Computer Aided Geometric Design 44 (2016) 1–9

Contents lists available at ScienceDirect

Computer Aided Geometric Design

www.elsevier.com/locate/cagd

Birational 2D Free-Form Deformation of degree $1 \times n$ ☆

Thomas W. Sederberg [a,*], Ronald N. Goldman [b], Xuhui Wang [c]

[a] Department of Computer Science, Brigham Young University, Provo, UT 84602, USA
[b] Department of Computer Science, Rice University, Houston, TX 77251, USA
[c] School of Mathematics, Hefei University of Technology, Hefei, Anhui 230009, China

ARTICLE INFO

Article history:
Received 8 December 2015
Received in revised form 20 February 2016
Accepted 22 February 2016
Available online 10 March 2016

Keywords:
Free-Form Deformation
Birational map

ABSTRACT

This paper shows that generic 2D-Free-Form Deformations of degree $1 \times n$ can be made birational by a suitable assignment of weights to the Bézier or B-spline control points. An FFD that is birational facilitates operations such as backward mapping for image warping, and Extended Free-Form Deformation. While birational maps have been studied extensively in the classical algebraic geometry literature, this paper is the first to present a family of non-linear birational maps that are flexible enough to be of practical use in geometric modeling.

© 2016 Elsevier B.V. All rights reserved.

1. Introduction

Free-Form Deformation (FFD) is an artist-friendly tool for deforming geometric models and graphical images by warping the ambient space (Sederberg and Parry, 1986). FFD is used in numerous applications such as geometric modeling (Hoffmann, 1989), shape optimization (Manzoni et al., 2012), character animation (Chadwick et al., 1989), facial animation (Kalra et al., 1992), image morphing (Lee et al., 1995), medical imaging (Frangi et al., 2001), shape registration (Huang et al., 2006), and sheet metal forming (Lingbeek et al., 2005).

2D-FFD is illustrated in Fig. 1. As a preprocess, the scene in Fig. 1.a is enclosed in a rectangle (Fig. 1.b) which is parametrized by $(s,t) = (\frac{x-x_0}{x_1-x_0}, \frac{y-y_0}{y_1-y_0})$. To explain the 2D-FFD illustrated in Fig. 1.c, we introduce the following notation. Upper-case bold letters denote elements of $\mathbb{P}^2$ and bold lower-case letters denote elements of $\mathbb{R}^2$. Given $\mathbf{P} = (P_x, P_y, P_w)$, the projection operator $\Pi : \mathbb{P}^2 \to \mathbb{R}^2$ is defined $\Pi(\mathbf{P}) = (P_x/P_w, P_y/P_w)$. The result of projection is denoted by the corresponding lower-case letter, for example, $\mathbf{p} = \Pi(\mathbf{P})$. Define

$$\mathbf{P}(s,t) = (p_x(s,t), p_y(s,t), p_w(s,t)) = \sum_{i=0}^{m}\sum_{j=0}^{n} w_{ij}\mathbf{P}_{ij}B_i^m(s)B_j^n(t) \tag{1}$$

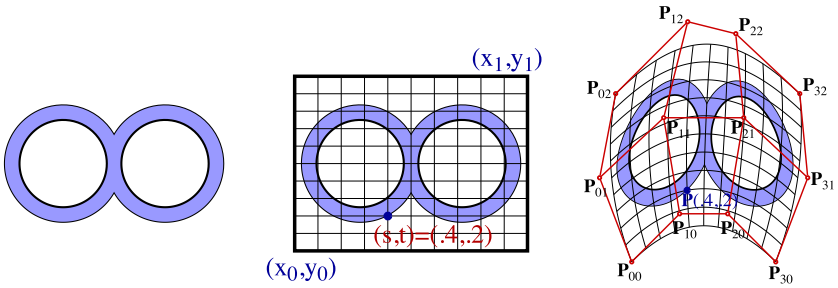
where control points $\mathbf{P}_{ij} = (x_{ij}, y_{ij}, 1)$ have $\mathbf{p}_{ij} = (x_{ij}, y_{ij})$ Cartesian coordinates, w_{ij} are weights, and $B_k^d(u)$, $u \in \{s,t\}$, is the kth Bernstein polynomial of degree $d \in \{m,n\}$. The FFD is given by

$$(x,y) = \mathbf{p}(s,t) = \Pi(\mathbf{P}(s,t)) = \left(\frac{p_x(s,t)}{p_w(s,t)}, \frac{p_y(s,t)}{p_w(s,t)}\right). \tag{2}$$

☆ This paper has been recommended for acceptance by Kai Hormann.

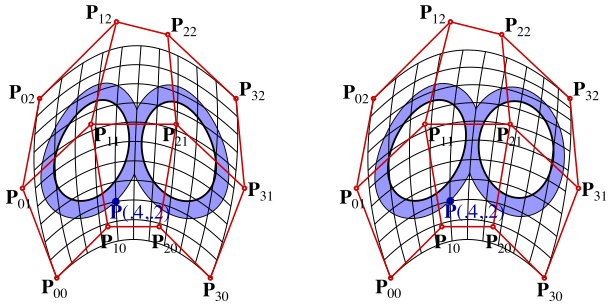
* Corresponding author.
E-mail address: tom@cs.byu.edu (T.W. Sederberg).

http://dx.doi.org/10.1016/j.cagd.2016.02.020
0167-8396/© 2016 Elsevier B.V. All rights reserved.



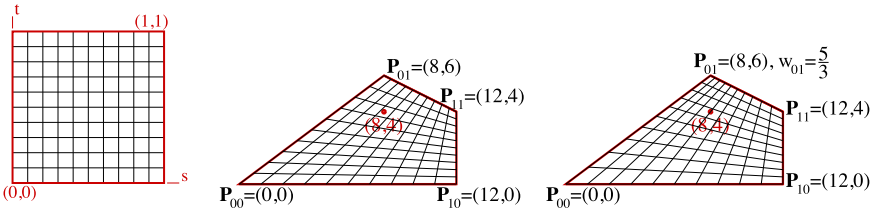
(a) Initial Geometry (b) Bounding Rectangle (c) FFD applied

Fig. 1. 2D FFD of bidegree 3×2. All control point weights are 1.



(a) All $w_{ij} = 1$ (b) $w_{32} = 2$

Fig. 2. Rational 2D FFD. All weights are 1 except w_{32}.



(a) Domain (s, t) (b) Bilinear FFD, weights all one (c) $w_{01} = \frac{5}{3}$

Fig. 3. 2D FFD of bidegree 1×1.

Artists typically manipulate an FFD by moving the control points. Changing the weights w_{ij} will also modify an FFD, but this is rarely done in practice because the influence of the weights is limited—for non-negative weights, the deformation will never escape the convex hull of the control points. Fig. 2 shows the subtle effect of changing weight w_{32} in the FFD in Fig. 1.c from one to two.

We define a *preimage* of a given (x, y) to be any (s, t) for which $\mathbf{p}(s, t) = (x, y)$. Computing FFD preimages is *required* in many applications — such as Extended FFD (Coquillart, 1990) and Directly Manipulated FFD (Gain and Dodgson, 2001) — and is *useful* in other applications, such as backward mapping for image warping.

A preimage (s, t) of a given (x, y) satisfies the equations

$$g_1(s,t) = x p_w(s,t) - p_x(s,t) = 0 \tag{3}$$

$$g_2(s,t) = y p_w(s,t) - p_y(s,t) = 0. \tag{4}$$

Computing preimages usually involves solving (3) and (4) using numerical methods such as multivariate Newton iteration, or using algebraic methods such as resultants or Gröbner bases. In the simple case of a bidegree 1×1 FFD, (3) and (4) can be solved using algebraic substitution. For the FFD in Fig. 3,

$$x = 12s(1-t) + 8(1-s)t + 12st$$

$$y = 6(1-s)t + 4st$$

from which we can solve

Download English Version:

https://daneshyari.com/en/article/441108

Download Persian Version:

https://daneshyari.com/article/441108

Daneshyari.com