# 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

October 2, 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. $\mathbf{D}, \mathbf{M}$)
  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 optimization algorithm to asses the quality and potential of such optimization.

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 [3] here and go into the extension to the 3D case in chapter 2.2.

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 $\tau_i$ in order, only a local interference of $d+1$ points.

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 $\tau_i$ and $\tau_{i+d}$ and out linearly between $\tau_{i+1}$ and $\tau_{i+d+1}$ 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

[1] one more for each recursive step.

[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

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[6], which we will use later on.

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.

**Unsure:** figure hier einfügen?

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

## 1.2 What is evolutional optimization?

**Change:** Write this section

## 1.3 Advantages of evolutional algorithms

**Change:** Needs citations

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 Criteria for the evolvability of linear deformations

### 1.4.1 Variability

In [1] variability is defined as

$$V(\mathbf{U}) := \frac{\text{rank}(\mathbf{U})}{n},$$

whereby $\mathbf{U}$ is the $m \times n$ deformation-Matrix used to map the $m$ control points onto the $n$ vertices.

Given $n = m$, an identical number of control-points and vertices, this quotient will be $= 1$ if all control points are independent of each other and the solution is to trivially move every control-point onto a target-point.

In praxis the value of $V(\mathbf{U})$ is typically $\ll 1$, because as there are only few control-points for many vertices, so $m \ll n$.

Additionally in our setup we connect neighbouring control-points in a grid so each control point is not independent, but typically depends on $4^d$ control-points for an $d$-dimensional control mesh.

### 1.4.2 Regularity

Regularity is defined[1] as

$$R(\mathbf{U}) := \frac{1}{\kappa(\mathbf{U})} = \frac{\sigma_{min}}{\sigma_{max}}$$

where $\sigma_{min}$ and $\sigma_{max}$ are the smallest and greatest right singular value of the deformation-matrix $\mathbf{U}$.

As we deform the given Object only based on the parameters as $\mathbf{p} \mapsto f(\mathbf{x} + \mathbf{U}\mathbf{p})$ this makes sure that $\|\mathbf{U}\mathbf{p}\| \propto \|\mathbf{p}\|$ when $\kappa(\mathbf{U}) \approx 1$. The inversion of $\kappa(\mathbf{U})$ is only performed to

map the criterion-range to $[0..1]$, whereas $1$ is the optimal value and $0$ is the worst value.

This criterion should be characteristic for numeric stability on the on hand[4, chapter 2.7] and for convergence speed of evolutional algorithms on the other hand[1] as it is tied to the notion of locality[8, 7].

### 1.4.3 Improvement Potential

In contrast to the general nature of variability and regularity, which are agnostic of the fitness-function at hand the third criterion should reflect a notion of potential.

As during optimization some kind of gradient $g$ is available to suggest a direction worth pursuing we use this to guess how much change can be achieved in the given direction.

The definition for an improvement potential $P$ is[1]:

$$P(\mathbf{U}) := 1 - \|(\mathbf{1} - \mathbf{U}\mathbf{U}^{+})(G)\|_F^2$$

given some approximate $n \times d$ fitness-gradient $\mathbf{G}$, normalized to $\|\mathbf{G}\|_F = 1$, whereby $\|\cdot\|_F$ denotes the Frobenius-Norm.

# 2 Implementation of Freeform-Deformation (FFD)

As general B-Splines have a free parameters $d$ and $\tau$.

As we usually work with regular grids in our FFD we define $\tau$ statically as

$$\tau_i = {}^{i}/_{n}$$

whereby $n$ is the number of control-points in that direction.

$d$ defines the *degree* of the B-Spline-Function (the number of times this function is differentiable) and for our purposes we fix $d$ to 3, but give the formulas for the general case so it can be adapted quite freely.

## 2.1 Adaption of FFD

As we have established in Chapter 1.1 we can define an FFD-displacement as

$$\Delta_x(u) = \sum_i N_{i,d,\tau_i}(u)\Delta_x c_i \tag{2.1}$$

Note that we only sum up the $\Delta$-displacements in the control points $c_i$ to get the change in position of the point we are interested in.

In this way every deformed vertex is defined by

$$\text{Deform}(v_x) = v_x + \Delta_x(u)$$

with $u \in [0..1[$ being the variable that connects the high-detailed vertex-mesh to the low-

detailed control-grid. To actually calculate the new position of the vertex we first have to calculate the $u$-value for each vertex. This is achieved by finding out the parametrization of $v$ in terms of $c_i$

$$v_x = \sum_i N_{i,d,\tau_i}(u)c_i$$

As the B-Spline-functions are smooth and convex we just derive by $u$ yielding

$$\begin{aligned} &\frac{\partial}{\partial u} \quad v_x - \sum_i N_{i,d,\tau_i}(u)c_i \\ &= \quad v_x - \sum_i \left( \frac{d}{\tau_{i+d} - \tau_i} N_{i,d-1,\tau}(u) - \frac{d}{\tau_{i+d+1} - \tau_{i+1}} N_{i+1,d-1,\tau}(u) \right) c_i \end{aligned}$$

and do a gradient-descend to approximate the value of $u$ up to an $\varepsilon$ of $0.0001$.

For this we use the Gauss-Newton algorithm[9] as the solution to this problem may not be deterministic, because we usually have way more vertices than control points ($\#v \gg \#c$).

## 2.2 Adaption of FFD for a 3D-Mesh

This is a straightforward extension of the 1D-method presented in the last chapter. But this time things get a bit more complicated. As we have a 3-dimensional grid we may have a different amount of control-points in each direction.

Given $n$,$m$,$o$ control points in $x$,$y$,$z$-direction each Point on the curve is defined by

$$V(u,v,w) = \sum_{i=0}^{n-d-2} \sum_{j=0}^{m-d-2} \sum_{k=0}^{o-d-2} N_{i,d,\tau_i}(u) N_{j,d,\tau_j}(v) N_{k,d,\tau_k}(w) \cdot C_{ijk}.$$

In this case we have three different B-Splines (one for each dimension) and also 3 variables $u$,$v$,$w$ for each vertex we want to approximate.

Given a target vertex $\mathbf{p}^*$ and an initial guess $\mathbf{p} = V(u,v,w)$ we define the error-function for the gradient-descent as:

$$Err(u,v,w,\mathbf{p}^*) = \mathbf{p}^* - V(u,v,w)$$

And the partial version for just one direction as

$$Err_x(u,v,w,\mathbf{p}^*) = p_x^* - \sum_{i=0}^{n-d-2} \sum_{j=0}^{m-d-2} \sum_{k=0}^{o-d-2} C_{ijk_x} N_{i,d,\tau_i}(u) N_{j,d,\tau_j}(v) N_{k,d,\tau_k}(w)$$

To solve this we derive partially, like before:

$$\begin{aligned} \frac{\partial Err_x}{\partial u} \quad & p_x^* - \sum_{i=0}^{n-d-2} \sum_{j=0}^{m-d-2} \sum_{k=0}^{o-d-2} C_{ijk_x} N_{i,d,\tau_i}(u) N_{j,d,\tau_j}(v) N_{k,d,\tau_k}(w) \\ = \quad & - \sum_{i=0}^{n-d-2} \sum_{j=0}^{m-d-2} \sum_{k=0}^{o-d-2} C_{ijk_x} N_i'(u) N_{j,d,\tau_j}(v) N_{k,d,\tau_k}(w) \end{aligned}$$

The other partial derivatives follow the same pattern yielding the Jacobian:

$$J(Err(u,v,w)) = \begin{pmatrix} \frac{\partial Err_x}{\partial u} & \frac{\partial Err_x}{\partial v} & \frac{\partial Err_x}{\partial w} \\ \frac{\partial Err_y}{\partial u} & \frac{\partial Err_y}{\partial v} & \frac{\partial Err_y}{\partial w} \\ \frac{\partial Err_z}{\partial u} & \frac{\partial Err_z}{\partial v} & \frac{\partial Err_z}{\partial w} \end{pmatrix}$$

**Unsure:** Should I add an informal complete derivative?
Like leaving out Sums $i,j,k$-Indices to make obvious what derivative belongs where in what case?

With the Gauss-Newton algorithm we iterate the formula

$$J(Err(u,v,w)) \cdot \Delta \begin{pmatrix} u \\ v \\ w \end{pmatrix} = -Err(u,v,w)$$

and use Cramers rule for inverting the small Jacobian and solving this system of linear equations.

## 2.3 Parametrisierung sinnvoll?

- Nachteile von Parametrisierung

- Deformation ist um einen Kontrollpunkt viel direkter zu steuern.
- => DM-FFD?

## 2.4 Test Scenario: 1D Function Approximation

### 2.4.1 Optimierungszenario

- Ebene -> Template-Fit

### 2.4.2 Matching in 1D

- Trivial

### 2.4.3 Besonderheiten der Auswertung

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

## 2.5 Test Scenario: 3D Function Approximation

### 2.5.1 Optimierungsszenario

- Ball zu Mario

### 2.5.2 Matching in 3D

- alternierende Optimierung

### 2.5.3 Besonderheiten der Optimierung

- Analytische Lösung nur bis zur Optimierung der ersten Punkte gültig

- Kriterien trotzdem gut

# 3 Evaluation of Scenarios

## 3.1 Spearman/Pearson-Metriken

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

## 3.2 Results of 1D Function Approximation

## 3.3 Results of 3D Function Approximation

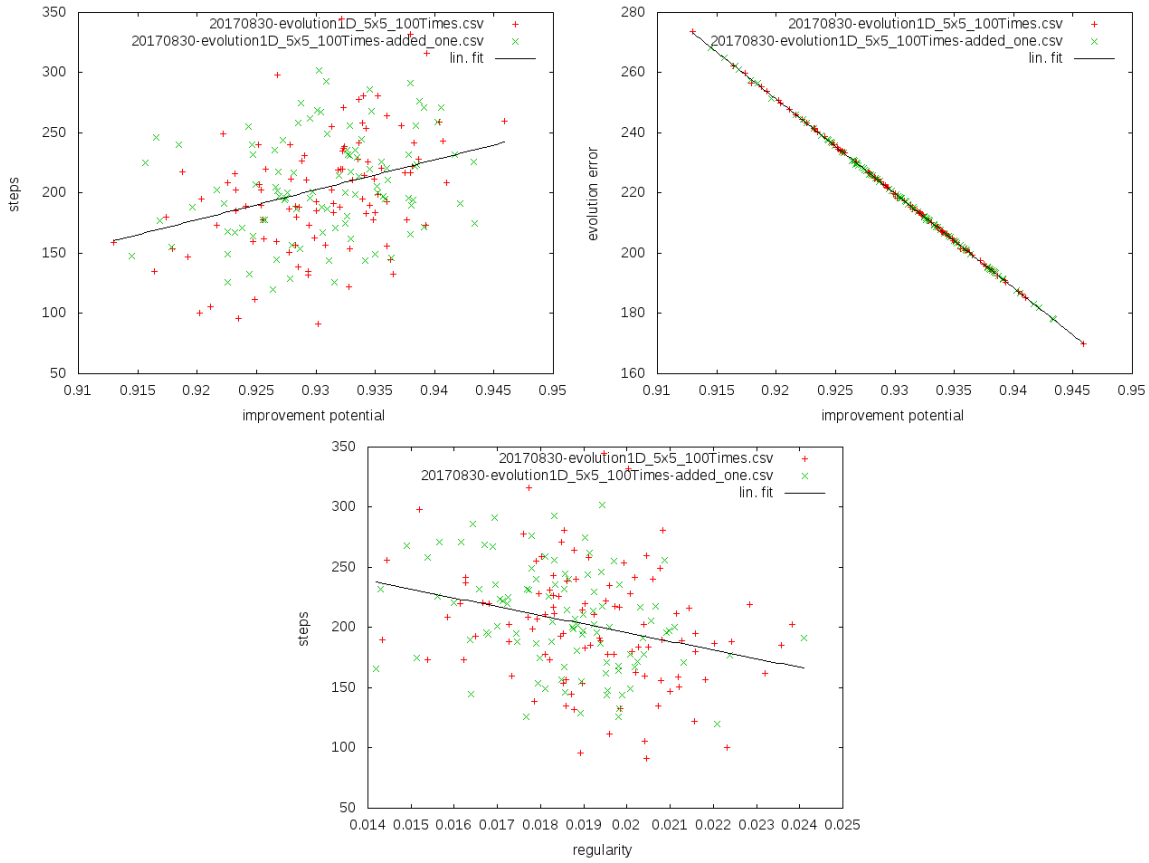


**Figure 3.1:** Results 1D

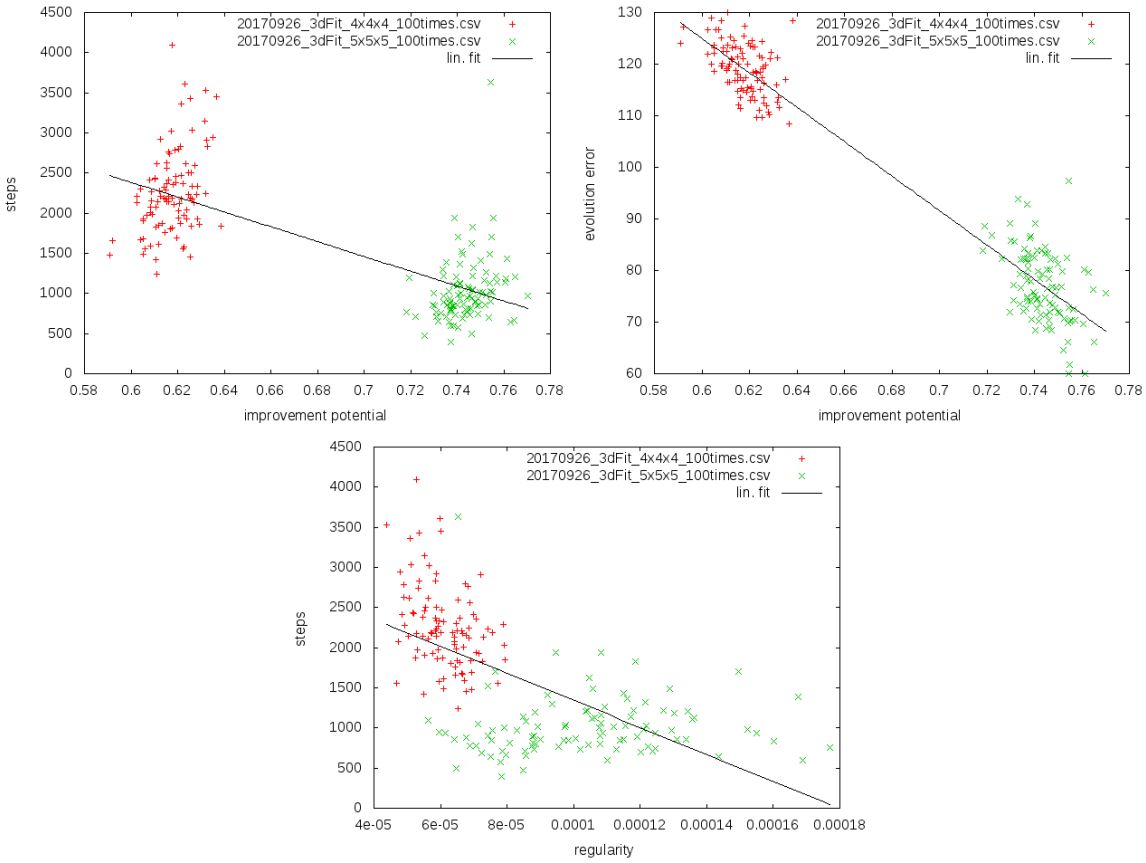


**Figure 3.2:** Results 3D

# 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] 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

[4] GOLUB, Gene H. ; VAN LOAN, Charles F.: *Matrix computations*. Bd. 3. JHU Press, 2012

[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

[6] 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

[7] THORHAUER, Ann ; ROTHLAUF, Franz: On the locality of standard search operators in grammatical evolution. In: *International Conference on Parallel Problem Solving from Nature* Springer, 2014, 465–475

[8] WEISE, Thomas ; CHIONG, Raymond ; TANG, Ke: Evolutionary Optimization: Pitfalls and Booby Traps. In: *J. Comput. Sci. & Technol* 27 (2012), Nr. 5. http://jcst.ict.ac.cn:8080/jcst/EN/article/downloadArticleFile.do?attachType=PDF&id=9543

[9] MARQUARDT, Donald W.: An Algorithm for Least-Squares Estimation of Nonlinear Parameters. In: *Journal of the Society for Industrial and Applied Mathematics* 11 (1963), Nr. 2, 431-441. http://dx.doi.org/10.1137/0111030. – DOI 10.1137/0111030

# 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 October 2, 2017 .....

Stefan Dresselhaus