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 Eng. Seoul National University Seoul 151-744, South Korea

Abstract

Functional composition can be computed efficiently, robustly, and precisely over polynomials and piecewise polynomials represented in the Bézier and B-spline forms [6, 8, 20]. 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.

Keywords: Symbolic computation, freeform deformations, blending, rounding, Hausdorff distance.

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 [24] and/or generic operations such as offsets [14] are not closed in the Bézier and NURBS representations and thus should be approximated, entailing all the difficulties that such approximations induce.

To alleviate some of these difficulties, Elber [8] 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:

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

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 Equation (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 [3, 8, 21], and a solver for systems of non-linear constraints [13, 26], robust computation methods were developed, for example, to evaluate offsets and sweeps [10, 14], to construct bisectors and Voronoi regions [17, 23], and to measure minimal and Hausdorff distances [1, 19] 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 [6, 8, 20]. 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 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 com-

position operator is laid out as well as some discussion on 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 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 [6, 8]. DeRose et al. [6] reduced the problem of function composition to Blossoming evaluations, and makes the observation that surface-surface composition in the B-spline domain can create non-rectangular regions, imposing a major barrier on the computation. Elber [8] 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 [20], an effort was made into further optimizing the composition computation of two polynomials [6].

Even before results [6, 8], Sederberg [25] proposed trivariate Bézier volumes as a deformation tool. The original proposal of Sederberg [25] was to map control points and so as to approximate the deformation. There are also some previous results developed for 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 [27] 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 [15] 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 [6].

Elber [9] 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. [5] employed the curve-curve composition for the elimination of self-intersections in planar ruled surfaces and in metamorphosis between two curves. Kim and Elber [18] 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 [18, 28]. 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 produce 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 [18], 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 $C_1(a)$ 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:

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 reparameterization that maps the domain of C_1 to that of C_2 and the curve-curve composition $C_2(b(a))$ is of the

[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.

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 $B(a,p) \Leftarrow \text{Blend}(R_1^e(a,p), R_2^e(a,p))$;

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 [9] that approximates an arc-length parameterization, or to match some shape similarities between the two curves, as is done, for example in [5]. 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:

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

Note that this result recovers the blending scheme of [18], where the tangent fields (required for the G^1 continuity) were computed along the rail curves by deriving the normal fields of S_i, n_i, only to cross-product n_i with the tangent field of C_i^o. The magnitude of these vectors fields, like n_i, must be controlled in [18] and (approximately) normalized. In contrast to [18], herein the tangent field is recovered directly from R_i^e and hence is expected to be of a lower order (the exact degree of which is dependent on the degrees of the ruled surfaces, R_i).

Figure 1 shows some examples. Figures 1 (a) and (b) show two G^1 ($k = 1$) blends between the left and the right sides of the spout and the body of the Utah teapot. Because $R_1^e(a,p)$ and $R_2^e(a,p)$ are precisely embedded in the input surfaces S_1 and S_2, one can generate a blending surface of *arbitrary continuity*, B^k, by computing all the needed partials of $R_1^e(a,p)$ and $R_2^e(a,p)$ with respect to p and employing degree $2k-1$ Hermite blending functions to achieve the G^k continuity. Figures 1 (c) and (d) show two G^2 ($k = 2$) blends between the left and the right sides of the spout and the body. All these examples are accurate to within machine precision.

Consider the effect of o_d from Algorithm 1. We parametrize the ruled surfaces, $R_i(a,p)$, in the p direction between zero and one (so we can then exploit the Bézier or Hermite basis functions in p with ease). Hence, the offset radius directly controls the *magnitude* of the 1st derivatives. Figure 2 shows a few examples demonstrating the effect of different o_d values.

So far, we did not take into account stretch due to the mapping of the blended surfaces. An offset by amount o_d in the domain of S_i can create an offset in Euclidean space, over S_i, of different and varying distances. In some cases, when a precise type of blend (i.e. a ball-end blend) is needed or the Jacobian of S_i is not well behaved, the stretching metric of S_i must be considered as well and algorithms for offset of varying-amount o_d, such as [11], must be employed.

4. Untrimming by Composition

Given a trimmed Bézier surface, S_t, it is often needed to convert the surface into a set of (untrimmed) tensor product surface patches. However, this conversion, if made precisely, has been considered to be a challenging problem. Using surface-surface composition (SSC), one can approach this problem in two steps:

1. A division of the valid domain of trimmed surface S_t into quadrilateral domains (with possibly freeform boundaries). Dividing a closed domain, possibly with holes, into quadrilateral regions is a well-known problem in various fields such as finite element generations and more recently in quad-remeshing of polygonal surfaces [2, 22]. In this work, one assumes such a tool is given as a block-box that decomposes the given domain into quads.
2. Once the division of parametric domain into quadrilateral domains is given, the precise evaluation of the corresponding 3D tensor product surface patches in the Euclidean space is again a simple instantiation of the SSC operator. One should

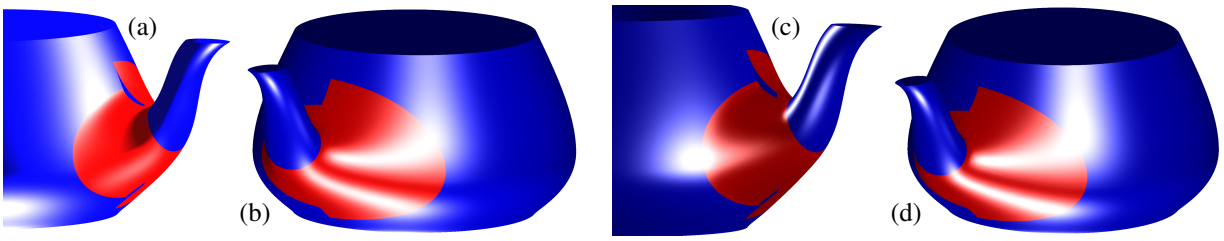

Figure 1: A precise G^1 Hermite blending surface of degrees (3×18) in (a) and (b) (See Equation (3)) between a linear B-spline curve on the bi-cubic spout of the teapot and a cubic B-spline curve on its bi-cubic body. A similar precise G^2 Hermite blending surface of degrees (5×18) is presented in (c) and (d). Note two different blends are actually shown between the two, left and right, halves. Compare with Figure 2.

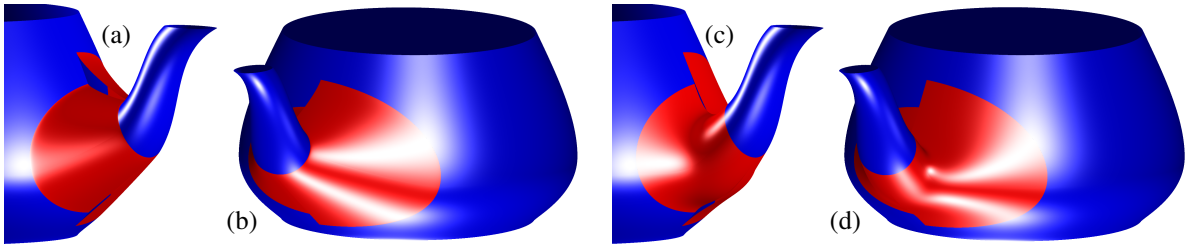

Figure 2: A precise G^1 blending surface of degrees (3×18) that shows the affect of the offset amount o_d from Algorithm 1 and the magnitude of the 1st derivative. (a) and (b) show the use of a small value of o_d whereas (c) and (d) portray a large o_d value. Compare with Figure 1 that uses a mid-range o_d value.

note that the question of parametrizing a general quadrilateral region can still be challenging and simple methods like Boolean-Sum [7] might fail at times. That said, one can verify the regularity of a given parametrization $f(u, v) : \mathbb{R}^2 \rightarrow \mathbb{R}^2$ by symbolically computing $n(u, v) = \frac{\partial f(u,v)}{\partial u} \times \frac{\partial f(u,v)}{\partial v}$ (a scalar field) and making sure $n(u, v)$ never vanish - for instance if all the coefficients of $n(u, v)$ share the same sign.

If the input trimmed surface S_t is a B-spline (or NURBS) surface, first divide S_t into a set of trimmed Bézier surfaces, by dividing S_t at all its interior knots. Then, apply steps 1 and 2 above to each Bézier surface, in this divided set of S_t.

Figures 3 and 4 show two simple examples of converting one trimmed Bézier surface into (untrimmed) tensor product B-spline surface patches via the SSC operator. One should note that even for an input trimmed Bézier surface, the resulting untrimmed patches might be forced to be B-spline surfaces due to existence of B-spline trimming curves.

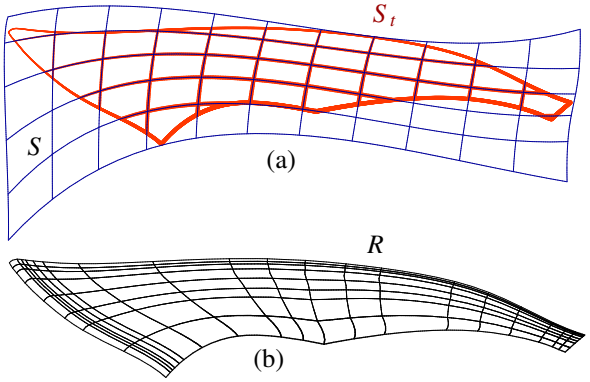

Figure 3: The bicubic trimmed Bézier surface S_t with a quadratic trimming B-spline curve in red in (a) is converted to a regular tensor product B-spline surface R of degrees (12×12) in (b) by forming a 4-sided B-spline parametric mapping to the trimming curve of S_t, and then applying surface-surface compositions to it over S. Note the C^1 discontinuity in the trimming curve is manifested itself in the resulting B-spline surface R.

5. Distance Bounds by Composition

In many applications, it is important to bound the (Hausdorff) distance between two similar models in close proximity. Examples include cases where a given freeform model consisting of (piecewise) polynomial surfaces is approximated with similar surfaces of low degree or surfaces of special type. In the application of manufacturing, a general freeform model can be approximated with piecewise developable or ruled surfaces so as to construct the model using manufacturing techniques such as hot wire or wire EDM [3]. Other examples include side milling that is based on the offsets of piecewise ruled surface approximations to the given

[3] See, for example http://en.wikipedia.org/wiki/Electrical_discharge_machining and http://en.wikipedia.org/wiki/Hot-wire_foam_cutter.

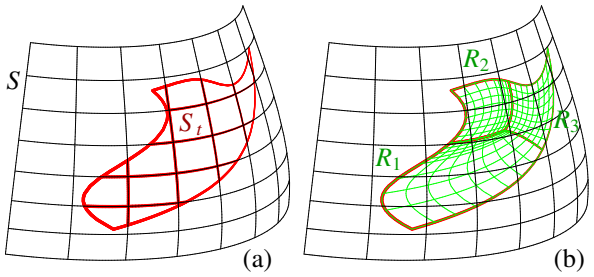

Figure 4: The bicubic trimmed Bézier surface S_t with a cubic trimming B-spline curve in red in (a) is converted to regular tensor product B-spline surfaces R_i in (b) by dividing the trimming region to three 4-sided regions and parametrizing these three 4-sided domains of S_t inside S's parametric domain as bicubic regions. Then, a surface-surface composition is applied to the three regions over S, creating three patches of degrees (18×18), in green.

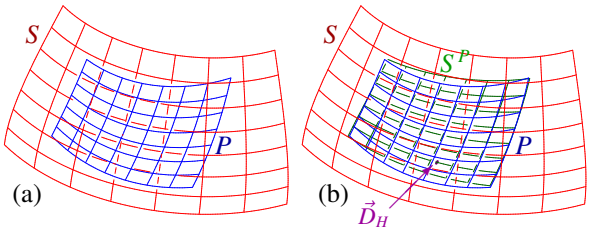

Figure 5: In (a), surface P in blue approximates a portion of the input bicubic surface S in red. In (b), the corresponding (aligned) region of S near surface P, S^P in green, is extracted via surface-surface composition of S and a bilinear, formed out of the four corners of P projected onto S. With S^P of degrees (6×6), the one sided Hausdorff distance from P to S can now be tightly bound as $\vec{D}_H(P, S^P)$. In this example, the established bound (using the difference of the 49 corresponding control points and following Equation (4)) was ~%65 percent larger than the actual Hausdorff distance, that is assumed at point $\vec{D}_H$ in (b), in magenta.

surface. In many cases and in practice, the lower degree and/or the ruled/developable surfaces do not share the same parameterization with the original input surfaces. The Hausdorff distance computation is thus needed so as to guarantee the quality of approximation in these applications.

Consider a 3-space parametric input surface $S(u, v)$ and an approximating patch $P(r, t)$ that spans a portion of S (see also Figure 5 (a)). How can one tightly bound the one-sided Hausdorff distance between P and the relevant portions of S near P? Typical practical solutions include sampling points $\{p_i\}$ on P and projecting them to S, as $\{s_i\}$ (and vice versa) and finding an estimation of the approximation error as $\max \|p_i - s_i\|$, or simply sampling points on both entities and computing distances between the sampled points [4]. Clearly, neither provide a guaranteed upper bound for the Hausdorff distance error.

Given surfaces S and P, one can bring them to a common function space, that is, by elevating them both to the same degrees, and if S or P are piecewise polynomials by also refining them so they both possess the same knot sequences. Then, a guaranteed yet somewhat loose upper bound on the one-sided Hausdorff distance, $\vec{D}_H$, can be easily established as

$$\begin{aligned}
\vec{D}_H(P, S) &= \max_{r,s} \min_{u,v} |S(u, v) - P(r, s)| \\
&\leq \max_{u,v} |S(u, v) - P(u, v)| \\
&= \max_{u,v} \left| \sum_{i,j} Q^S_{ij} B_{ij}(u, v) - Q^P_{ij} B_{ij}(u, v) \right| \\
&\leq \max_{i,j} \left| Q^S_{ij} - Q^P_{ij} \right|, \qquad (4)
\end{aligned}$$

where Q_{ij} are the control points of the respective surfaces.

In order to be able to employ Equation (4) as a tighter upper bound, one can extract the region of S near P, denoted S^P (See Figure 5 (b)), parametrize it similarly to P, and compare the control points between P and S^P. How to extract this S^P region is application dependent. Algorithm 2 presents one simple approach to perform this task. Line 2.4 of Algorithm 2 computes S^P using the surface surface composition, composing the bilinear surface determined by the four projected points of P onto S with the surface S itself. As a result, the surface patch $S^P(r, t)$ (and its parameterization) is aligned with $P(r, t)$ and Equation (4) can now be used to more tightly bound the one-sided Hausdorff distance from P to S. By construction, the parametrization of P will be close to that of S^P. Simple degree raising and/or refinement of the surfaces to bring them to the common space, can provide a fairly tight bound, using Equation (4). One can further improve this bound by using the spread of the control points of one surface to refine the other, mimicking the parametrizations' speed of the other surface as much as possible.

Algorithm 2: EXTRACTING A REGION OF THE INPUT SURFACE S CLOSE TO SURFACE P

input : $S(u, v)$, a general surface;
$P(r, t)$, a surface over a portion of S;
output $S^P(r, t)$, a portion of S close to P;
:
2.1 C_{ij}, $i, j = 1, 2 \Leftarrow$ four corner points of surface P;
2.2 $UV^S_{ij} \Leftarrow$ UV values of C_{ij} projected onto S;
2.3 $B(r, t) \Leftarrow$ bilinear through four points UV^S_{ij}, in S's domain;
2.4 $S^P(r, t) \Leftarrow S(B(r, t))$;

For the case of Figure 5, the 49 corresponding control points of the two surfaces of degrees (6×6) in the common space (Figure 5 (b)) are compared, resulting in a bound on the Hausdorff distance that is ~%65 percent

larger than the actual Hausdorff distance, computed precisely using [12].

6. Conclusions

We presented several applications that employ the SSC operator in geometric design. The accuracy of the presented operations is within machine precision while we have shown abilities to compute composed surfaces with high quality and continuity. These present applications deserve further research. For instance, how can one find a better (best?) parameterization to S^P in Section 5 in order to minimize the error in the bound? We hope that this work will renew interest in the composition operation and specifically other applications to the SSC will be explored as well. Nevertheless, the degrees of the composition results can be quite high and as an additional future work, one should develop techniques to improve the efficiency of evaluating high-degree surfaces thus generated as the result of function composition.

Acknowledgments

The research leading to these results has received partial funding from the People Programme (Marie Curie Actions) of the European Union's Seventh Framework Programme FP7/2007-2013/ under REA grant agreement PIAP-GA-2011-286426, and was supported in part by the Technion Vice President for Research Fund - Glasberg-Klein research fund.

References

[1] Alt, H., and Scharf, L. Computing the hausdorff distance between sets of curves. In *Proceedings of the 20th European Workshop on Computational Geometry (EWCG)* (2004), pp. 233–236.

[2] Campen, M., Bommes, D., and Kobbelt, L. Dual loops meshing: quality quad layouts on manifolds. *ACM Trans. Graph. 31*, 4 (Apr. 2012).

[3] Chen, X., Riesenfeld, R. F., and Cohen, E. Sliding windows algorithm for b-spline multiplication. In *Proceedings of the 2007 ACM symposium on Solid and physical modeling* (New York, NY, USA, 2007), SPM '07, ACM, pp. 265–276.

[4] Cignoni, P., Rocchini, C., and Scopigno, R. Metro: measuring error on simplified surfaces. *Computer Graphics Forum 17*, 2 (1998), 167–174.

[5] Cohen, S., Elber, G., and Bar-Yehuda, R. Matching of freeform curves. *Computer-Aided Design 29*, 5 (1997), 369 – 378.

[6] DeRose, T. D., Goldman, R. N., Hagen, H., and Mann, S. Functional composition algorithms via blossoming. *ACM Trans. Graph. 12*, 2 (Apr. 1993), 113–135.

[7] E. Cohen, R. F. Riesenfeld, G. E. *Geometric Modeling with Splines*. A. K. Peters, New York, 2001.

[8] Elber, G. *Free Form Surface Analysis Using A Hybrid of Symbolic and Numerical Computation*. PhD thesis, University of Utah, 1992.

[9] Elber, G. Symbolic and numeric computation in curve interrogation. *Computer Graphics Forum 14*, 1 (1995), 25–34.

[10] Elber, G. Global error bounds and amelioration of sweep surfaces. *Computer-Aided Design 29*, 6 (1997), 441 – 447.

[11] Elber, G., and Cohen, E. Error bounded variable distance offset operator for free form curves and surfaces. *The International Journal of Computational Geometry & Applications 1*, 1 (1991), 67–78.

[12] Elber, G., and Grandine, T. Hausdorff and minimal distances between parametric freeforms in $\mathbb{R}^2$ and $\mathbb{R}^3$. In *Proceedings of the 5th international conference on Advances in geometric modeling and processing* (2008), GMP'08, Springer-Verlag, pp. 191–204.

[13] Elber, G., and Kim, M.-S. Geometric constraint solver using multivariate rational spline functions. In *Proceedings of the sixth ACM symposium on Solid modeling and applications* (New York, NY, USA, 2001), SMA '01, ACM, pp. 1–10.

[14] Elber, G., Lee, I.-K., and Kim, M.-S. Comparing offset curve approximation methods. *IEEE Comput. Graph. Appl. 17*, 3 (May 1997), 62–71.

[15] Feng, J., and Peng, Q. Functional compositions via shifting operators for bezier patches and their applications. *Chinese Journal of Advanced Software Research* (1999).

[16] Feng, J., and Peng, Q. B-spline free-form deformation of polygonal objects through fast functional composition. In *Geometric Modeling and Processing* (Hong Kong, China, Apr. 2000), pp. 408–415.

[17] Hanniel, I., Muthuganapathy, R., Elber, G., and Kim, M.-S. Precise voronoi cell extraction of free-form rational planar closed curves. In *Proceedings of the 2005 ACM symposium on Solid and physical modeling* (New York, NY, USA, 2005), SPM '05, ACM, pp. 51–59.

[18] Kim, K., and Elber, G. A symbolic approach to freeform surface blends. *THE J. OF VISUALIZATION AND COMPUTER ANIMATION 8*, 2 (1997), 69–80.

[19] Kim, Y.-J., Oh, Y.-T., Yoon, S.-H., Kim, M.-S., and Elber, G. Precise hausdorff distance computation for planar freeform curves using biarcs and depth buffer. *Vis. Comput. 26*, 6-8 (Jun. 2010), 1007–1016.

[20] Liu, W., and Mann, S. An optimal algorithm for expanding the composition of polynomials. *ACM Trans Graph 16* (1997), 155–178.

[21] Morken, K. M. Some identities for products and degree raising of splines. *Constructive Approximation*, 7 (1991), 195–208.

[22] Pellenard, B., Morvan, J.-M., and Alliez, P. Anisotropic rectangular metric for polygonal surface remeshing. In *International Meshing Roundtable* (Oct. 2012).

[23] Peternell, M. Geometric properties of bisector surfaces. *Graphical Models 62*, 3 (2000), 202–236.

[24] Sederberg, T. W., Finnigan, G. T., Li, X., Lin, H., and Ipson, H. Watertight trimmed nurbs. *ACM Trans. Graph. 27*, 3 (Aug. 2008), 79:1–79:8.

[25] Sederberg, T. W., and Parry, S. R. Free-form deformation of solid geometric models. *SIGGRAPH Comput. Graph. 20*, 4 (Aug. 1986), 151–160.

[26] Sherbrooke, E. C., and Patrikalakis, N. M. Computation of the solutions of nonlinear polynomial systems. *Comput. Aided Geom. Des. 10*, 5 (Oct. 1993), 379–405.

[27] Surazhsky, T., and Elber, G. Artistic surface rendering using layout of text. *Computer Graphics Forum 21*, 2 (2002), 99–110.

[28] Vida, J., Martin, R. R., and Varady, T. A survey of blending methods that use parametric surfaces. *Computer-Aided Design 26*, 5 (1994), 341 – 365.