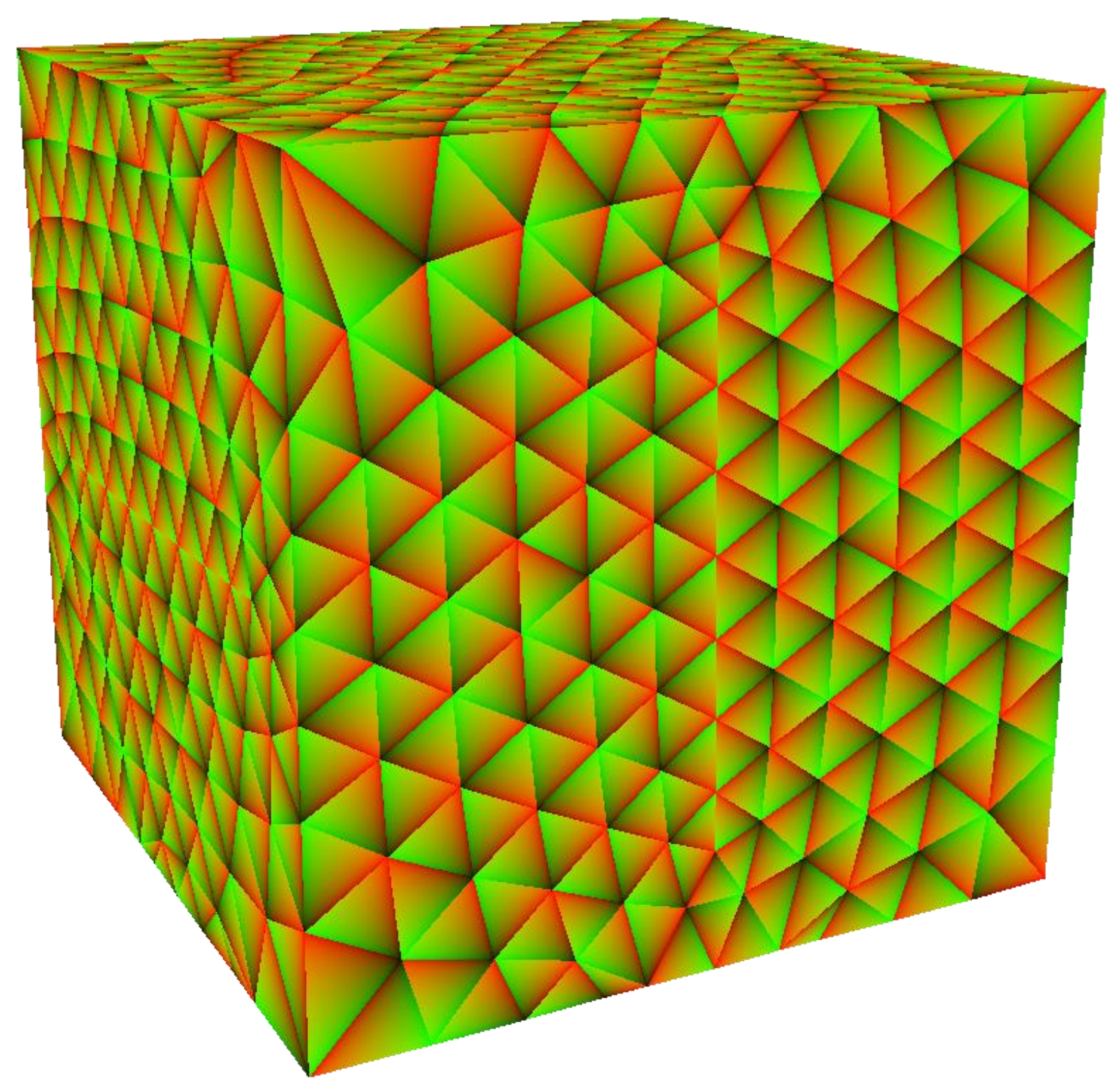

Figure 3. Sphere to cube with edge length penalty

Sphere to Bunny

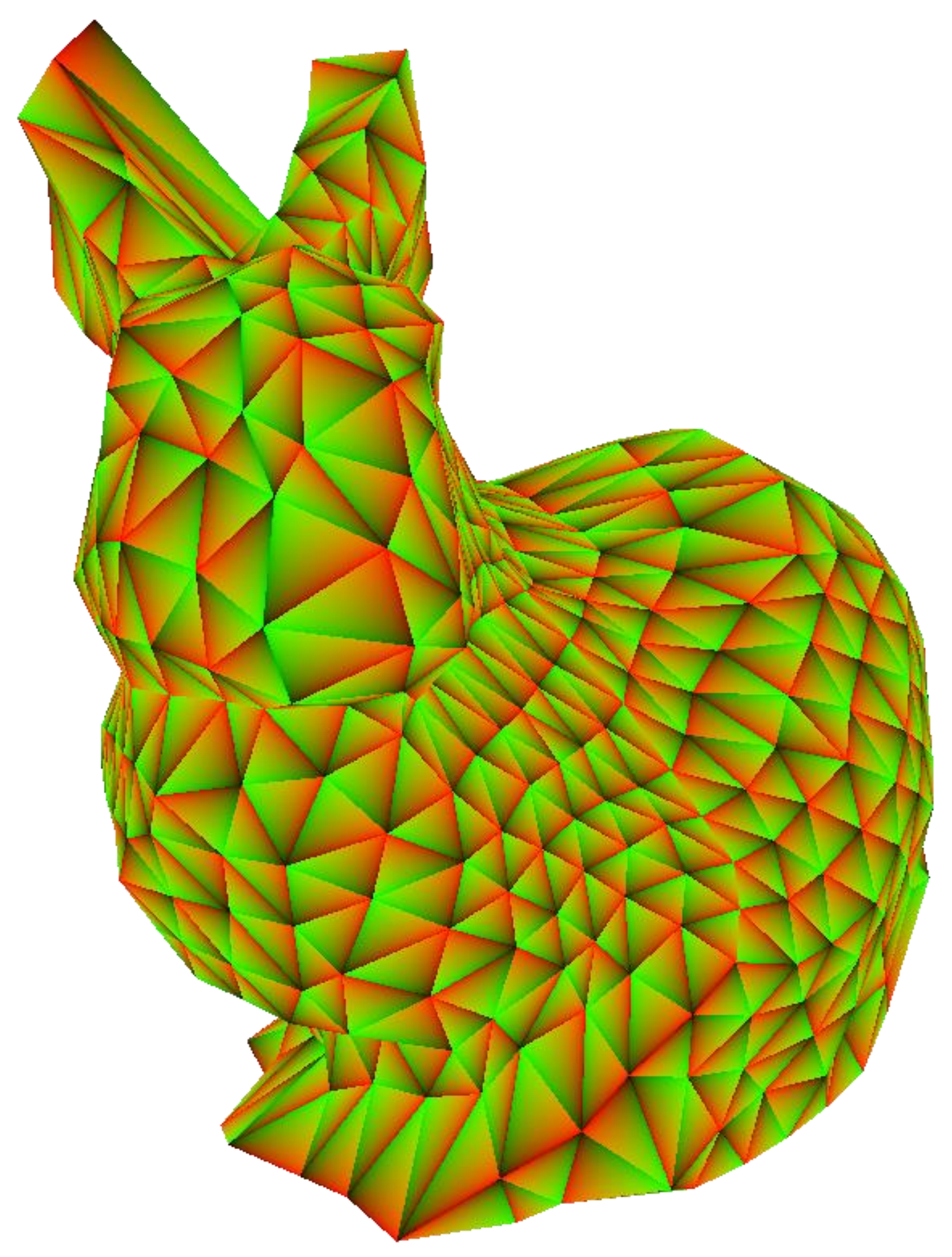

Figure 4. Sphere to bunny with triangle area and edge length penalty

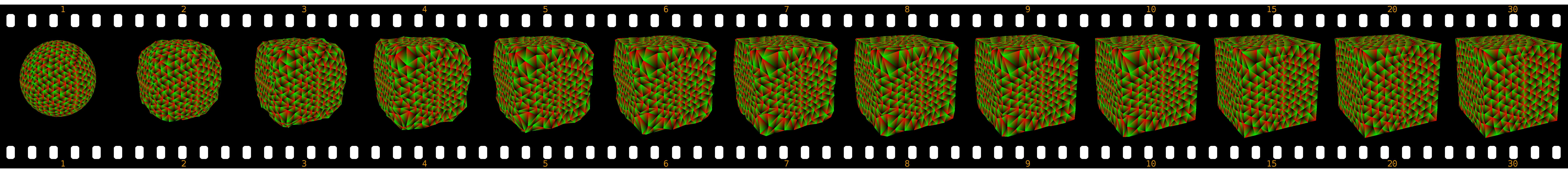

Figure 5. Sphere observed by fifty cameras is optimized to look like cube. Penalizing both pixel differences and edge lengths gives better final mesh. Iteration numbers in orange.

Mesh quality

Without any penalty on the mesh quality, the resulting meshes often have overlapping triangles, varying triangle sizes and sharp corners, which motivates us to penalize bad meshes.

We have attempted to regularize the mesh by penalizing the deviation of a triangle's area from the mean triangle area. This improves quality, but still yields very sharp angles and long edges, and triangles move around in rotation motions when it is applied, as seen in Figure 2.

Therefore we introduced a penalty on edge lengths, where each edge acts like a spring. The result of this can be seen in Figures 3-5.

In the future we hope to introduce methods for remeshing in our optimization pipeline, such that we every n iterations allow the insertion and removal of vertices.

This should preferably also be able to introduce and remove holes in the mesh, as we are unable to modify the topology of the starting mesh during the optimization.

Future work

It would be helpful to allow the model to modify the mesh by inserting and removing vertices from the mesh where this is needed.

In addition to this we're currently just projecting the projectors x-coordinate directly on the mesh. We will replace this with a phase shifting strategy which is more transferrable to a real camera-projector setup.

The method is currently also very sensitive to the starting mesh, and we hope to find a fast method that can create a rough version of the correct geometry, as our method just needs to be near the correct solution in order to converge.