Furniture Design from Pose

Devdeep Ray, Pratik Fegade, Siddhant Ranade, Anant Gupta

Introduction

The process of designing furniture can be thought of as an optimisation problem, where, for a given pose of a human being, say sitting, you try to optimise for the following:

1. Maximise the region of contact between the body and the surface - this can be thought of as a measure of "comfort".
2. Keep the top surface of the furniture as "smooth" as possible - you don't want sharp edges, this has to do with practicality (in terms of creation) and comfort.

However directly optimising for the region of contact - however that may be defined - using hard constraints will not yield a very solvable problem. We hence come up with the following problem statement.

Problem Statement and Our Work

We start with a skeleton in a sitting pose, and a height-field. The skeleton is held in a fixed position and the height field around it "evolves" into a chair. To evolve this chair, an error minimisation problem is set up with the following error components:

- Error for separation between the surface and points on the skeleton - this is a soft constraint for maximising the region of contact. This applies when the point on the skeleton is above the surface. Proportional to the vertical distance.
- Error for "collision" - you don't want a part of the skeleton to be *under* the surface. This soft constraint prevents that. Proportional to the vertical distance, but with a much larger weight.
- Error to ensure flatness - to try and prevent sharp edges. The region on which you sit should be as flat as possible - penalty proportional to the sum of absolute values of the laplacians at all points.

Finally, gradient descent is used to minimise this error function.

We also had another error metric: some prior knowledge about the surface, but we ended up not using it later.

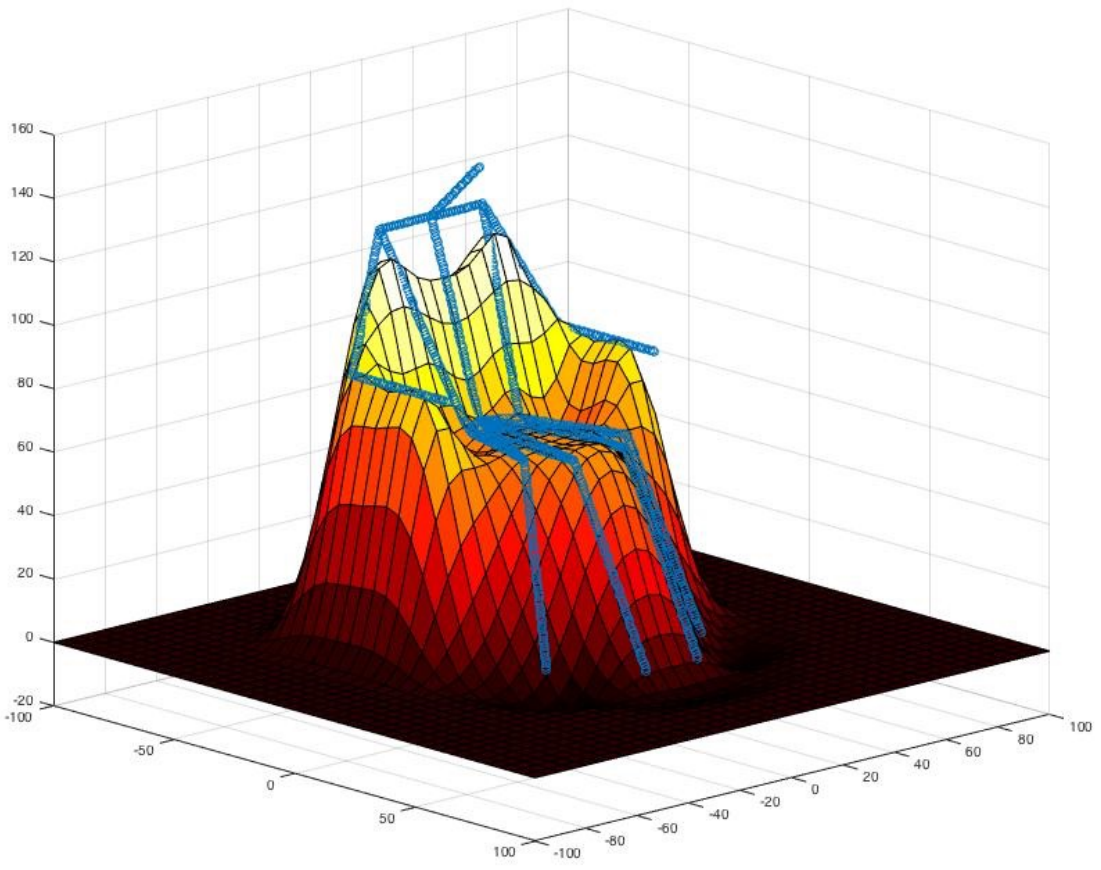
We created a "composite skeleton" with > 1 set of limbs to simulate the small movements that we make while sitting.

We also tried a "variant" of gradient descent - point-wise gradient descent - where instead of computing the whole gradient, you find find the change in error when varying one particular component, and then change only that component. This overfits single points on the skeleton and hence we switched back to the regular gradient descent.

For smoothness, we tried many different things:

- After each step of gradient descent, artificially smoothen surface.
- When computing gradients - do a “smooth perturbation” - add a (tiny) gaussian to the surface instead of an impulse for computing change in error. Use a smoothness error term that penalises large magnitudes of laplacian. This ensures that you always end up with a smooth surface, and the smoothness error doesn’t get out of hand. In this case, the gradient descent itself is also done the same way.
- Compute the regular gradient, without the smoothness term, but while doing the descent, add tiny Gaussians instead of impulses. This is what we used.

Results



Distribution of Work

- Creation of the skeletons - Pratik, Siddhant
- Code to parse skeleton in Matlab - Anant, Devdeep
- Code for gradient descent - Pratik, Devdeep
- Code for “smooth” gradient descent (modifying the gradient descent)- Siddhant
- Error function for flatness - Siddhant
- Error function for gaps, collisions - Devdeep, Siddhant
- Error function for prior information - Anant
- Code to parse skeleton in Python - Anant
- Surface plotting, updates - Anant
- Main function - Pratik, Devdeep, Siddhant
- Report - Siddhant