Computer-Aided Design 46 (2014) 200–204

Contents lists available at ScienceDirect

Computer-Aided Design

journal homepage: www.elsevier.com/locate/cad

Technical note

Modeling by composition

Gershon Elber [a,*], Myung-Soo Kim [b]

[a] Department of Computer Science, Technion, Israel Institute of Technology, Haifa 32000, Israel
[b] School of Computer Science and Engineering, Seoul National University, Seoul 151-744, South Korea

ARTICLE INFO

Keywords:
Symbolic computation
Freeform deformations
Blending
Rounding
Hausdorff distance

ABSTRACT

Functional composition can be computed efficiently, robustly, and precisely over polynomials and piecewise polynomials represented in the Bézier and B-spline forms (DeRose et al., 1993) [13], (Elber, 1992) [3], (Liu and Mann, 1997) [14]. Nevertheless, the applications of functional composition in geometric modeling have been quite limited. In this work, as a testimony to the value of functional composition, we first recall simple applications to curve–curve and curve–surface composition, and then more extensively explore the surface–surface composition (SSC) in geometric modeling. We demonstrate the great potential of functional composition using several non-trivial examples of the SSC operator, in geometric modeling applications: blending by composition, untrimming by composition, and surface distance bounds by composition.

1. Introduction

Splines are a common representation in virtually almost all computer aided geometric design (CAGD) systems. The Bézier and NURBS representations almost solely govern the geometric modeling industry. Excellent techniques to create and modify these representations have been developed in the CAGD community, which made these representations so common. On the other hand, the Bézier and NURBS representations are often too complex to be handled precisely. Boolean operations and intersections [1] and/or generic operations such as offsets [2] are not closed in the Bézier and NURBS representations and thus should be approximated, entailing all the difficulties such approximations induce.

To alleviate some of these difficulties, Elber [3] introduced *symbolic tools*, which mean computational schemes that allow one to evaluate a symbolic expression once a real numeric input is provided. For example, given two parametric curves $C_1(u)$ and $C_2(v)$, the simultaneous zeros of the following two expressions:

$$\left\langle C_1(u) - C_2(v), \frac{dC_1(u)}{du} \right\rangle = 0,$$
$$\left\langle C_1(u) - C_2(v), \frac{dC_2(v)}{dv} \right\rangle = 0, \tag{1}$$

prescribes one type of distance-extrema event, which is characterized by a bi-normal line (a line normal to both curves at its intersection points with the two curves). Given two Bézier and/or B-spline curves $C_1(u)$ and $C_2(v)$, the numeric representations of $C_1(u)$ and $C_2(v)$ can be plugged into Eq. (1), producing a non-linear system of two equations and two unknowns, whose solution(s) detects all the mutual bi-normals of $C_1(u)$ and $C_2(v)$.

Symbolic manipulation tools have been used in the last couple of decades, offering robust solutions to many computational queries regarding freeform curves and surfaces. With the aid of algebraic operators to add, subtract, and multiply splines [4,3,5], and a solver for systems of non-linear constraints [6,7], robust computation methods were developed, for example, to evaluate offsets and sweeps [8,2], to construct bisectors and Voronoi regions [9,10], and to measure minimal and Hausdorff distances [11,12] between freeform curves and surfaces.

The *composition operator* is one additional symbolic algebraic tool that is worth exploring more extensively. The composition is a well defined operation. Techniques to evaluate the composition of freeforms, directly in the spline (Bézier or B-spline) domains are well-known [13,3,14]. Nevertheless, this operator has not been fully exploited in geometric modeling to its great potential. In particular, surface–surface composition (SSC) has rarely been used. In this work, we show that the composition operator has a lot to offer in geometric modeling. We first discuss the existing examples and applications for curve–curve composition, curve–surface composition, and then focus on surface–surface composition (SSC).

The rest of this work is organized as follows. In Section 2, previous work on the computation of the composition operator is laid out as well as some discussion on the previous uses of this operator. In Section 3, we show how SSC can be used to create a general blending between two surfaces and with arbitrary continuity. In Section 4, we present a paradigm that can convert trimmed surfaces to regular tensor product patches, again using SSC, and

* Corresponding author.
E-mail address: gershon@cs.technion.ac.il (G. Elber).

0010-4485/$ – see front matter © 2013 Elsevier Ltd. All rights reserved.
http://dx.doi.org/10.1016/j.cad.2013.08.032

Section 5 considers the question of bounding the maximum distance between two adjacent patches that are parameterized differently and exploits SSC to much improve on the established distance bound. Then, we conclude in Section 6.

2. Previous work

The composition of two spline functions, in the Bézier and B-spline bases, was first discussed in [13,3]. DeRose et al. [13] reduced the problem of function composition to Blossoming evaluations, and made the observation that surface–surface composition in the B-spline domain can create non-rectangular regions, imposing a major barrier on the computation. Elber [3] reduced the problem to basic symbolic operations such as additions and products of splines, and extended the composition operator to both the polynomial Bézier and piecewise polynomial B-spline domains. In [14], an effort was made into further optimizing the composition computation of two polynomials [13].

Even before the results [13,3], Sederberg [15] proposed trivariate Bézier volumes as a deformation tool. The original proposal of Sederberg [15] was to map control points and so as to approximate the deformation. There are also some previous results developed for the precise freeform deformation using the composition operator such as Feng and Peng [16], where the composition computation was resolved by posing it as a polynomial interpolation problem. Surazhsky and Elber [17] is another example of precise[1] deformation using the composition operator. They employed the curve–surface composition for a precise text deformation of piecewise Bézier outline fonts, where the underlying deformation functions were represented as bivariate B-splines.

In the last couple of decades, other results were also developed for various specific applications using the composition operator. For example, using the surface–surface composition for bilinear patches, Feng and Peng [18] showed how to transform a rectangular (tensor product) patch into two triangular patches and how to convert a triangular patch into three rectangular ones, a problem that was also examined by [13].

Elber [19] used the curve–curve composition to normalize vector fields in general, and to approximate piecewise polynomial arc-length curves in specific. Moreover, Cohen et al. [20] employed the curve–curve composition for the elimination of self-intersections in planar ruled surfaces and in metamorphosis between two curves. Kim and Elber [21] developed a precise G^1 surface blending scheme that exploits the curve–surface composition to precisely locate the rail curves of the blending surface over the given input surfaces.

The coming sections of this work focus on the surface–surface composition (SSC) and present results in a variety of applications.

3. Blending by composition

Surface rounding and/or blending is a well-known problem that has been extensively investigated in many previous results [21,22]. Nevertheless, few results offer blending algorithms that are precise to within machine precision. Typical blending solutions derive the rail curves of the blend (the two curves between which the blending surface is defined) as a solution of some offset or as a surface–surface intersection (SSI) operation, which produces rail curves that are within the tolerance of offset or SSI computation. The error is typically much larger than that of machine precision. One exception is the approach of Kim and Elber [21], where the rail curves are specified in the parametric domains of the two input surfaces, $S_1(u, v)$ and $S_2(r, t)$. In this approach, using the curve–surface composition, the rail curves (and the tangent field) over S_1 and S_2 can be located within machine precision.

The SSC operations can be used to derive precise blending and/or rounding surfaces with a *continuity of arbitrary order*. Consider the two surfaces $S_1(u, v)$ and $S_2(r, t)$ and the two rail curves $C_1(a) = (u(a), v(a))$ and $C_2(b) = (r(b), t(b))$ in the parametric domains of S_1 and S_2.

Assume that $C_1(a)$ and $C_2(b)$ are interior to the respective domains of S_1 and S_2 so that a small offset approximation of $C_1(a)$ and $C_2(b)$ remains interior to S_1 and S_2.[2] Then, the following procedure will generate such a precise G^k continuous blending surface:

Algorithm 1: BUILDING A PRECISE G^k BLENDING SURFACE BETWEEN TWO GENERAL RAIL CURVES $C_1(a)$ IN $S_1(u, v)$ DOMAIN AND $C_2(b)$ IN $S_2(r, t)$ DOMAIN:

input : $S_1(u, v)$, first surface to blend;
$S_2(r, t)$, second surface to blend;
$C_1(a) = (u(a), v(a))$, rail curve in S_1;
$C_2(b) = (r(b), t(b))$, rail curve in S_2;
o_d, offset amount to apply to C_1 and C_2;
output A blending surface B between C_1 and C_2;
:
1.1 $C_1^o(a) \Leftarrow$ Offset of $C_1(a)$ by o_d in the domain of S_1;
1.2 $C_2^o(b) \Leftarrow$ Offset of $C_2(b)$ by $-o_d$ in the domain of S_2;
1.3 $R_1(a, p) \Leftarrow$ ruled surface from $C_1(a)$ to $C_1^o(a)$;
1.4 $R_2(a, p) \Leftarrow$ ruled surface from $C_2^o(b(a))$ to $C_2(b(a))$;
1.5 $R_1^e(a, p) \Leftarrow S_1(R_1(a, p))$;
1.6 $R_2^e(a, p) \Leftarrow S_2(R_2(a, p))$;
1.7 $B(a, p) \Leftarrow \text{Blend}(R_1^e(a, p), R_2^e(a, p))$;

Lines 1.1 and 1.2 of Algorithm 1 compute the offsets by a small radius o_d to the input rail curves that are assumed to be contained in the domains of S_i. The influence of o_d on the outcome will be discussed later. In Lines 1.3 and 1.4, two ruled surfaces are constructed in the parametric spaces of both S_1 and S_2. One should note that in Line 1.4, we can also control the mapping between the two curves' parameterizations using $b(a)$. As a first order approximation, $b(a)$ can be a linear re-parameterization that maps the domain of C_1 to that of C_2 and the curve–curve composition $C_2(b(a))$ is of the same degree as C_2. However, an additional degree of freedom is now being added, by $b(a)$, to possibly control the speed of C_2 and possibly match it to (a scaled constant factor of) the speed of C_1. Alternatively, $b(a)$ can be used to induce a desired speed on both C_1 and C_2 as is done in [19] that approximates an arc-length parameterization, or to match some shape similarities between the two curves, as is done, for example in [20]. Then, the two ruled surfaces are mapped to the Euclidean space in Lines 1.5 and 1.6 of Algorithm 1 by using the SSC operator.

The final step, in Line 1.7, computes the desired blending surface. Denote by B^k a blending surface with a G^k-continuity. For a C^0 blending surface, one can use linear (Bézier basis) functions:

$$B^0(a, p) = (1 - p)R_1^e(a, p) + pR_2^e(a, p). \tag{2}$$

For a G^1 blending surface, one can use the cubic Hermite basis functions:

$$B^1(a, p) = H_{00}(p)R_1^e(a, 0) + H_{10}\left(\left.\frac{\partial R_1^e(a, p)}{\partial p}\right|_{p=0}\right) + H_{01}(p)R_2^e(a, 1) + H_{11}\left(\left.\frac{\partial R_2^e(a, p)}{\partial p}\right|_{p=1}\right). \tag{3}$$

[1] In this work the term *precise* denotes *machine precision*.

[2] Otherwise, one can always C^k-extend the domain of S_i a bit, computing a larger S_i^L surface that identifies with S_i in the original domain.