Evaluation of the Performance of Randomized FFD Control Grids

Master Thesis

at the

AG Computer Graphics

at the Faculty of Technology
of Bielefeld University

by

Stefan Dresselhaus

September 27, 2017

Supervisor:	Prof. Dr.	Mario Botsch
	Dipl. Math.	Alexander Richter

Contents

How to read this Thesis

As a guide through the nomenclature used in the formulas we prepend this chapter.

Unless otherwise noted the following holds:

- lowercase letters x,y,z
  refer to real variables and represent a point in 3D-Space.
- lowercase letters u,v,w
  refer to real variables between 0 and 1 used as coefficients in a 3D B-Spline grid.
- other lowercase letters
  refer to other scalar (real) variables.
- lowercase **bold** letters (e.g. **x**,**y**)
  refer to 3D coordinates
- uppercase **BOLD** letters (e.g. **D**, **M**)
  refer to Matrices

1 Introduction

In this Master Thesis we try to extend a previously proposed concept of predicting the evolvability of Freeform-Deformation (FFD) given a Deformation-Matrix[1]. In the original publication the author used random sampled points weighted with Radial Basis Function (RBF) to deform the mesh and defined three different criteria that can be calculated prior to using an evolutional optimisation algorithm to asses the quality and potential of such optimisation.

We will replicate the same setup on the same meshes but use Freeform-Deformation (FFD) instead of Radial Basis Function (RBF) to create a deformation and evaluate if the evolution-criteria still work as a predictor given the different deformation scheme.

1.1 What is Freeform-Deformation (FFD)?

First of all we have to establish how a FFD works and why this is a good tool for deforming meshes in the first place. For simplicity we only summarize the 1D-case from [4] here and go into the extension to the 3D case in chapter 2.1.

Given an arbitrary number of points p_i alongside a line, we map a scalar value $\tau_i \in [0,1[$ to each point with $\tau_i < \tau_{i+1} \forall i$. Given a degree of the target polynomial d we define the curve $N_{i,d,\tau_i}(u)$ as follows:

$$N_{i,0,\tau}(u) = \begin{cases} 1, & u \in [\tau_i, \tau_{i+1}[\\ 0, & \text{otherwise} \end{cases} \tag{1.1}$$

and

$$N_{i,d,\tau}(u) = \frac{u - \tau_i}{\tau_{i+d}} N_{i,d-1,\tau}(u) + \frac{\tau_{i+d+1} - u}{\tau_{i+d+1} - \tau_{i+1}} N_{i+1,d-1,\tau}(u) \tag{1.2}$$

If we now multiply every p_i with the corresponding $N_{i,d,\tau_i}(u)$ we get the contribution of each point p_i to the final curve-point parameterized only by $u \in [0,1[$. As can be seen from (1.2) we only access points $[i..i + d]$ for any given i[1], which gives us, in combination with choosing p_i and τ_i in order, only a local interference of $d + 1$ points.

[1] one more for each recursive step.

We can even derive this equation straightforward for an arbitrary N[2]:

$$\frac{\partial}{\partial u} N_{i,d,r}(u) = \frac{d}{\tau_{i+d} - \tau_i} N_{i,d-1,\tau}(u) - \frac{d}{\tau_{i+d+1} - \tau_{i+1}} N_{i+1,d-1,\tau}(u)$$

For a B-Spline

$$s(u) = \sum_i N_{i,d,\tau_i}(u) p_i$$

these derivations yield $\frac{\partial^d}{\partial u} s(u) = 0$.

Another interesting property of these recursive polynomials is that they are continuous (given $d \geq 1$) as every p_i gets blended in linearly between τ_i and τ_{i+d} and out linearly between τ_{i+1} and τ_{i+d+1} as can bee seen from the two coefficients in every step of the recursion.

1.1.1 Why is FFD a good deformation function?

The usage of FFD as a tool for manipulating follows directly from the properties of the polynomials and the correspondence to the control points. Having only a few control points gives the user a nicer high-level-interface, as she only needs to move these points and the model follows in an intuitive manner. The deformation is smooth as the underlying polygon is smooth as well and affects as many vertices of the model as needed. Moreover the changes are always local so one risks not any change that a user cannot immediately see.

But there are also disadvantages of this approach. The user loses the ability to directly influence vertices and even seemingly simple tasks as creating a plateau can be difficult to achieve[5, chapter 3.2][2].

This disadvantages led to the formulation of Direct Manipulation Freeform-Deformation (DM-FFD)[5, chapter 3.3] in which the user directly interacts with the surface-mesh. All interactions will be applied proportionally to the control-points that make up the parametrization of the interaction-point itself yielding a smooth deformation of the surface *at* the surface without seemingly arbitrary scattered control-points. Moreover this increases the efficiency of an evolutionary optimization[3], which we will use later on.

figure hier einfügen?

But this approach also has downsides as can be seen in [5, figure 7], as the tessellation of the invisible grid has a major impact on the deformation itself.

All in all FFD and DM-FFD are still good ways to deform a high-polygon mesh albeit the downsides.

[2] *Warning:* in the case of $d = 1$ the recursion-formula yields a 0 denominator, but N is also 0. The right solution for this case is a derivative of 0

1.2 What is evolutional optimization?

1.3 Wieso ist evo-Opt so cool?

The main advantage of evolutional algorithms is the ability to find optima of general functions just with the help of a given error-function (or fitness-function in this domain). This avoids the general pitfalls of gradient-based procedures, which often target the same error-function as an evolutional algorithm, but can get stuck in local optima.

This is mostly due to the fact that a gradient-based procedure has only one point of observation from where it evaluates the next steps, whereas an evolutional strategy starts with a population of guessed solutions. Because an evolutional strategy modifies the solution randomly, keeps the best solutions and purges the worst, it can also target multiple different hypothesis at the same time where the local optima die out in the face of other, better candidates.

If an analytic best solution exists (i.e. because the error-function is convex) an evolutional algorithm is not the right choice. Although both converge to the same solution, the analytic one is usually faster. But in reality many problems have no analytic solution, because the problem is not convex. Here evolutional optimization has one more advantage as you get bad solutions fast, which refine over time.

1.4 Evolvierbarkeitskriterien

- Konditionszahl etc.

2 Hauptteil

2.1 Was ist FFD?

- Definition
- Wieso Newton-Optimierung?
- Was folgt daraus?

2.2 Szenarien vorstellen

2.2.1 1D

Optimierungszenario

- Ebene -> Template-Fit

Matching in 1D

- Trivial

Besonderheiten der Auswertung

- Analytische Lösung einzig beste
- Ergebnis auch bei Rauschen konstant?
- normierter 1-Vektor auf den Gradienten addieren
  - Kegel entsteht

2.2.2 3D

Optimierungsszenario

- Ball zu Mario

Matching in 3D

- alternierende Optimierung

Besonderheiten der Optimierung

- Analytische Lösung nur bis zur Optimierung der ersten Punkte gültig
- Kriterien trotzdem gut

3 Evaluation

3.1 Spearman/Pearson-Metriken

- Was ist das?
- Wieso sollte uns das interessieren?
- Wieso reicht Monotonie?
- Haben wir das gezeigt?
- Stastik, Bilder, blah!

4 Schluss

HAHA .. als ob -.-

Appendix

Bibliography

[1] RICHTER, Andreas ; ACHENBACH, Jascha ; MENZEL, Stefan ; BOTSCH, Mario: Evolvability as a Quality Criterion for Linear Deformation Representations in Evolutionary Optimization. In: *IEEE Congress on Evolutionary Computation*, IEEE, 2016. – http://graphics.uni-bielefeld.de/publications/cec16.pdf, https://pub.uni-bielefeld.de/publication/2902698

[2] HSU, William M. ; HUGHES, John F. ; KAUFMAN, Henry: Direct Manipulation of Free-Form Deformations. In: *Computer Graphics* 26 (1992), 2. http://graphics.cs.brown.edu/~jfh/papers/Hsu-DMO-1992/paper.pdf

[3] MENZEL, Stefan ; OLHOFER, Markus ; SENDHOFF, Bernhard: Direct Manipulation of Free Form Deformation in Evolutionary Design Optimisation. In: *Proceedings of the 9th International Conference on Parallel Problem Solving from Nature*. Berlin, Heidelberg : Springer-Verlag, 2006 (PPSN'06). – ISBN 3–540–38990–3, 978–3–540–38990–3, 352–361

[4] SPITZMÜLLER, Klaus: Partial derivatives of Bèzier surfaces. In: *Computer-Aided Design* 28 (1996), Nr. 1, 67–72. https://doi.org/10.1016/0010-4485(95)00044-5

[5] HSU, William M.: A direct manipulation interface to free-form deformations. In: *Master's thesis, Brown University* (1991). https://cs.brown.edu/research/pubs/theses/masters/1991/hsu.pdf

Abbreviations

FFD Freeform-Deformation

DM-FFD Direct Manipulation Freeform-Deformation

RBF Radial Basis Function

List of Algorithms

List of Figures

List of Tables

Todo list

Erklärung

I hereby declare that this thesis is my own work and effort. Where other sources of information have been used, they have been acknowledged. blah blah

Bielefeld, den September 27, 2017 . . . .

Stefan Dresselhaus