Snap Rounding of Bézier Curves

Arno Eigenwillig Lutz Kettner
Nicola Wolpert

MPI–I–2006–1–005 December 2006

Authors' Addresses

Arno Eigenwillig
Max-Planck-Institut für Informatik
Stuhlsatzenhausweg 85
66123 Saarbrücken
Germany

Lutz Kettner*
Max-Planck-Institut für Informatik
Stuhlsatzenhausweg 85
66123 Saarbrücken
Germany

* Now at mental images GmbH, Berlin, Germany.

Nicola Wolpert
Hochschule für Technik Stuttgart
70174 Stuttgart
Germany

Acknowledgements

The authors thank Kurt Mehlhorn for useful comments.

This work was partially supported by the IST Programme of the European Union as a Shared-cost RTD (FET Open) Project under Contract No. IST-006413 (ACS – Algorithms for Complex Shapes)

Abstract

We present an extension of snap rounding from straight-line segments (see Guibas and Marimont, 1998) to Bézier curves of arbitrary degree, and thus the first method for geometric rounding of curvilinear arrangements. Our algorithm takes a set of intersecting Bézier curves and directly computes a geometric rounding of their true arrangement, without the need of representing the true arrangement exactly. The algorithm's output is a deformation of the true arrangement that has all Bézier control points at integer points and comes with the same geometric guarantees as in straight-line snap rounding: during rounding, objects do not move further than the radius of a pixel, and features of the arrangement may collapse but do not invert.

Contents

1 Introduction and related work **2**

2 Basics on Bézier curves **5**

3 The snap rounding algorithm for Bézier curves **8**
3.1 The preprocessing phase 9
3.2 The graph building phase 9
3.3 The conflict removal phase 10
3.4 The merging phase 11
3.5 The rounding phase 14

4 Geometric analysis **15**

5 Concluding remarks and future work **21**

A Proof details **24**

B Pseudocode **27**
B.1 The preprocessing phase 27
B.2 The graph building phase 28
B.3 The conflict removal phase 29
B.4 The merging phase 30

1 Introduction and related work

Putting geometric algorithms into practice is a challenging task for several well-known reasons, most prominently the delicate dependency between numerical and combinatorial computations. Often, algorithms are designed for exact arithmetic. Naive use of finite-precision arithmetic in its place is a recipe for disaster.

One solution is to design specific algorithms that can cope with numerical imprecision in a controlled manner. We mention two recent examples for arrangements of curves. Milenkovic and Sacks [15] compute an approximate arrangement and prove, under certain empirically justified assumptions on the underlying numerical solver, that there exists a perturbation of the input which realizes the computed arrangement. Halperin and Leiserowitz [9] show how to actually carry out a *controlled perturbation* of the input such that fixed-precision arithmetic suffices; see also [14]. The guarantee offered by these methods bounds *backward error*: the result is correct for a slight (implicit or explicit) perturbation of the original input.

A more general approach is the *Exact Geometric Computation* (EGC) paradigm coined by Yap (see [20]) that demands exact determination of geometric relations, using as much numerical precision as necessary. General-purpose geometry libraries such as CGAL [3] [12] and LEDA [13] [12] have successfully implemented EGC. Yap [21] gave an EGC algorithm for intersecting Bézier curves, solving the difficult case of tangential intersections with subdivision up to a separation bound, which may be costly. EGC delivers error-free results, but these results live on an island: Exact coordinates of newly computed objects cannot be used "as is" in traditional file formats or APIs with fixed-precision number types; it is necessary to round them. Also, EGC implementations with exact construction of objects suffer from exponential coordinate growth in cascaded constructions (see [16]) if no intermediate rounding takes place.

Our contribution follows a third approach, which we introduce by way of its classical example: Consider a planar arrangement of straight-line segments represented by a graph. Vertices are labelled by their cartesian coordinates. Edges represent the line segment between their endpoints; they may not intersect in their

interiors. We want to round all vertex coordinates to, say, integers. Rounding means moving each vertex, and thus implicitly also its incident line segments. Doing that may introduce spurious crossings of segments, so a mere numerical rounding of vertex coordinates would make the geometry implied by the vertex coordinates inconsistent with the graph data. Also, it would invert the true topology. What one desires instead is a *geometric rounding* that modifies the graph data as well, such that (i) the data structure remains consistent, and (ii) the original topology is preserved to some extent. A method to perform such geometric rounding can be used to reduce numeric precision in an existing arrangement; e.g., to export an exact arrangement from the EGC island. Moreover, such a method gives rise to algorithms that compute a rounded arrangement directly and thus can avoid the costly exact computation of intersection points. In either case, one obtains a result with bounded *forward error*: the output is close to the true result for the given input.

The problem of rounding planar arrangements of straight-line segments, as well as its three-dimensional extensions, have been in the focus of the research literature on geometric rounding. Formulations that prescribe full preservation of topology are often NP-hard, as shown by the fundamental result on the hardness of preserving the nesting relationship of planar polygons within a given tolerance [17]. Efficient methods are known if the geometric rounding is allowed to collapse small features and to create new contacts in narrow settings. For planar arrengements, we mention the early work by Greene and Yao [7], followed by the popular *snap rounding*, due to Hobby [11] and independently Greene [6], and *shortest-path rounding* by Milenkovic [16], which can result in less additional vertices than snap rounding. Efficient implementations of snap rounding have been treated by Guibas and Marimont [8], Goodrich at al. [5] and de Berg et al. [1]. Variations of snap rounding are iterated snap rounding by Halperin and Packer [10], and the extension to boolean operations on polygons by Devillers and Guigue [2].

Our result We extend snap rounding from straight-line segments to Bézier curves of arbitrary degree, and thus attain the first method for geometric rounding of curvilinear arrangements. We have chosen snap rounding, because it allows proof techniques that generalize well from the straight-line case to our setting. We study Bézier curves, because they are ubiquitous in applications and can represent all polynomially parameterized curves, e.g., pieces of splines. Also, it is clear how to round one Bézier curve, namely by rounding its defining control points, and how this affects the curve. (In contrast, it would not be so clear how to round a segment of an algebraic curve.) The convex hull of the control points encloses the curve, and this straight-line enclosure gives us a point of attack for the generalization from line segments.

In Section 2, we review basics on Bézier curves necessary to describe the actual algorithm in Section 3. The algorithm receives a set of input curves and subdivides them until it has approximated intersection points sufficiently to determine their rounding, and until the enclosing polygons are tight enough so that rounding succeeds. Here, subdivision serves a double purpose: locating intersection points (cf. [21] [18, §3.7]) and breaking curves, like ursegments are broken at hot pixels in straight-line snap rounding. Section 4 presents a geometric analysis of the output. We extend the arguments of [8] for straight-line snap rounding to the polygonal enclosures of the rounded Bézier curves, and thus to the rounded curves themselves. We arrive at the same geometric guarantees as known from straight-line snap rounding: During rounding, *objects do not move further than the radius of a pixel* (Theorem 8). Features of the arrangement may collapse but do not invert; in particular, *the cyclic order of non-collapsed edges around a vertex is preserved* (Theorem 18).

2 Basics on Bézier curves

A *Bézier curve* [18] [4] of formal degree $n > 0$ is a parameterized curve $\mathbf{b}\colon [0,1] \to \mathbb{R}^2$ expressed as $\mathbf{b}(t) = \sum_{i=0}^{n} \mathbf{b}_i B_i^n(t)$ using arbitrary *control points* $\mathbf{b}_0, \ldots, \mathbf{b}_n \in \mathbb{R}^2$ and the *Bernstein polynomials* $B_i^n(t) = \binom{n}{i} t^i (1-t)^{n-i}$ for $i = 0, \ldots, n$. Notice that $\mathbf{b}(0) = \mathbf{b}_0$ and $\mathbf{b}(1) = \mathbf{b}_n$. The polyline $\overline{\mathbf{b}_0 \mathbf{b}_1 \cdots \mathbf{b}_n} := \overline{\mathbf{b}_0 \mathbf{b}_1} \cup \overline{\mathbf{b}_1 \mathbf{b}_2} \cup \cdots \cup \overline{\mathbf{b}_{n-1} \mathbf{b}_n}$ is called the *control polygon*[1] of $\mathbf{b}(t)$. As the Bernstein polynomials on $[0,1]$ form a non-negative partition of unity, the trace $\mathbf{b}([0,1])$ of $\mathbf{b}(t)$ is a subset of the convex hull $\operatorname{conv}\{\mathbf{b}_i\}_i = \operatorname{conv}\{\mathbf{b}_0, \ldots, \mathbf{b}_n\}$ of the control points. We call $\operatorname{conv}\{\mathbf{b}_i\}_i$ the *enclosing polygon* of $\mathbf{b}$. Recall that a parameterized curve is said to be *regular* at t if the *tangent vector* $\mathbf{b}'(t)$ is non-zero.

Given $\mathbf{b}(t) = \sum_{i=0}^{n} \mathbf{b}_i^0 B_i^n(t)$ and $\alpha \in (0,1)$, one can *subdivide* $\mathbf{b}(t)$ into $\mathbf{b}^-(t) := \sum_{i=0}^{n} \mathbf{b}_0^i B_i^n(t) := \mathbf{b}(\alpha t)$ and $\mathbf{b}^+(t) := \sum_{i=0}^{n} \mathbf{b}_i^{n-i} B_i^n(t) := \mathbf{b}((1-\alpha)t + \alpha)$ with the *de Casteljau algorithm* by setting $\mathbf{b}_i^j := (1-\alpha)\mathbf{b}_i^{j-1} + \alpha \mathbf{b}_{i+1}^{j-1}$ for all $j = 1, \ldots, n$ and $i = 0, \ldots, n-j$. Unless indicated otherwise, we subdivide at $\alpha = \frac{1}{2}$. The new control polygons $\overline{\mathbf{b}_0^0 \mathbf{b}_0^1 \cdots \mathbf{b}_0^n}$ and $\overline{\mathbf{b}_0^n \mathbf{b}_1^{n-1} \cdots \mathbf{b}_n^0}$ are contained in $\operatorname{conv}\{\mathbf{b}_i\}_i$. Their concatenation $\overline{\mathbf{b}_0^0 \cdots \mathbf{b}_0^n \cdots \mathbf{b}_n^0}$ is a better polyline approximation of $\mathbf{b}(t)$ than the original control polygon. Repeating subdivision recursively on the subcurves produces a quadratically convergent sequence of polyline and polygon approximations to the Bézier curve formed by the concatenated control polygons or enclosing polygons, respectively. We look at a Bézier curve in exactly this way: it is a curve whose position is known through a *polygonal enclosure* that can be refined further if needed.

The derivative of $\mathbf{b}(t) = \sum_{i=0}^{n} \mathbf{b}_i B_i^n(t)$ is again a Bézier curve: As $\frac{\mathrm{d}}{\mathrm{d}t} B_i^n(t) = n(B_{i-1}^{n-1}(t) - B_i^{n-1}(t))$, one has $\mathbf{b}'(t) = n \sum_{i=0}^{n-1} \Delta\mathbf{b}_i B_i^{n-1}(t)$ with $\Delta\mathbf{b}_i := \mathbf{b}_{i+1} - \mathbf{b}_i$. Let $\mathbf{m}$ be a unit-length vector. We call a sequence or function *increasing in direction* $\mathbf{m}$, or $\mathbf{m}$-*increasing*, if its image $\langle \mathbf{m}, \cdot \rangle$ under projection onto $\mathbf{m}$ is increasing (in the strict sense, equality is not permitted). We also say *monotone* instead of $\mathbf{m}$-increasing if any increasing direction $\mathbf{m}$ exists. We say that $\mathbf{b}(t)$ is *bcp-increasing*

[1] This is the established terminology, even though the control polygon is not a polygon but a polyline.

in direction $\mathbf{m}$, or *bcp-monotone*, if its Bézier control points $\mathbf{b}_0, \ldots, \mathbf{b}_n$ are increasing in direction $\mathbf{m}$.

Lemma 1. *A Bézier curve* $\mathbf{b}(t) = \sum_{i=0}^{n} \mathbf{b}_i B_i^n(t)$ *is bcp-monotone if and only if* $0 \notin \operatorname{conv}\{\Delta\mathbf{b}_0, \ldots, \Delta\mathbf{b}_{n-1}\}$.

Proof. If $\langle \mathbf{m}, \mathbf{b}_{i+1} \rangle > \langle \mathbf{m}, \mathbf{b}_i \rangle$, then $\langle \mathbf{m}, \Delta\mathbf{b}_i \rangle > 0$ for all $i < n$, so any convex combination of the $\Delta\mathbf{b}_i$ lies in the half-plane $\langle \mathbf{m}, \cdot \rangle > 0$; this implies $0 \notin \operatorname{conv}\{\Delta\mathbf{b}_i\}_i$. Conversely, $0 \notin \operatorname{conv}\{\Delta\mathbf{b}_i\}_i$ implies the existence of a half-plane $\langle \mathbf{m}, \cdot \rangle > 0$ comprising $\operatorname{conv}\{\Delta\mathbf{b}_i\}_i$, giving us a bcp-increasing direction $\mathbf{m}$. □

Rounding may create repeated control points $\mathbf{b}_i = \mathbf{b}_{i+1}$. Let $\operatorname{uniq}((\mathbf{b}_0, \ldots, \mathbf{b}_n))$ denote the maximal subsequence of $(\mathbf{b}_0, \ldots, \mathbf{b}_n)$ in which no two successive points are equal. We call a non-constant Bézier curve $\mathbf{b}(t)$ *weakly bcp-monotone* or *weakly bcp-increasing in direction* $\mathbf{m}$, if $\operatorname{uniq}((\mathbf{b}_0, \ldots, \mathbf{b}_n))$ is $\mathbf{m}$-increasing.

Lemma 2. *A Bézier curve* $\mathbf{b}(t) = \sum_{i=0}^{n} \mathbf{b}_i B_i^n(t)$ *is weakly bcp-monotone if and only if* $0 \notin \operatorname{conv}(\{\Delta\mathbf{b}_i\}_i \setminus \{0\})$.

Lemma 3. *Let* $\mathbf{b}(t) = \sum_{i=0}^{n} \mathbf{b}_i B_i^n(t)$ *be a Bézier curve that is weakly bcp-increasing in direction* $\mathbf{m}$.

(i) The function $t \mapsto \mathbf{b}(t)$ *is* $\mathbf{m}$*-increasing and thus injective for* $t \in [0,1]$. *It is regular for* $t \in (0,1)$*; if* $\mathbf{b}(t)$ *is bcp-increasing, it is regular for* $t \in [0,1]$.

(ii) The endpoints $\mathbf{b}(0)$ *and* $\mathbf{b}(1)$ *are extreme points of* $\operatorname{conv}\{\mathbf{b}_0, \ldots, \mathbf{b}_n\}$.

(iii) Subdivision of $\mathbf{b}(t)$ *at* $\alpha \in (0,1)$ *yields subcurves which are weakly bcp-increasing in direction* $\mathbf{m}$ *and whose enclosing polygons intersect only at* $\mathbf{b}(\alpha)$. *If* $\mathbf{b}(t)$ *is bcp-increasing, so are the subcurves.*

Proof. Ad (i). It suffices to show $\langle \mathbf{m}, \mathbf{b}'(t) \rangle > 0$ for $t \in (0,1)$; injectivity at 0 and 1 follows from continuity. We have $\mathbf{b}'(t) = n \sum_{i=0}^{n-1} \Delta\mathbf{b}_i B_i^{n-1}(t)$, and $B_i^{n-1}(t) > 0$ for $t \in (0,1)$. If $\Delta\mathbf{b}_i \neq 0$, then $\langle \mathbf{m}, \Delta\mathbf{b}_i \rangle > 0$. Hence all terms of $\langle \mathbf{m}, \mathbf{b}'(t) \rangle$ are non-negative, and there is at least one positive term, as $\mathbf{b}(t)$ is non-constant. If $\mathbf{b}(t)$ is bcp-increasing, we also have $\mathbf{b}'(0) = \Delta\mathbf{b}_0 \neq 0$ and $\mathbf{b}'(1) = \Delta\mathbf{b}_{n-1} \neq 0$.

Ad (ii). $\mathbf{b}_0$ and $\mathbf{b}_n$ are the unique minimizer and maximizer, resp., of $\langle \mathbf{m}, \cdot \rangle$ in the set $\{\mathbf{b}_0, \ldots, \mathbf{b}_n\}$.

Ad (iii). The two subcurves are $\mathbf{b}^-(t) = \sum_{i=0}^{n} \mathbf{b}_0^i B_i^n(t)$ and $\mathbf{b}^+(t) = \sum_{i=0}^{n} \mathbf{b}_i^{n-i} B_i^n(t)$. The concatenated left, lower, and right sides of a truncated de Casteljau triangle consisting only of rows $0, \ldots, j$ form a sequence $(\mathbf{b}_0^0, \ldots, \mathbf{b}_0^j, \ldots, \mathbf{b}_{n-j}^j, \ldots, \mathbf{b}_n^0)$. For $j = 0$, this is the control polygon of $\mathbf{b}(t)$, hence two successive points are $\mathbf{m}$-increasing (or equal). This property is preserved by the de Casteljau algorithm as j increases and thus holds at $j = n$ for $(\mathbf{b}_0^0, \ldots, \mathbf{b}_0^n, \ldots, \mathbf{b}_n^0)$. Hence $\mathbf{b}^-(t)$ and $\mathbf{b}^+(t)$ are (weakly) bcp-increasing in direction $\mathbf{m}$. The common control point $\mathbf{b}_0^n$ is the unique maximizer of $\langle \mathbf{m}, \cdot \rangle$ in the set $\{\mathbf{b}_0^0, \ldots, \mathbf{b}_0^n\}$ and its unique minimizer in the set $\{\mathbf{b}_0^n, \ldots, \mathbf{b}_n^0\}$. This implies $\operatorname{conv}\{\mathbf{b}_0^0, \ldots, \mathbf{b}_0^n\} \cap \operatorname{conv}\{\mathbf{b}_0^n, \ldots, \mathbf{b}_n^0\} = \{\mathbf{b}_0^n\}$. □

We call a weakly bcp-monotone Bézier curve $\mathbf{b}(t) = \sum_{i=0}^{n} \mathbf{b}_i B_i^n(t)$ *straight* if $\mathbf{b}_0, \ldots, \mathbf{b}_n$ are collinear.

Lemma 4. *Let* $\mathbf{b}(t) = \sum_{i=0}^{n} \mathbf{b}_i B_i^n(t)$ *be a weakly bcp-monotone Bézier curve.*

(i) If $\mathbf{b}(t)$ *is straight, then* $\mathbf{b}([0,1]) = \mathrm{conv}\{\mathbf{b}_i\}_i = \overline{\mathbf{b}(0)\,\mathbf{b}(1)}$.

(ii) If $\mathbf{b}(t)$ *is not straight,* $\mathbf{b}((0,1))$ *is contained in the interior of* $\mathrm{conv}\{\mathbf{b}_i\}_i$.

By Lemma 3(ii), the enclosing polygon f of $\mathbf{b}$ has $\mathbf{b}(0)$ and $\mathbf{b}(1)$ among its vertices. We will use f in place of $\mathbf{b}$ as an edge in a planar graph. This prompts the following definition: A *fat edge* is a convex polygon f, two of whose vertices are distinguished as *endpoints*, in our case: $\mathbf{b}(0)$ and $\mathbf{b}(1)$. A *fat planar graph* is an undirected graph (V,F) with a finite set $V \subseteq \mathbb{R}^2$ of *fat vertices* and a finite edge set F of fat edges with endpoints in V such that any intersection point of two distinct fat edges is a common endpoint.

3 The snap rounding algorithm for Bézier curves

Consider the half-open *unit pixel* $U := [-\frac{1}{2}, +\frac{1}{2}) \times [-\frac{1}{2}, +\frac{1}{2})$. We partition $\mathbb{R}^2$ into *pixels* $\mathbf{a}+U$ around *centers* $\mathbf{a} \in \mathbb{Z}^2$. Each point $(x,y) \in \mathbb{R}^2$ is contained in a unique pixel with center $\rho((x,y)) = (\lfloor x+\frac{1}{2} \rfloor, \lfloor y+\frac{1}{2} \rfloor)$. A straight-line segment $\overline{\mathbf{pq}}$ *passes through* a pixel V if $V \cap \overline{\mathbf{pq}} \neq \emptyset$ and $V \cap \{\mathbf{p}, \mathbf{q}\} = \emptyset$. As in straight-line snap rounding, we say that $\overline{\mathbf{pq}}$ and a point $\mathbf{r}$ *conflict* if $\overline{\mathbf{pq}}$ passes through the pixel around $\mathbf{r}$.

Let S be an input set of non-overlapping Bézier curves. We assume pixel boundaries to be in general position to S. In particular, we assume that intersections of curves, irregular points and self-intersections lie in the interiors of pixels, and that no curve intersects a pixel boundary tangentially or in a corner.

An implementation can overcome this restriction by adding a randomized fractional offset vector $\mathbf{v}$ to the integer grid. As all significant points have to lie on either side of the pixel boundary, but never on it, an implementation can choose the coordinates of $\mathbf{v}$ randomly bit after bit and never has to commit to an exact value. In expectancy, the required number of bits is logarithmic in the number of comparisons made. For ease of exposition, we do not consider this in the sequel and simply make the assumptions stated above.

Our algorithm receives as input the set S of Bézier curves, which we call *urcurves*. In a **preprocessing phase**, we subdivide urcurves into bcp-monotone pieces. In two further subdivision phases, we achieve that their enclosing polygons form a fat planar graph T (**graph building phase**), and that there are no conflicts between control points and straight-line segments linking them (**conflict removal phase**). In principle, we are then be ready for rounding, but this would not yet produce good results, because our algorithm subdivides a lot; and often this merely serves to enclose a curve more tightly but is not required for rounding. Hence there is a **merging phase** for subcurves before they are rounded and output in the **rounding phase**. Figure 3.1, computed by our experimental implementation, shows an example of avoidable fragmentation.

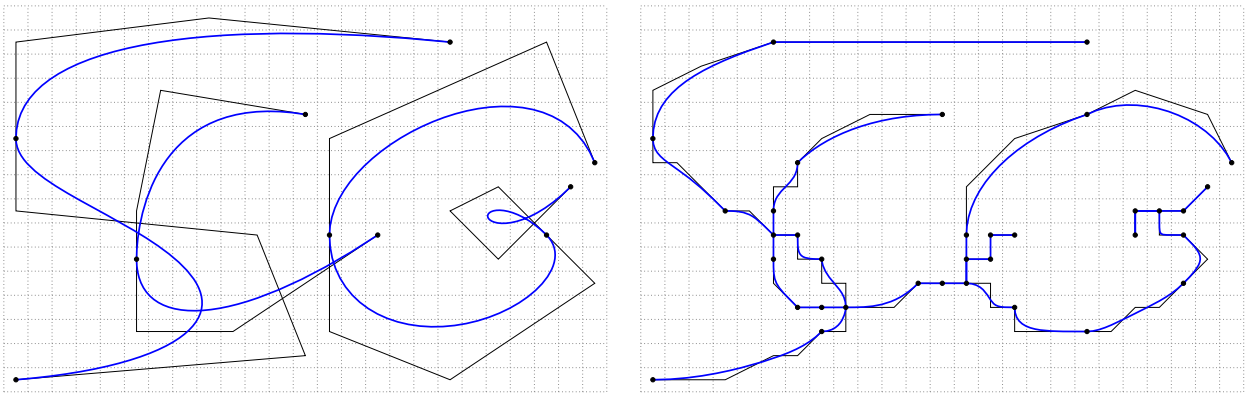

Figure 3.1: The left side shows seven intersecting Bézier curves, which form the letters SCG. Computing their snap-rounded arrangement has resulted in 35 different fragments. For demonstration purposes, the right side shows the rounded arrangement that would result if merging was not applied. The top parts of letters S and G are needlessly fragmented.

3.1 The preprocessing phase

If an urcurve does not have an irregular point, repeated subdivision will partition it into bcp-monotone pieces. (Proof: If $0 \notin \mathbf{b}'([0,1])$, repeated subdivision of $\mathbf{b}'(t)$ will refine its polygonal enclosure away from 0. Eventually, Lemma 1 applies to all subcurves.) But what if there is an irregular point? By our genericity assumption, this point lies in the interior of a pixel. Under repeated subdivision, it will end up on a subcurve that is *trivial*, meaning all its control points are in the same pixel. We can round a trivial curve to a single vertex, namely the center of the pixel containing it. Hence we can reduce it, already before rounding, to one of its endpoints (or both), which will round to a vertex at the pixel center and represent the irregular point.

So the preprocessing phase takes a set S, initially the set of urcurves, and as long as there is a curve $\mathbf{b} \in S$, $\mathbf{b}$ is extracted from S and examined. If $\mathbf{b}$ is bcp-monotone, $\mathbf{b}$ is added to the set Q. If $\mathbf{b}$ is not bcp-monotone but trivial, the endpoint $\mathbf{b}(0)$ is added to the set Q. Otherwise, $\mathbf{b}$ is subdivided and both subcurves are put back into S. When S has become empty, the preprocessing phase terminates with output Q.

3.2 The graph building phase

In this phase, we insert curves from Q and their endpoints into a fat planar graph T. For each fat edge of T, we do not just store the curve $\mathbf{b}$ it stands for, but also the

urcurve $ur[\mathbf{b}]$ of which it is a subcurve, and its parameter interval $[t^0, t^1]$ on the urcurve. For each fat vertex v of T, we store its *preimages*; that is, all pairs $(ur[\mathbf{b}], t_v)$ of an urcurve and a parameter $t_v \in [0,1]$ such that $ur[\mathbf{b}](t_v)$ is the point at which vertex v resides.

We insert any points in Q as vertices into T. While there is a curve $\mathbf{b} \in Q$, we extract it from Q and check whether there is an obstacle to the insertion of its enclosing polygon as a fat edge into T. An *obstacle* is a fat edge or fat vertex in T that has an intersection point with the fat edge to be inserted other than a common endpoint. If there is no obstacle, we simply insert the new fat edge and reiterate. If there is an obstacle, we need to do subdivision. In the general case, we remove one of the obstacles from T, subdivide both the removed obstacle and $\mathbf{b}$, and put their subcurves back into Q. However, there are special cases: If an endpoint of one fat edge (that is, $\mathbf{b}(0)$, $\mathbf{b}(1)$, or a fat vertex of T) is a non-endpoint of another fat edge, only the other fat edge's curve is subdivided, because an endpoint is a point on the curve and will not go away by subdivision. If a curve is trivial, we do not subdivide it further, we discard it (but retain the endpoints).

If two curves are to be subdivided, and the intersection of their enclosing polygons lies inside one pixel V, then, as an optimization, one can split both curves at V, discard the small intersecting pieces inside V, and attain non-intersecting subcurves ending at V. This reproduces the method from CAGD practice [18, §3.7] for intersection of long flat curves in our framework, with "inside one pixel" in place of "error $< \varepsilon$". It improves on repeated subdivision at $\alpha = \frac{1}{2}$ by replacing binary search with direct computation.

3.3 The conflict removal phase

In this phase, we *subdivide edges* of T. That means, we remove an edge, subdivide the curve it stands for, and then re-insert the enclosing polygons of the two resulting subcurves into T. This re-insertion never faces an obstacle in T: The two new fat edges are subsets of the removed fat edge, so no other fat edge is an obstacle to their insertion, and by Lemma 3(iii), they are not obstacles to each other.

We subdivide edges for two reasons. One reason is to ensure that each curve remains weakly bcp-monotone when rounded. The primary reason, however, is to remove conflicts between points and segments in the sense of straight-line snap rounding: A control point $\mathbf{r}$ of some curve must not conflict with a straight-line segment $\overline{\mathbf{pq}}$ between two control points of another curve; otherwise, $\rho(\mathbf{r})$ might end up on the wrong side of $\overline{\rho(\mathbf{p})\rho(\mathbf{q})}$. For this, we need to consider the whole *control clique*, or *CC*, of a Bézier curve, that is the graph of all control points and all straight-line segments between them.

Conflict removal proceeds as follows:

If there exists a fat edge f and a fat vertex v that is not an endpoint of f such that v conflicts with a CC edge of the curve represented by f, then we subdivide f and reiterate.

If there exist two distinct fat edges such that a CC vertex of one conflicts with a CC edge of the other, we subdivide both and reiterate.

If there exists a non-trivial fat edge f representing a Bézier curve $\sum_i \mathbf{b}_i B_i^n(t)$ such that its rounding $\sum_i \rho(\mathbf{b}_i) B_i^n(t)$ is not weakly bcp-monotone, then we subdivide f and reiterate.

If none of the three conditions applies, the conflict removal phase terminates. This happens in the worst case when every urcurve has been subdivided into pieces that are either trivial or have a weakly monotone control polygon $(\mathbf{p},\dots,\mathbf{p},\mathbf{q},\dots,\mathbf{q})$ that connects two adjacent pixels.

All we have done so far is to subdivide urcurves and to and discard some trivial subcurves (except at least one endpoint of each). We call the remaining subcurves the *unrounded fragments*.

Proposition 5. *Let* $\mathbf{b}$ *be a non-trivial unrounded fragment. Let* $V = \rho(\mathbf{b}_{i_0}) + U$ *be a pixel that contains a control point of* $\mathbf{b}$ *and a control point* $\mathbf{q}$ *of another unrounded fragment. Then* $\operatorname{conv}\{\mathbf{b}_i\}_i \cap \partial V$ *is connected.*

The symbol ∂ denotes the boundary of a point set. The proposition follows directly from the next lemma.

Lemma 6. *With notation as above,* $\operatorname{conv}\{\mathbf{b}_i\}_i \setminus V$ *is connected.*

Proof. Elementary arguments (see appendix) show: every connected component C_1 contains a control point $\mathbf{b}_{j_1}$. A second connected component $C_2 \ni \mathbf{b}_{j_2}$ gives rise to a segment $\overline{\mathbf{b}_{j_1}\mathbf{b}_{j_2}}$, which is a subset of $\operatorname{conv}\{\mathbf{b}_i\}_i$ but, by disconnectedness, not of $\operatorname{conv}\{\mathbf{b}_i\}_i \setminus V$. Hence $\overline{\mathbf{b}_{j_1}\mathbf{b}_{j_2}}$ conflicts with $\mathbf{q}$ at V, a contradiction. □

3.4 The merging phase

We will now reduce fragmentation by merging certain chains of unrounded fragments. A *chain* is a sequence $e = (\mathbf{b}^1,\dots,\mathbf{b}^k)$ of fragments that have a common urcurve $\mathbf{b} = ur[\mathbf{b}^j]$ and whose parameter intervals $[t^0[\mathbf{b}^j], t^1[\mathbf{b}^j]]$ on $\mathbf{b}$ are adjacent, that is, $t^1[\mathbf{b}^j] = t^0[\mathbf{b}^{j+1}]$. We call a chain *trivial* if all fragments on it are trivial. We can regard the chain e as one subcurve $\mathbf{c}^e(t) = \sum_{i=0}^n \mathbf{c}_i^e B_i^n(t)$ of $\mathbf{b}$ that corresponds to the parameter interval $I^e := [t^0[\mathbf{b}^1], t^1[\mathbf{b}^k]]$ of $\mathbf{b}$. If the chain e is non-trivial, so is the curve $\mathbf{c}^e$. The control points of each unrounded fragment $\mathbf{b}^j$ on $\mathbf{c}^e$ are convex combinations $\mathbf{b}_i^j = \sum_{\ell=0}^n \mu_\ell \mathbf{c}_\ell^e$, with coefficients μ_ℓ that depend only on the parameter subinterval of $\mathbf{b}^j$ on $\mathbf{c}^e$, not on the location of control points. So when we round $\mathbf{c}^e(t) = \sum_{i=0}^n \mathbf{c}_i^e B_i^n(t)$ to $\rho\mathbf{c}^e(t) = \sum_{i=0}^n \rho(\mathbf{c}_i^e) B_i^n(t)$, this

implicitly moves $\mathbf{b}_i^j = \sum_{\ell=0}^n \mu_\ell \mathbf{c}_\ell^e$ to $\tilde{\mathbf{b}}_i^j = \sum_{\ell=0}^n \mu_\ell \rho(\mathbf{c}_\ell^e)$. Except for $\mathbf{b}_0^1 = \mathbf{c}^e(0)$ and $\mathbf{b}_n^k = \mathbf{c}^e(1)$, all control points $\mathbf{b}_i^j$ are rounded to locations $\tilde{\mathbf{b}}_i^j$, which, in general, are not pixel centers. This requires new ideas to rule out inversion of topology during rounding.

The first idea is this: The displacements $\tilde{\mathbf{b}}_i^j - \mathbf{b}_i^j$ are elements of the inverted unit pixel $-U$, because $\rho(\mathbf{c}_i^e) - \mathbf{c}_i^e$ is an element of the convex set $-U$ for all i. We define the *pixel neighbourhood* $N(\mathbf{p})$ of a point $\mathbf{p}$ to be the pixel around $\mathbf{p}$ and its eight neighbours; that is, $N(\mathbf{p}) := (\rho(\mathbf{p}) + \{-1,0,+1\}^2) + U$. Objects outside $N(\mathbf{b}_i^j)$ cannot collide with $\mathbf{b}_i^j$ during rounding, as we will see in the proof of Lemma 11.

The second idea addresses objects inside $N(\mathbf{b}_i^j)$, provided that they have the same urcurve $\mathbf{b}$. This is to be expected for nearby fragments on the same chain. We will build a subcurve of $\mathbf{b}$ comprising everything that is in the pixel neighbourhood of a control point $\mathbf{b}_i^j$ and make sure that this curve is bcp-monotone after rounding; then Lemma 3(iii) yields separation of the rounded fragments. To guarantee bcp-monotonicity after rounding, we modify the criterion for bcp-monotonicity from Lemma 1 by introducing some slack: We say that a Bézier curve $\mathbf{b}(t) = \sum_{i=0}^n \mathbf{b}_i B_i^n(t)$ is λ*-robustly bcp-monotone* if

$$0 \notin \operatorname{int}\operatorname{conv}\left(\{\Delta\mathbf{b}_0, \ldots, \Delta\mathbf{b}_{n-1}\} + \lambda\{-1,+1\}^2\right), \quad \lambda > 0. \tag{3.1}$$

If $\mathbf{b}$ is bcp-monotone, there exists a maximal value of λ that satisfies (3.1), and we call it the *robustness number* of $\mathbf{b}$. It indicates how much larger a curve $\mathbf{c}$ has to be compared to its subcurve $\mathbf{b}$ so that the bcp-monotonicity of $\mathbf{b}$ is not destroyed when the control points of $\mathbf{c}$ are translated by vectors from $-U$.

Lemma 7. *Consider* $\mathbf{c}(t) = \sum_{i=0}^n \mathbf{c}_i B_i^n(t)$ *and its subcurve* $\mathbf{b}(t) = \mathbf{c}((1-t)a+tb)$ *with* $0 \le a < b \le 1$. *Let* $\mathbf{c}$ *be* $(b-a)$*-robustly bcp-monotone. Then there exists a unit-length vector* $\mathbf{m}$ *with the following property:*
Whenever a perturbed curve $\tilde{\mathbf{c}}(t) = \sum_{i=0}^n \tilde{\mathbf{c}}_i B_i^n(t)$ *satisfies the error bound* $\mathbf{e}_i := \tilde{\mathbf{c}}_i - \mathbf{c}_i \in -U$ *for* $i = 0, \ldots, n$, *then its subcurve* $\tilde{\mathbf{b}}(t) = \tilde{\mathbf{c}}((1-t)a+tb)$ *is still bcp-increasing in direction* $\mathbf{m}$.

Proof. Let $C = \operatorname{int}\operatorname{conv}\left(\{\Delta\mathbf{b}_0, \ldots, \Delta\mathbf{b}_{n-1}\} + (b-a)\{-1,+1\}^2\right)$. By (3.1), we have $0 \notin C$, so there is a half-plane $\langle \mathbf{m}, \cdot \rangle > 0$ containing C. The lemma claims $\langle \mathbf{m}, \Delta\tilde{\mathbf{b}}_i \rangle > 0$ for all i. To prove this, we show $\Delta\tilde{\mathbf{b}}_i - \Delta\mathbf{b}_i \in (b-a)(-1,+1)^2$ and thus $\Delta\tilde{\mathbf{b}}_i \in C$. To do so, we invoke the theory of *blossoms* [19] [18, 3.1] [4, 4.7]. There exists a unique n-affine symmetric form $\mathbf{c}[t_1, \ldots, t_n]$ with $\mathbf{c}(t) = \mathbf{c}[t, \ldots, t]$. The control points of $\mathbf{b}$ are $\mathbf{b}_i = \mathbf{c}[a, \ldots, a, b, \ldots, b]$, where exactly i arguments are a. From $t = t \cdot 1 + (1-t) \cdot 0$ it follows that $\mathbf{c}[\ldots t \ldots] = t\mathbf{c}[\ldots 1 \ldots] + (1-t)\mathbf{c}[\ldots 0 \ldots]$; hence $\Delta\mathbf{b}_i = \mathbf{c}[\ldots b \ldots] - \mathbf{c}[\ldots a \ldots] = (b-a)(\mathbf{c}[\ldots 1 \ldots] - \mathbf{c}[\ldots 0 \ldots])$ and so $\Delta\tilde{\mathbf{b}}_i - \Delta\mathbf{b}_i = (b-a)(\mathbf{e}[\ldots 1 \ldots] - \mathbf{e}[\ldots 0 \ldots]) \in (b-a)(-U+U) = (b-a)(-1,+1)^2$, as desired. □

Now we give conditions (M1–3) under which a chain $e = (\mathbf{b}^1, \ldots, \mathbf{b}^k)$ with urcurve $\mathbf{b}$ can be merged. The fat vertices at $\mathbf{b}_0^1$ and $\mathbf{b}_n^k$ are the *endpoints* of e. The fat vertices at $\mathbf{b}_0^2, \ldots, \mathbf{b}_0^k$ are *fat inner vertices* of e. The remaining control points $\mathbf{b}_i^j$ ($1 \leq j \leq k$, $0 < i < n$) are *thin inner vertices* of e.

(M1) An inner fat vertex v of e has only one preimage $(\mathbf{b}, t_v)$ (see §3.2) and exactly two incident fat edges. (Necessarily, these two fat edges are the two unrounded fragments of e incident to v, and t_v is the common boundary point of their parameter intervals on $\mathbf{b}$.)

For a point $\mathbf{p}$, we let $\mathcal{N}(\mathbf{p})$ denote the set comprising all fat vertices $v \in N(\mathbf{p}) \setminus \{\mathbf{p}\}$ of T and comprising all fat edges f of T such that $N(\mathbf{p})$ intersects an edge of $CC(f)$ which does not have $\mathbf{p}$ as an endpoint.

(M2) For an inner vertex v of e, all elements of $\mathcal{N}(v)$ have the same urcurve $\mathbf{b}$ as the chain e.

Suppose (M2) holds. If v is an inner fat vertex, let I^v be the smallest subinterval of $[0,1]$ that contains the parameter values of $\mathbf{b}$ for the elements of $\mathcal{N}(v) \cup \{v\}$; or formally, $I^v = \mathrm{conv}(\mathbf{b}^{-1}(\mathcal{N}(v) \cup \{v\}))$. Likewise, if v is an inner thin vertex of e belonging to fat edge f, let $I^v = \mathrm{conv}(\mathbf{b}^{-1}(\mathcal{N}(v) \cup \{f\}))$. In either case, denote by $\mathbf{h}^v$ be the subcurve of $\mathbf{b}$ corresponding to I^v. We call $\mathbf{h}^v$ the *neighbourhood curve* of v. In case v is an inner fat vertex with $\mathcal{N}(v) = \emptyset$, I^v has only one element. Otherwise, $\mathbf{h}^v$ is non-constant, because $\mathbf{b}$ is non-constant. The control points of $\mathbf{h}^v$ are convex combinations of $\mathbf{c}_0, \ldots, \mathbf{c}_n$ with fixed coefficients, so going from $\mathbf{c}$ to $\rho\mathbf{c}$ turns $\mathbf{h}^v$ into the rounded neighbourhood curve $\tilde{\mathbf{h}}^v$, as for fragments. Let $\lambda_{\mathbf{h}^v}$ be the robustness number of $\mathbf{h}^v$, or 0 if $\mathbf{h}^v$ is constant. Recall that I^e is the parameter interval of $\mathbf{c}^e$ on $\mathbf{b}$.

(M3) For an inner vertex v of e, it holds that $I^v \subseteq I^e$ and $|I^v|/|I^e| \leq \lambda_{\mathbf{h}^v}$.

We call a chain e *admissible* if it satisfies (M1–3) or consists of just one fragment. Given the fat graph T as resulting from the conflict removal phase, the merging phase computes admissible chains of maximal length by first forming maximal candidate chains satisfying (M1) and then breaking up candidate chains iteratively until (M2) and (M3) are also satisfied or until the length has dropped to 1. Trivial chains are thrown away. The output of the merging phase are the curves $\mathbf{c}^e$ for all non-trivial maximal admissible chains e. We call these curves the *unrounded meta-fragments*. Each meta-fragment $\mathbf{c}^e$ arises either from a chain of length $k = 1$, in which case we call $\mathbf{c}^e$ a *singleton*, or from a chain of length $k \geq 2$ that satisfies (M1–3). Let E be the set of all unrounded meta-fragments. Let V^* be the set of all fat vertices of T except the inner fat vertices of meta-fragments. The graph (V^*, E) is the unrounded form of the arrangement that we compute.

Observe that every non-trivial fragment appears in exactly one meta-fragment; every trivial fragment appears in at most one meta-fragment. If a fragment appears

in a non-singleton meta-fragment, we call it *merged*, otherwise *unmerged*. If $\mathbf{p}$ is a control point of a merged fragment and is not the endpoint of its meta-fragment, we call $\mathbf{p}$ *merged*, otherwise *unmerged*. We say two unrounded fragments *have a common neighbourhood curve* $\mathbf{h}$, if they belong to the same meta-fragment $\mathbf{c}$, and $\mathbf{c}$ has an inner vertex v such that $\mathbf{h} = \mathbf{h}^v$ comprises both fragments. By (M3) in conjunction with Lemma 7, its rounding $\tilde{\mathbf{h}}$ is bcp-monotone.

3.5 The rounding phase

We obtain a snap rounding $\mathcal{A}'$ of the arrangement $\mathcal{A}$ induced by the set S of ur-curves as follows. The vertex set of $\mathcal{A}'$ is $V' := \rho V^* = \{\rho(v) \mid v \in V^*\}$. This unites previously distinct vertices within a pixel. The edge set is $E' := \rho E := \{\rho\mathbf{c}^e \mid e \in E\}$, whose elements we call *rounded meta-fragments*. Each rounded meta-fragment $\rho\mathbf{c}$ is computed from a corresponding unrounded meta-fragment $\mathbf{c}$ by just rounding each control point $\mathbf{c}_i$ to $\rho(\mathbf{c}_i)$. (As $\mathbf{c}$ is non-trivial, $\rho\mathbf{c}$ is non-constant.) In our graph, we regard $\rho\mathbf{c}$ as an edge from vertex $\rho\mathbf{c}(0)$ to vertex $\rho\mathbf{c}(1)$. That concludes the algorithm. We analyze the properties of $\mathcal{A}'$ in the next section.

4 Geometric analysis

To what extent does the rounded arrangement $\mathcal{A}'$ reflect the geometry of the true arrangement $\mathcal{A}$?

Theorem 8. *Let* $\mathbf{c}$ *be an unrounded meta-fragment. The rounded meta-fragment* $\rho\mathbf{c}$ *lies in the* sausage region $\mathbf{c}([0,1]) + (-U)$ *of* $\mathbf{c}$*, and thus also in the sausage region of the urcurve* $\mathbf{b}$ *of* $\mathbf{c}$*. Conversely, let* $\mathbf{b}$ *be an urcurve. Then there exists a subset* $F_\mathbf{b}$ *of* $V' \cup E'$ *such that* $\mathbf{b}([0,1])$ *is contained in* $(\bigcup F_\mathbf{b}) + U$*. Finally, let* $\mathbf{p}$ *be an endpoint of an urcurve, an irregular point of an urcurve, a self-intersection point of an urcurve, or an intersection point of two urcurves. Then* $\rho(\mathbf{p})$ *is a vertex in* V'*.*

Proof. Consider $\mathbf{c}$. For any $t \in [0,1]$ we have $(\rho\mathbf{c})(t) - \mathbf{c}(t) = \sum_{i=0}^{n} (\rho(\mathbf{c}_i) - \mathbf{c}_i) B_i^n(t) \in -U$, The claim on $\mathbf{b}$ is immediate from $\mathbf{c}([0,1]) \subseteq \mathbf{b}([0,1])$. The converse inclusion holds, because the only subcurves $\mathbf{b}^*$ of $\mathbf{b}$ that we ever delete are trivial, but we retain the endpoint $\mathbf{b}^*(0)$ as a non-mergeable fat vertex, and we have $\mathbf{b}^*([0,1]) \subseteq \rho(\mathbf{b}^1(0)) + U$. The statement on $\mathbf{p}$ is easily verified by inspection of the algorithm. □

So much for approximate preservation of metric properties; now we turn to the topological properties of $\mathcal{A}'$. We generalize the approach of [8] from independent straight-line segments to our setting with enclosing polygons: We view rounding as a deformation $\mathcal{D}_s$ continuous in time $s \in [0,1]$ that interpolates between "zero snaproundedness" at $s = 0$ and "full snaproundedness" at $s = 1$. $\mathcal{D}_s$ deforms the control cliques of unrounded fragments to the control cliques of rounded fragments.[1] To define $\mathcal{D}_s$, we distinguish those pixels as *hot* that contain unmerged control points of at least *two* different unrounded fragments.

Lemma 9. *No control clique edge passes through a hot pixel.*
All control points in a hot pixel are unmerged and thus round to the pixel center.

[1] This notion of "deformation" matches [8]. It is not a deformation of the entire plane as in [16].

Proof. The first claim is an immediate consequence of conflict removal: If a CC edge of one unrounded fragment would pass through a hot pixel, there would still be a conflict with the CC vertex of the other fragment. For the second claim, assume there is a merged control point $\mathbf{r}$ in the hot pixel. Its neighbourhood curve $\mathbf{h}$ has to comprise both unmerged control points. As the unmerged control points coincide in the pixel center after rounding, this contradicts the bcp-monotonicity of $\tilde{\mathbf{h}}$. □

This lemma does not hold for a pixel with just one control point, hence our new definition of "hot".

Consider the control clique of an unrounded fragment $\mathbf{b}^j$. For every hot pixel V, we split a CC edge $\overline{\mathbf{pq}}$ by introducing an additional vertex $\mathbf{a}$ wherever the edge intersects the pixel boundary ∂V, and we assign the center of V as rounded position $\tilde{\mathbf{a}}$ to the vertex $\mathbf{a}$. (If $\overline{\mathbf{pq}}$ intersects the common boundary line segment of two adjacent hot pixels, this means we introduce two new vertices at that point, one belonging to each hot pixel.) The result of this we call *extended control clique*, or *ECC*. There are three kinds of vertices in an ECC: the *boundary vertices* which we just introduced; the *internal vertices* inside and the *external vertices* outside hot pixels. Likewise, there are two kinds of ECC edges: *internal edges* within hot pixels and *external edges* outside hot pixels. By Lemma 9, all internal edges have at least one internal vertex among their endpoints.

Let us now define how $\mathcal{D}_s$ acts on ECCs. Each ECC vertex $\mathbf{a}$ has been assigned a rounded position $\tilde{\mathbf{a}}$. (For control points this was done in §3.4.) $\mathcal{D}_s$ interpolates ECC vertices linearly: $\mathcal{D}_s(\mathbf{a}) = (1-s)\mathbf{a} + s\tilde{\mathbf{a}}$. We have $\mathcal{D}_s(\mathbf{p}) - \mathbf{p} \in s(-\frac{1}{2}, +\frac{1}{2}]^2 = -s \cdot U$ for internal and external ECC vertices (i.e., fragment control points) and $\mathcal{D}_s(\mathbf{a}) - \mathbf{a} \in s[-\frac{1}{2}, +\frac{1}{2}]^2$ for boundary ECC vertices. For an ECC edge $\overline{\mathbf{pq}}$, we define $\mathcal{D}_s(\overline{\mathbf{pq}}) = \overline{\mathcal{D}_s(\mathbf{p})\,\mathcal{D}_s(\mathbf{q})}$.

At $s = 1$, all ECC boundary vertices and internal edges collapse with the respective internal vertices at pixel centers. Thus, the deformed ECC coincides with the control clique of the rounded fragment.

For any $s \in [0,1]$, the deformed ECC is a graph in the plane; its edges are straight-line segments that may cross. If we imagine further vertices at crossing points (*crossing vertices*), we have a planar graph with exactly one unbounded face. The union of all vertices, edges, and bounded faces is a closed polygon, which we call *ECC polygon* $\mathcal{P}_s(\mathbf{b}^j)$ of the fragment $\mathbf{b}^j$ at time s. We have $\mathcal{P}_0(\mathbf{b}^j) = \operatorname{conv}\{\mathbf{b}_i^j\}_i$ and $\mathcal{P}_1(\mathbf{b}^j) = \operatorname{conv}\{\tilde{\mathbf{b}}_i^j\}_i$, so that the ECC polygon encloses the fragment at $s = 0$ and $s = 1$, respectively. This is the link between the ECC polygons and curves. However, we have no control over curves defined by control points $\mathcal{D}_s(\mathbf{b}_i^j)$ for $0 < s < 1$, because we cannot break Bézier control polygons at the boundaries of hot pixels.

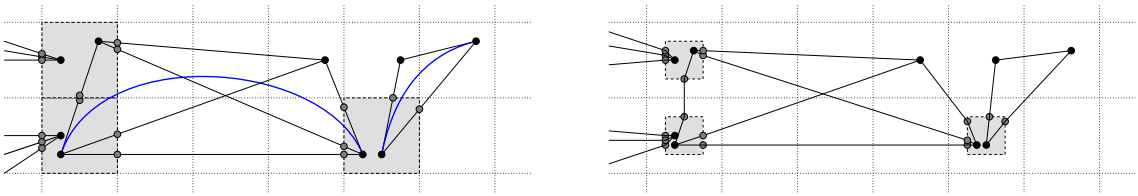

Figure 4.1: extended control cliques under the deformation $\mathcal{D}_s$ at $s = 0$ (left) and $s = \frac{1}{2}$ (right).

How do deformed ECC polygons interact? We begin with the case of a common neighbourhood curve.

Proposition 10. *Let* $\mathbf{c}$ *be an unrounded meta-fragment of degree n that consists of fragments* $\mathbf{b}^1, \ldots, \mathbf{b}^k$. *Consider indices* $1 \le j_1 < j_2 \le k$ *such that* $\mathbf{b}^{j_1}$ *and* $\mathbf{b}^{j_2}$ *have a common neighbourhood curve* $\mathbf{h}$.

(i) If $j_2 > j_1 + 1$, *then* $\forall s \in [0,1]\colon \mathcal{P}_s(\mathbf{b}^{j_1}) \cap \mathcal{P}_s(\mathbf{b}^{j_2}) = \emptyset$.

(ii) If $j_2 = j_1 + 1$, *then* $\forall s \in [0,1]\colon \mathcal{P}_s(\mathbf{b}^{j_1}) \cap \mathcal{P}_s(\mathbf{b}^{j_2}) = \{\mathcal{D}_s(\mathbf{b}_n^{j_1})\} = \{\mathcal{D}_s(\mathbf{b}_0^{j_2})\}$.

Proof. We show (ii); the proof of (i) is similar and simpler. Overloading notation, let $\mathcal{D}_s(\mathbf{h}_i) = (1-s)\mathbf{h}_i + s\tilde{\mathbf{h}}_i$. By (M3) and Lemma 7, there exists $\mathbf{m}$ such that $\mathcal{D}_s(\mathbf{h}_0), \ldots, \mathcal{D}_s(\mathbf{h}_n)$ is $\mathbf{m}$-increasing for all $s \in [0,1]$. By Lemma 3(iii), also $\mathcal{D}_s(\mathbf{b}_0^{j_1})$, $\mathcal{D}_s(\mathbf{b}_1^{j_1})$, $\ldots$, $\mathcal{D}_s(\mathbf{b}_{n-1}^{j_1})$, $\mathcal{D}_s(\mathbf{b}_n^{j_1}) = \mathcal{D}_s(\mathbf{b}_0^{j_2})$, $\mathcal{D}_s(\mathbf{b}_1^{j_2})$, $\ldots$, $\mathcal{D}_s(\mathbf{b}_n^{j_2})$ is $\mathbf{m}$-increasing. Thus, the line $\ell_s(\mathbf{x}) := \langle \mathbf{m}, \mathbf{x} - \mathcal{D}_s(\mathbf{b}_n^{j_1}) \rangle$ separates the internal and external ECC vertices of $\mathbf{b}^{j_1}$ from those of $\mathbf{b}^{j_2}$, except for $\mathcal{D}_s(\mathbf{b}_n^{j_1}) = \mathcal{D}_s(\mathbf{b}_0^{j_2})$. Likewise, ℓ_s separates the boundary ECC vertices: This holds for $s = 0$ and $s = 1$, because then the boundary vertices are convex combinations of CC edge endpoints. So if $\mathbf{a}$ is any boundary ECC vertex, the linear expression $\ell_s(\mathcal{D}_s(\mathbf{a}))$ has the same sign at $s = 0$ and $s = 1$ and hence at any $s \in [0,1]$. It follows that the halfplanes $\ell_s(\,\cdot\,) \lesseqgtr 0$ contain one ECC polygon each, except for the common vertex $\mathcal{D}_s(\mathbf{b}_n^{j_1})$ on the boundary line. □

Now we look at fragments without common neighbourhood curve.

Lemma 11 (Main Lemma). *Consider two unrounded fragments* $\mathbf{b}$ *and* $\mathbf{b}^*$ *that do not have a common neighbourhood curve. Let* $\overline{\mathbf{pq}}$ *be an ECC edge of* $\mathbf{b}$ *and* $\mathbf{r}$ *an ECC vertex of* $\mathbf{b}^*$ *such that* $\mathbf{r} \notin \{\mathbf{p}, \mathbf{q}\}$.

(i) $\forall s \in [0,1)\colon \mathcal{D}_s(\mathbf{r}) \notin \overline{\mathcal{D}_s(\mathbf{p})\,\mathcal{D}_s(\mathbf{q})}$.

(ii) $\mathcal{D}_1(\mathbf{r}) \notin \overline{\mathcal{D}_1(\mathbf{p})\,\mathcal{D}_1(\mathbf{q})} \vee \mathcal{D}_1(\mathbf{r}) \in \{\mathcal{D}_1(\mathbf{p}), \mathcal{D}_1(\mathbf{q})\}$.
If $\mathbf{r}$ *is an external ECC vertex, the second alternative is excluded.*

Proof. We begin with an observation on a hot pixel $\mathbf{a} + U$ ($\mathbf{a} \in \mathbb{Z}^2$). Consider the squares $Q_s = \mathbf{a} + (1-s)[-\frac{1}{2}, +\frac{1}{2}]^2$. Q_0 is the closure of the hot pixel. By Lemma 9, all ECC vertices in it round to $\mathbf{a}$. Thus, for all ECC vertices and edges in Q_0, $\mathcal{D}_s$

acts as the affine shrinking map $Q_0 \to Q_s$. For $s < 1$, this map is bijective, but for $s = 1$, the entire pixel collapses to $Q_1 = \{\mathbf{a}\}$. Let $\mathbf{u}$ be any point on any ECC edge $\overline{\mathbf{vw}}$ outside Q_0. There is a line ℓ comprising a boundary edge of Q_0 that separates $\mathbf{u}$ from the pixel. We assume w.l.o.g. that it is the left edge of Q_0. Then ℓ equals ℓ_0, where $\ell_s := \mathbf{a} + (\{-\frac{1}{2}(1-s)\} \times \mathbb{R})$. The speed vector $\frac{\mathrm{d}}{\mathrm{d}s}\mathcal{D}_s(\mathbf{u})$ is a convex combination of $\frac{\mathrm{d}}{\mathrm{d}s}\mathcal{D}_s(\mathbf{v})$, $\frac{\mathrm{d}}{\mathrm{d}s}\mathcal{D}_s(\mathbf{w})$ and thus an element of $[-\frac{1}{2}, +\frac{1}{2}]^2$. However, the line ℓ_s moves right with speed $\frac{1}{2}$, so for all $s \in [0,1]$, the point $\mathcal{D}_s(\mathbf{u})$ remains left and Q_s remains right of or on ℓ_s.

We have thus shown: *No point on an ECC edge, in particular no ECC vertex, moves from the outside into a shrinking hot pixel during the deformation $\mathcal{D}$.* We can now prove the lemma by case distinction.

Case 1: $\overline{\mathbf{pq}}$ is an internal edge. If $\mathbf{r}$ lies outside the hot pixel comprising $\overline{\mathbf{pq}}$, it never enters that pixel and thus cannot become a point of $\overline{\mathcal{D}_s(\mathbf{p})\,\mathcal{D}_s(\mathbf{q})}$ for any s. If $\mathbf{r}$ lies inside that hot pixel or on its boundary, then initially $\mathbf{r} \notin \overline{\mathbf{pq}}$, because $\mathbf{r}$ is neither an endpoint (by premise) nor any other point (by construction). Hence $\mathcal{D}_s(\mathbf{r}) \notin \overline{\mathcal{D}_s(\mathbf{p})\,\mathcal{D}_s(\mathbf{q})}$ for $s < 1$, proving (i); and $\{\mathcal{D}_1(\mathbf{r})\} = \overline{\mathcal{D}_1(\mathbf{p})\,\mathcal{D}_1(\mathbf{q})}$, proving (ii).

Case 2: $\overline{\mathbf{pq}}$ is an external edge. First, let $\mathbf{r}$ be an internal or boundary ECC vertex of some hot pixel Q_0. The points of $\overline{\mathcal{D}_s(\mathbf{p})\,\mathcal{D}_s(\mathbf{q})}$ stay outside the shrinking hot pixel Q_s, with the potential exception of an endpoint $\mathcal{D}_s(\mathbf{p})$ that is a boundary ECC vertex of Q_s and coincides with $\mathcal{D}_s(\mathbf{r})$ in the pixel center at $s = 1$.

Next, let $\mathbf{r}$ be an unmerged external ECC vertex. We have $\mathcal{D}_1(\mathbf{r}) = \rho(\mathbf{r})$, and we know from conflict removal that $\overline{\mathbf{pq}}$ does not intersect $\rho(\mathbf{r}) + U$. In complete analogy to the argument above, we see that $\overline{\mathcal{D}_s(\mathbf{p})\,\mathcal{D}_s(\mathbf{q})}$ does not enter the shrinking pixel containing $\mathcal{D}_s(\mathbf{r})$, so $\mathcal{D}_s(\mathbf{r}) \notin \overline{\mathcal{D}_s(\mathbf{p})\,\mathcal{D}_s(\mathbf{q})}$ for all $s \in [0,1]$.

Finally, let $\mathbf{r}$ be a merged external ECC vertex. It moves from $\mathcal{D}_0(\mathbf{r}) = \mathbf{r} \in \rho(\mathbf{r}) + U$ along $\mathcal{D}_s(\mathbf{r}) \in \mathbf{r} + (-s) \cdot U \subseteq \rho(\mathbf{r}) + (U + (-U)) = \rho(\mathbf{r}) + (-1,+1)^2$. Since $\mathbf{b}$ and $\mathbf{b}^*$ have no common neighbourhood curve, $\overline{\mathbf{pq}}$ does not intersect $N(\mathbf{r}) = \rho(\mathbf{r}) + [-\frac{3}{2}, +\frac{3}{2})^2$. As s grows, no point on $\overline{\mathcal{D}_s(\mathbf{p})\,\mathcal{D}_s(\mathbf{q})}$ changes by more than $\frac{1}{2}$ in any coordinate. Hence $\overline{\mathcal{D}_s(\mathbf{p})\,\mathcal{D}_s(\mathbf{q})} \cap (\rho(\mathbf{r}) + (-1,+1)^2) = \emptyset$ for all s, proving the claim. □

Proposition 12. *Let the two unrounded fragments $\mathbf{b}$ and $\mathbf{b}^*$ not have a common neighbourhood curve.*

(i) If $\mathcal{P}_0(\mathbf{b}) \cap \mathcal{P}_0(\mathbf{b}^*) = \emptyset$, then

$$\mathcal{P}_s(\mathbf{b}) \cap \mathcal{P}_s(\mathbf{b}^*) = \begin{cases} \emptyset & \text{for } s < 1, \\ \partial\mathcal{P}_1(\mathbf{b}) \cap \partial\mathcal{P}_1(\mathbf{b}^*) & \text{for } s = 1. \end{cases}$$

(ii) If $\mathcal{P}_0(\mathbf{b}) \cap \mathcal{P}_0(\mathbf{b}^*) \neq \emptyset$, then $\mathcal{P}_0(\mathbf{b}) \cap \mathcal{P}_0(\mathbf{b}^*) = \{\mathbf{r}\}$ where $\mathbf{r}$ is a common

endpoint, and

$$\mathcal{P}_s(\mathbf{b}) \cap \mathcal{P}_s(\mathbf{b}^*) = \begin{cases} \{\mathcal{D}_s(\mathbf{r})\} & \text{for } s < 1, \\ \partial\mathcal{P}_1(\mathbf{b}) \cap \partial\mathcal{P}_1(\mathbf{b}^*) \supseteq \{\mathcal{D}_1(\mathbf{r})\} & \text{for } s = 1. \end{cases}$$

Proof. By construction, $\mathrm{conv}\{\mathbf{b}_i\}_i$ and $\mathrm{conv}\{\mathbf{b}_i^*\}_i$ intersect at most in common endpoints. If they had two endpoints in common, then, by convexity, also the straight-line segment joining them – a contradiction. This shows the initial claim in (ii). For the remainder, we defer the technicalities to the appendix and just state the general principle: By continuity of $\mathcal{D}_s$, a common interior point cannot occur for any $s \leq 1$, because this would require an additional common boundary point at $s' < s$, but that is excluded by Lemma 11(i). □

Corollary 13. *All control points of $\tilde{\mathbf{b}}$ in $\partial\mathcal{P}_1(\mathbf{b}) \cap \partial\mathcal{P}_1(\mathbf{b}^*)$ are also control points of $\tilde{\mathbf{b}}^*$ and vice versa; so $\partial\mathcal{P}_1(\mathbf{b}) \cap \partial\mathcal{P}_1(\mathbf{b}^*)$ is a chain of common boundary vertices and edges. If, after rounding, $\tilde{\mathbf{b}}_i = \tilde{\mathbf{b}}_j^*$, then $\mathbf{b}_i$, $\mathbf{b}_j^*$ are unmerged control points or have already been equal before rounding.*

Proof. Immediate with Lemma 11(ii); noting that merged control points are external ECC vertices. □

Remark 14. The statements of the preceding proposition and corollary include the statements of Proposition 10 as special cases and are thus valid for any pair of fragments.

We are now ready to prove that the rounded vertices and edges of $\mathcal{A}'$ really form an arrangement.

Proposition 15. *Consider two meta-fragments $\mathbf{c}$, $\mathbf{c}^*$ and the edges of $\mathcal{A}'$ resulting from them after rounding. If $\rho\mathbf{c}([0,1]) \neq \rho\mathbf{c}^*([0,1])$, then $\rho\mathbf{c}([0,1]) \cap \rho\mathbf{c}^*([0,1]) \subseteq \{\rho\mathbf{c}(0), \rho\mathbf{c}(1)\}$ ("intersections are endpoints").*

Proof. If $\mathbf{c}$ is non-singleton, it is immediate from Corollary 13 that the polygonal enclosure of $\rho\mathbf{c}$ meets the enclosing polygon of any fragment of $\rho\mathbf{c}^*$ at most in $\rho\mathbf{c}(0)$ or $\rho\mathbf{c}(1)$. Hence let $\mathbf{c}$ be a singleton meta-fragment and distinguish two cases. If $\rho\mathbf{c}$ is not straight, $\rho\mathbf{c}((0,1)) \subseteq \mathrm{int}\,\mathrm{conv}\{\rho(\mathbf{c})_i\}_i$ by Lemma 4(ii), and that set is disjoint from the polygonal enclosure of $\rho\mathbf{c}^*$ by Proposition 12. So let $\rho\mathbf{c}$ be straight and singleton. As $\rho\mathbf{c}([0,1])$ is equal to the ECC edge $\overline{\rho(\mathbf{c}_0)\,\rho(\mathbf{c}_n)}$, the control points $\tilde{\mathbf{b}}_0, \ldots, \tilde{\mathbf{b}}_{n'}$ of any fragment $\tilde{\mathbf{b}}$ of $\rho\mathbf{c}^*$ can coincide with $\rho(\mathbf{c}_0)$ or $\rho(\mathbf{c}_n)$ but not, by Lemma 11(ii), with $\rho(\mathbf{c}_1), \ldots, \rho(\mathbf{c}_{n-1})$. So if a boundary edge $\overline{\tilde{\mathbf{b}}_{j_1}\,\tilde{\mathbf{b}}_{j_2}}$ of $\mathrm{conv}\{\tilde{\mathbf{b}}_i\}_i$ intersects $\overline{\rho(\mathbf{c}_0)\,\rho(\mathbf{c}_n)}$ in more than either $\rho(\mathbf{c}_0)$ or $\rho(\mathbf{c}_n)$, then $\overline{\tilde{\mathbf{b}}_{j_1}\,\tilde{\mathbf{b}}_{j_2}} = \overline{\rho(\mathbf{c}_0)\,\rho(\mathbf{c}_n)}$. In that case, we distinguish further whether $\tilde{\mathbf{b}}$ is straight. If not, the boundary edge $\overline{\tilde{\mathbf{b}}_{j_1}\,\tilde{\mathbf{b}}_{j_2}}$ does not contain a point of $\rho\mathbf{c}^*$ except maybe at its

endpoints, so the claim holds. But if $\tilde{\mathbf{b}}$ is straight, then both $\rho\mathbf{c}$ and $\tilde{\mathbf{b}}$ have control polygons $\rho(\mathbf{c}(0)),\dots,\rho(\mathbf{c}(0)),\rho(\mathbf{c}(1)),\dots,\rho(\mathbf{c}(1))$ and hence are equal (as point sets). □

Corollary 16. *If $\mathbf{c} \neq \mathbf{c}^*$ and $\rho\mathbf{c}([0,1]) = \rho\mathbf{c}^*([0,1])$, then $\rho\mathbf{c}$ and $\rho\mathbf{c}^*$ are straight singleton meta-fragments whose control polygons consist of repetitions of one endpoint followed by repetitions of the other endpoint.*

Proposition 17. *Let $\rho\mathbf{c}$ be a rounded meta-fragment. If $0 < t < 1$, then $\rho\mathbf{c}'(t) \neq 0$ ("no interior irregular points"). If $0 \leq t_1 < t_2 \leq 1$ and $\rho\mathbf{c}(t_1) = \rho\mathbf{c}(t_2)$, then $t_1 = 0$ and $t_2 = 1$ ("no interior self-intersection").*

Proof (sketch). The full proof does not provide new insights, so we defer it to the appendix. Its basic ideas are: (i) (weak) bcp-monotonicity of rounded (un)merged fragments excludes irregular points; (ii) interior self-intersections are excluded by Propositions 10 and 12, similar to the proof of Proposition 15. □

The topology of an arrangement is given by the cyclic order of edges around vertices. So we show that the orientation of any three edges adjacent to one vertex in $\mathcal{A}'$, if not destroyed by collape of edges, agrees with the unrounded situation. In an arrangement of curves, there may be loops. Hence we do not consider entire edges $\rho\mathbf{c}$, consisting of fragments $\tilde{\mathbf{b}}^1,\dots,\tilde{\mathbf{b}}^k$, but instead the *terminal fragment* $\tilde{\mathbf{b}}^1$ or $\tilde{\mathbf{b}}^k$ of each edge.

Theorem 18. *Consider a vertex $\mathbf{a} \in \mathbb{Z}^2$ of $\mathcal{A}'$. Let $\rho\mathbf{c}^1$, $\rho\mathbf{c}^2$, $\rho\mathbf{c}^3$ be three rounded meta-fragments incident to $\mathbf{a}$. For $j = 1,2,3$, let $\tilde{\mathbf{b}}^j$ be a terminal fragment of $\rho\mathbf{c}^j$ incident to $\mathbf{a}$ so that the cyclic order of $(\tilde{\mathbf{b}}^1,\tilde{\mathbf{b}}^2,\tilde{\mathbf{b}}^3)$ around $\mathbf{a}$ is positive (i.e., counterclockwise). For $j = 1,2,3$, there exist intersection points $\mathbf{p}^j$ of unrounded fragments $\mathbf{b}^j([0,1])$ and the pixel boundary $A := \partial(\mathbf{a}+U)$. Furthermore, any triple $(\mathbf{p}^1,\mathbf{p}^2,\mathbf{p}^3)$ of such points has positive cyclic order on the topological circle A.*

Proof. Let $j \in \{1,2,3\}$. Exactly one of $\mathbf{b}^j(0)$, $\mathbf{b}^j(1)$ lies in $V = \mathbf{a}+U$: if $\mathbf{b}^j$ is merged, this follows from Lemma 9; if $\mathbf{b}^j$ is unmerged, we know $\tilde{\mathbf{b}}^j$ is weakly bcp-monotone, so $\rho(\mathbf{b}^j_0) = \tilde{\mathbf{b}}^j(0) \neq \tilde{\mathbf{b}}^j(1) = \rho(\mathbf{b}^j_n)$, hence only one endpoint them rounds to $\mathbf{a}$, the other one lies outside V. In either case, $\mathbf{b}^j$ intersects $A = \partial V$ in some point $\mathbf{p}^j$. By Proposition 5, any choice of $\mathbf{p}^j \in \operatorname{conv}\{\mathbf{b}^j_i\}_i \cap A$ results in the same cyclic order of $(\mathbf{p}^1,\mathbf{p}^2,\mathbf{p}^3)$, so the sets $\operatorname{conv}\{\mathbf{b}^j_i\}_i \cap A$ have a well-defined cyclic order on A; in particular, this is the cyclic order of any triple $\mathbf{p}^j \in \mathbf{b}^j([0,1]) \cap A$, $j = 1,2,3$. It remains to argue that this cyclic order of ECC polygons is preserved during rounding. This is because $\mathcal{D}_s$ shrinks pixel boundaries continuously for all $s \in [0,1]$ and bijectively for $s < 1$. If no collapse happens at $s = 1$, the cyclic order is preserved. □

5 Concluding remarks and future work

We have presented a new algorithm for approximate arrangement computation of Bézier curves. Its salient and novel feature are the topological guarantees on the output; in particular, inversions of topology are avoided. Thereby, the algorithm produces a geometric rounding of the true arrangement induced by its input. To achieve this, we have replaced the ε-approximation of points in conventional intersection algorithms with locating points in a pixel grid, followed by generalized snap rounding of control polygons onto the grid. Regarding implementation, we remark that for input control points whose coordinates are integers, or more generally finite binary fractions, all control point coordinates constructed are also finite binary fractions; costly rational and algebraic coordinates are avoided.

Clearly, the number of subdivisions determines the complexity of the algorithm. Unfortunately, this number has no bound in terms of the number of input curves: even for two curves, any fixed number of subdivisions may be insufficient. Despite this shortcoming, intersection computation by repeated subdivision is very widely used in practice, and adding a guard against inversion of topology has its merits. It remains for future work to identify good parameters for a rigorous analysis.

We have given a deformation proof for topology preservation when rounding chains of fragments (Section 4), which is applicable to a wider class of algorithms. Such algorithms may drive subdivision and merging in more sophisticated ways, e.g., to allow a unique static definition of the output in terms of the input geometry, or to attain guarantees on the success of merging in terms of the input geometry. We expect that our proof technique may serve as a starting point for research into rounding arrangements of other classes of curves that are defined in terms of control points, or consist of Bézier pieces, such as B-splines.

Bibliography

[1] M. de Berg, D. Halperin, and M. Overmars. An intersection-sensitive algorithm for snap rounding. *Comput. Geom.*, 36(4):159–165, 2007.

[2] O. Devillers and P. Guigue. Inner and outer rounding of set operations on lattice polygonal regions. In *Proc. 20th Annu. Sympos. Comput. Geom.*, pages 429–437, 2004.

[3] A. Fabri, G.-J. Giezeman, L. Kettner, S. Schirra, and S. Schönherr. On the design of CGAL, a computational geometry algorithms library. *Software – Practice and Experience*, 30:1167–1202, 2000.

[4] G. Farin. *Curves and Surfaces for Computer-Aided Geometric Design*. Academic Press, 4th edition, 1997.

[5] M. Goodrich, L. J. Guibas, J. Hershberger, and P. Tanenbaum. Snap rounding line segments efficiently in two and three dimensions. In *Proc. ACM Sympos. Comput. Geom.*, pages 284–293, 1997.

[6] D. H. Greene. Integer line segment intersection. Unpublished manuscript, cited after [8].

[7] D. H. Greene and F. F. Yao. Finite-resolution computational geometry. In *Proc. 27th Annu. IEEE Sympos. Found. Comput. Sci.*, pages 143–152, 1986.

[8] L. J. Guibas and D. H. Marimont. Rounding arrangements dynamically. *Internat. J. of Comput. Geometry & Applications*, 8:157–176, 1998.

[9] D. Halperin and E. Leiserowitz. Controlled perturbation for arrangements of circles. *Internat. J. of Comput. Geometry & Applications*, 14(4-5):277–310, 2004. Special issue, papers from SCG 2003.

[10] D. Halperin and E. Packer. Iterated snap rounding. *Comput. Geom.*, 23(2):209–225, 2002.

[11] J. D. Hobby. Practical segment intersection with finite precision output. *Comput. Geom.*, 13:199–214, 1999.

[12] L. Kettner and S. Näher. Two computational geometry libraries: LEDA and CGAL. In J. E. Goodman and J. O'Rourke, editors, *Handbook of Discrete and Computational Geometry*, chapter 65, pages 1435–1463. CRC Press LLC, Boca Raton, FL, second edition, 2004.

[13] K. Mehlhorn and S. Näher. *LEDA: A Platform for Combinatorial and Geometric Computing*. Cambridge University Press, 1999.

[14] K. Mehlhorn, R. Osbild, and M. Sagraloff. Reliable and efficient computational geometry via controlled perturbation. In *Automata, Languages and Programming, 33rd Internat. Colloquium, ICALP 2006, Part I*, volume 4051 of *LNCS*, pages 299–310. Springer, 2006.

[15] V. Milenkovic and E. Sacks. An approximate arrangement algorithm for semi-algebraic curves. In *Proc. 22nd Annu. Sympos. Comput. Geom.*, pages 237–246, 2006. Full version to appear in *Internat. J. of Comput. Geometry & Applications*.

[16] V. J. Milenkovic. Shortest path geometric rounding. *Algorithmica*, 27:57–86, 2000.

[17] V. J. Milenkovic and L. R. Nackman. Finding compact coordinate representations for polygons and polyhedra. *IBM J. Res. Develop.*, 34(5):753–769, Sept. 1990.

[18] H. Prautzsch, W. Boehm, and M. Paluszny. *Bézier and B-Spline Techniques*. Springer, 2002.

[19] L. Ramshaw. Blossoms are polar forms. Research Report 34, Digital, Systems Research Center, Palo Alto, CA, USA, January 1989. `ftp://ftp.digital.com/pub/DEC/SRC/research-reports/SRC-034.pdf`.

[20] C. K. Yap. Robust geometric computation. In J. E. Goodman and J. O'Rourke, editors, *Handbook of Discrete and Computational Geometry*, chapter 41, pages 927–952. CRC Press, Boca Raton, FL, 2nd edition, 2004.

[21] C. K. Yap. Complete subdivision algorithms, I: Intersection of Bezier curves. In *Proc. 22nd Annu. Sympos. Comput. Geom.*, pages 217–226, 2006.

Appendix A Proof details

In the proof of **Lemma 6** (page 11), we claimed that *any connected component* C_1 *of* $\mathrm{conv}\{\mathbf{b}_i\}_i \setminus V$ *has to contain some control point* $\mathbf{b}_{j_1}$.

Proof of claim. $C_1 \neq \emptyset$, so there exists $\mathbf{p} \in C_1$. As $\mathbf{p} \notin V$, there exists a half-plane H that contains $\mathbf{p}$ but is disjoint from V. As the complementary half-plane $\mathbb{R}^2 \setminus H$ is convex but does not contain the element $\mathbf{p}$ of $\mathrm{conv}\{\mathbf{b}_i\}_i$, it cannot contain all points $\mathbf{b}_i$. Hence there is $\mathbf{b}_{j_1} \in H$; it remains to show $\mathbf{b}_{j_1} \in C_1$. To see this, we observe that $\mathrm{conv}\{\mathbf{b}_i\}_i \cap H$ is convex and thus connected. As $\mathbf{p} \in \mathrm{conv}\{\mathbf{b}_i\}_i \cap H \subseteq \mathrm{conv}\{\mathbf{b}_i\}_i \setminus V$, it follows that $\mathrm{conv}\{\mathbf{b}_i\}_i \cap H \subseteq C_1$ and so $\mathbf{b}_{j_1} \in C_1$. □

Proof of **Proposition 12** *(page 18).* We already showed that $\mathcal{P}_0(\mathbf{b}) \cap \mathcal{P}_0(\mathbf{b}^*)$ contains at most a common endpoint but did not carry out the arguments "by continuity". We begin with claim (i). Let N denote the total number of internal, boundary, and external vertices in the ECCs of $\mathbf{b}$ and $\mathbf{b}^*$. All their possible positions in the plane are given by N pairs of coordinates. Concatenating them, we get one vector in $\mathbb{R}^{2N}$. Vice versa, any vector in $\mathbb{R}^{2N}$ gives positions for the ECC vertices and thus defines two ECC polygons $\mathcal{Q}$ and $\mathcal{Q}^*$. We consider two subsets of $\mathbb{R}^{2N}$ defined by conditions on $\mathcal{Q}$ and $\mathcal{Q}^*$ as resulting from the respective vertex positions.

$$U_1 = \{\mathcal{Q} \cap \mathcal{Q}^* = \emptyset\} \qquad \text{("have no intersection"),}$$
$$U_2 = \{\operatorname{int}\mathcal{Q} \cap \operatorname{int}\mathcal{Q}^* \neq \emptyset\} \qquad \text{("have interior intersection").}$$

It is clear that U_1 and U_2 are open subsets of $\mathbb{R}^{2N}$, as their defining conditions remain valid under sufficiently small perturbations of vertex locations. It is also clear that the open sets U_1 and U_2 are disjoint. Thus, the topological space $U_1 \cup U_2$ is *not connected*. By the standard argument from elementary point-set topology, it follows that there is no continuous path in $U_1 \cup U_2$ that connects a point in U_1 with a point in U_2; rather, any such path has to pass through the complement $C := \mathbb{R}^{2N} \setminus (U_1 \cup U_2)$.

The deformation $\mathcal{D}_s$, $s \in [0,1]$, induces a continuous path in $\mathbb{R}^{2N}$, which starts at $s = 0$ with an element of U_1. We will demonstrate that it does not meet a point in C for $s < 1$.

An easy argument shows: If two full-dimensional polygons (or more generally, two sets that are closures of open sets) have a common point that is an interior point of one of them, then they also have a common point that is an interior point of both of them. With that, we can describe C as

$$C = \{\emptyset \neq \mathcal{Q} \cap \mathcal{Q}^* = \partial\mathcal{Q} \cap \partial\mathcal{Q}^*\} \qquad \text{(“have boundary intersection only”).}$$

If two polygons intersect in this way, then, by elementary geometric arguments, there is a convex corner point v of one polygon that lies on a boundary edge of the other polygon. $\mathcal{D}_s$ does not reach such a situation for $s < 1$: it does not happen for an ECC vertex v because of Lemma 11(i); neither does it happen with an additional crossing vertex v used for the definition of ECC polygons, because the corner at a crossing vertex is always concave.

Let us summarize: We have partitioned the space $\mathbb{R}^{2N}$ of all possible ECC vertex positions into the three sets U_1, C, and U_2. The path induced by $\mathcal{D}_s$ in $\mathbb{R}^{2N}$ begins at $s = 0$ in U_1. For $s < 1$, it does not meet C and is thus contained in $U_1 \cup U_2$. However, the open sets U_1 and U_2 are disconnected, hence the path stays in U_1 for $s < 1$ and in $U_1 \cup C$ for $s \leq 1$. That proves (i).

The argument for (ii) is similar but needs to take the following aspects into account. When counting the number N of ECC vertices, the common vertex $\mathbf{r}$ is counted only once. In all conditions on ECC polygons, “[no] intersection” becomes “[no] intersection other than $\mathbf{r}$”. The disconnectedness argument for $s < 1$ is done in the topologigal space $\mathbb{R}^{2N} \setminus E$ from which we have excluded the exceptional positions

$$E = \{\text{another ECC vertex coincides with } \mathbf{r}\}.$$

This is necessary, otherwise U_1 would not be open. (If another ECC vertex of $\mathcal{Q}$ is at point $\mathbf{r}$, an arbitrarily small movement may bring it into the interior of $\mathcal{Q}^*$.) Lemma 11 guarantees that $\mathcal{D}_s$ does not reach E for $s < 1$. □

Proof of **Proposition 17** *(page 20).* If $\mathbf{c}$ is an unrounded singleton meta-fragment consisting of one fragment $\mathbf{b}^1$, the claims are immediate from Lemma 3, since it was explicitly checked during conflict removal that $\rho\mathbf{c} = \rho\mathbf{b}^1$ is weakly bcp-monotone. If $\mathbf{c}$ consists of $\mathbf{b}^1, \ldots, \mathbf{b}^k$, $k > 1$, then each rounded fragment $\tilde{\mathbf{b}}^j$, being a subcurve of some neighbourhood curve, is bcp-monotone and thus regular and free of self-intersections at all points. It remains to argue that there is no interior self-intersection of $\rho\mathbf{c}$ arising from an intersection between two different rounded

fragments $\tilde{\mathbf{b}}^{j_1}$, $\tilde{\mathbf{b}}^{j_2}$. If $\rho\mathbf{c}$ is straight, this is immediate, so we may assume now that $\rho\mathbf{c}$ is not straight. It folllows that $\tilde{\mathbf{b}}^{j_1}$, $\tilde{\mathbf{b}}^{j_2}$ are not straight and thus, with the exception of endpoints, contained in the disjoint interiors of their enclosing polygons. But Corollary 13 excludes collapse of merged fragment endpoints as well. □

Appendix B Pseudocode

In pseudocode, we reserve bold-face letters for keywords and use variable names like b even for vector-valued quantities. Each curve is represented as a sequence $b = (b_0, \ldots, b_n)$ of control points. We write b.first for b_0 and b.last for b_n. For later use, we store the urcurve of which b is a subcurve in $ur[b]$, together with the parameter interval $(t^0[b], t^1[b])$ of $ur[b]$ to which b corresponds. The function $(b^-, b^+) :=$ subdivide(b) performs de Casteljau subdivision of b and creates entries for b^- and b^+ in $ur[]$, $t^0[]$, and $t^1[]$. For uniformity, a single point p is represented as a Bézier curve $b = (p)$ of degree 0.

B.1 The preprocessing phase

```
 1: for each b in S do
 2:     ur[b] := b;   t^0[b] := 0;   t^1[b] := 1
 3: od
 4: Q := ∅
 5: while S ≠ ∅ do
 6:     b^0 := S.pop()
 7:     if is_bcp_monotone(b^0) then
 8:         Q.add(b^0)
 9:     elsif is_trivial(b^0) then
10:         b^1 := (b^0.first);   Q.add(b^1)
11:         ur[b^1] := ur[b^0];   t^0[b^1] := t^1[b^1] := t^0[b^0]
12:     else
13:         (b^1, b^2) := subdivide(b^0)
14:         S.add(b^1);   S.add(b^2)
15:     fi
16: od
```

B.2 The graph building phase

We use a number of operations on the fat graph T. Inserting an endpoint p is done via $v := T.\text{get\_vertex}(p)$ which returns the vertex v in T located at point p. If it did not exist before, it is created. However, this can fail if point p lies on a fat edge. This has to be queried in advance with $T.\text{can\_get\_vertex}(p)$. If it returns false, then $T.\text{obstacle}(p)$ returns the fat edge containing p.

Inserting a convex polygon h as fat edge f from v to v' is done by $f := T.\text{insert\_fat\_edge}(v, v', h, b)$. Here, b is the curve enclosed by f, which can later be queried as $T.\text{curve}(f)$. Inserting a fat edge fails if it intersects any fat edge or fat vertex of T in a point other than a common endpoint. This has to be queried in advance via $T.\text{can\_insert\_fat\_edge}(v, v', h)$. If this returns false, $T.\text{obstacle}(v, v', h)$ names one of the culprits (fat edge or fat vertex). (The exact strategy is not specified.) One can also delete a fat edge f from T using $T.\text{delete}(f)$.

```
17: T := (∅, ∅)
18: while Q ≠ ∅ do
19:     R := ∅    // relegated curves
20:     b := Q.pop()
21:     if degree(b) = 0 then
22:         if ¬T.can_get_vertex(b.first) then
23:             f := T.obstacle(b.first)
24:             R.add(T.curve(f));   T.delete(f)
25:         fi
26:         v := T.get_vertex(b.first)
27:         T.preimages(v).add((ur[b], t^0[b]))
28:     else
29:         p^1 := b.first;   p^2 := b.last
30:         for i from 1 to 2 do
31:             if ¬T.can_get_vertex(p^i) then
32:                 f := T.obstacle(p^i)
33:                 R.add(T.curve(f));    T.delete(f)
34:             fi
35:             v^i := T.get_vertex(p^i)
36:             T.preimages(v^i).add((ur[b], t^(i−1)[b]))
37:         od
38:         h := conv_hull(b)
39:         if T.can_insert_fat_edge(v^1, v^2, h) then
40:             T.insert_fat_edge(v^1, v^2, h, b)
41:         else
42:             R.add(b)
```

```
43:             f := T.obstacle(v^1, v^2, h)
44:             // f is either fat edge or fat vertex
45:             if is_fat_edge(f) then
46:                 R.add(T.curve(f));   T.delete(f)
47:             fi
48:         fi
49:     fi
50:     for each c^0 in R do
51:         if ¬is_trivial(c^0) then
52:             (c^1, c^2) := subdivide(c^0)
53:             Q.add(c^1);   Q.add(c^2)
54:         fi
55:     od
56: od
```

B.3 The conflict removal phase

The conflict removal phase requires certain existential queries on vertices and edges of control cliques as detailed below. Given a fat edge f, we write $CC(f)$ for the control clique of T.curve(f). A fat edge f can be queried whether some point p is one of its endpoints or not via f.ends_at(p).

```
57: done := false
58: while ¬done do
59:     L := ∅
60:     if ∃ fat edge f in T, fat vertex v in T, edge e in CC(f):
61:                                         ¬f.ends_at(v) ∧ conflicts(e, v) then
62:         L.add(f)
63:     elsif ∃ fat edges f, f' in T, vertex p in CC(f), edge e in CC(f'):
64:                                     f ≠ f' ∧ ¬f.ends_at(p) ∧ conflicts(e, p) then
65:         L.add(f);   L.add(f')
66:     elsif ∃ non-trivial fat edge f in T:
67:                                 ¬is_weakly_bcp_monotone(ρ(T.curve(f))) then
68:         L.add(f)
69:     else
70:         done := true
71:     fi
72:     for each f in L do
73:         c^0 := T.curve(f);   T.delete(f)
74:         (c^1, c^2) := subdivide(c^0)
75:         v^1 := T.get_vertex(c^0.first)     // exists already
```

76: $v^2 := T$.get_vertex(c^2.first) // new vertex
77: T.preimages(v^2).add(($ur[c^2], t^0[c^2]$))
78: $v^3 := T$.get_vertex(c^0.last) // exists already
79: **for** i **from** 1 **to** 2 **do**
80: T.insert_fat_edge(v^i, v^{i+1}, conv_hull(c^i), c^i)
81: **od**
82: **od**
83: **od**

B.4 The merging phase

Given the fat graph T as resulting from the conflict removal phase, we perform the merging of fat edges in it by marking those fat vertices of T that are to become inner fat vertices of some chain. For this purpose, T maintains a flag T.is_merged(v) for each fat vertex v.

84: **for each** fat vertex v **in** T **do**
85: T.is_merged(v) := (#(T.preimages(v)) = 1 $\wedge$ T.degree(v) = 2) // (M1)?
86: **od**
87: **for each** fat edge f **in** T **do**
88: Let v^1, v^2 be the endpoints of f
89: **if** T.is_merged(v^1) $\vee$ T.is_merged(v^2) **then**
90: $b^* := T$.curve(f)
91: **for** i **from** 0 **to** degree(b^*) **do**
92: Compute $\mathcal{N}(b_i^*)$
93: **if** $\exists$ fat vertex $v \in \mathcal{N}(b_i^*)$: T.preimages(v) $\neq \{(ur[b^*], ...)\}$
94: $\vee$ $\exists$ fat edge $f' \in \mathcal{N}(b_i^*)$: $ur[T.\text{curve}(f')] \neq ur[b^*]$ **then**
95: T.is_merged(v^1) := false; T.is_merged(v^2) := false
96: **break for** // ends "**for** i" loop
97: **fi**
98: **od**
99: **fi**
100: **od**
101: Do DFS on T to compute the set Γ of all fat paths e of length two or more
102: in which precisely the endpoints have T.is_merged($\cdot$) = false.
103: **while** $\Gamma \neq \emptyset$ **do**
104: Remove one fat path $e = (f^1, \ldots, f^k)$ from Γ
105: Let n be the common degree of all T.curve(f^j)
106: **for** j **from** 2 **to** k **do**
107: Let v be the fat vertex joining f^{j-1} and f^j.
108: **if** v violates (M3) **then**

109: T.is_merged(v) := false
110: **if** $j > 2$ **then** $\Gamma := \Gamma \cup \{(f^1, \ldots, f^{j-1})\}$ **fi**
111: **if** $j < k$ **then** $\Gamma := \Gamma \cup \{(f^j, \ldots, f^k)\}$ **fi**
112: **continue while** // next iteration of "**while** $\Gamma \ldots$"
113: **fi**
114: **od**
115: **for** j **from** 1 **to** k **do**
116: Let fat vertices v^1, v^2 be the endpoints of f^j
117: $b^j := T$.curve(f^j)
118: **for** i **from** 1 **to** $n-1$ **do**
119: Let v be the thin vertex at point b_i^j
120: Compute I^v from $\mathcal{N}(v)$
121: **if** v violates (M3) **then**
122: T.is_merged(v^1) := false; T.is_merged(v^2) := false
123: **if** $j > 2$ **then** $\Gamma := \Gamma \cup \{(b^1, \ldots, b^{j-1})\}$ **fi**
124: **if** $j < k-1$ **then** $\Gamma := \Gamma \cup \{(b^{j+1}, \ldots, b^k)\}$ **fi**
125: **continue while** // next iteration of "**while** $\Gamma \ldots$"
126: **fi**
127: **od**
128: **od**
129: **od**

The final values of the T.is_merged($\cdot$) flags define the merged curves we are going to round. Hence $V^* = \{v \in V \mid \neg T.\text{is\_merged}(v)\}$. An unrounded meta-fragment is a curve of the form $\mathbf{c}^e$ where $e = (\mathbf{b}^1, \ldots, \mathbf{b}^k)$ is a non-trivial chain with endpoints in V^* whose inner fat vertices have T.is_merged($\cdot$) = true.

Below you find a list of the most recent technical reports of the Max-Planck-Institut für Informatik. They are available by anonymous ftp from `ftp.mpi-sb.mpg.de` under the directory `pub/papers/reports`. Most of the reports are also accessible via WWW using the URL `http://www.mpi-sb.mpg.de`. If you have any questions concerning ftp or WWW access, please contact `reports@mpi-sb.mpg.de`. Paper copies (which are not necessarily free of charge) can be ordered either by regular mail or by e-mail at the address below.

Max-Planck-Institut für Informatik
Library
attn. Anja Becker
Stuhlsatzenhausweg 85
66123 Saarbrücken
GERMANY
e-mail: `library@mpi-sb.mpg.de`

MPI-I-2006-5-006	G. Kasnec, F.M. Suchanek, G. Weikum	Yago - A Core of Semantic Knowledge
MPI-I-2006-5-005	R. Angelova, S. Siersdorfer	A Neighborhood-Based Approach for Clustering of Linked Document Collections
MPI-I-2006-5-002	H. Bast, D. Majumdar, R. Schenkel, M. Theobald, G. Weikum	IO-Top-k: Index-access Optimized Top-k Query Processing
MPI-I-2006-4-010	A. Belyaev, T. Langer, H. Seidel	Mean Value Coordinates for Arbitrary Spherical Polygons and Polyhedra in $\mathbb{R}^3$
MPI-I-2006-4-009	J. Gall, J. Potthoff, B. Rosenhahn, C. Schnoerr, H. Seidel	Interacting and Annealing Particle Filters: Mathematics and a Recipe for Applications
MPI-I-2006-4-008	I. Albrecht, M. Kipp, M. Neff, H. Seidel	Gesture Modeling and Animation by Imitation
MPI-I-2006-4-005	A. Belyaev, H. Seidel, S. Yoshizawa	Skeleton-driven Laplacian Mesh Deformations
MPI-I-2006-4-004	V. Havran, R. Herzog, H. Seidel	On Fast Construction of Spatial Hierarchies for Ray Tracing
MPI-I-2006-4-003	E. de Aguiar, R. Zayer, C. Theobalt, M. Magnor, H. Seidel	A Framework for Natural Animation of Digitized Models
MPI-I-2006-2-001	T. Wies, V. Kuncak, K. Zee, A. Podelski, M. Rinard	On Verifying Complex Properties using Symbolic Shape Analysis
MPI-I-2006-1-006	M. Kerber	Division-Free Computation of Subresultants Using Bezout Matrices
MPI-I-2005-4-006	C. Fuchs, M. Goesele, T. Chen, H. Seidel	An Emperical Model for Heterogeneous Translucent Objects
MPI-I-2005-4-003	T. Langer, A.G. Belyaev, H. Seidel	Analysis and Design of Discrete Normals and Curvatures
MPI-I-2005-4-001	M. Fuchs, V. Blanz, H. Lensch, H. Seidel	Reflectance from Images: A Model-Based Approach for Human Faces
MPI-I-2005-2-001	J. Hoffmann, C. Gomes, B. Selman	Bottleneck Behavior in CNF Formulas
MPI-I-2005-1-002	I. Katriel, M. Kutz, M. Skutella	Reachability Substitutes for Planar Digraphs
MPI-I-2005-1-001	D. Michail	Rank-Maximal through Maximum Weight Matchings
MPI-I-2004-4-004	R. Zayer, C. Rssl, H. Seidel	r-Adaptive Parameterization of Surfaces
MPI-I-2004-2-007	S. Wagner	Summaries for While Programs with Recursion
MPI-I-2004-2-001	H. de Nivelle, Y. Kazakov	Resolution Decision Procedures for the Guarded Fragment with Transitive Guards
MPI-I-2004-1-005	S. Schmitt, L. Fousse	A comparison of polynomial evaluation schemes
MPI-I-2004-1-002	P. Sanders, S. Pettie	A Simpler Linear Time 2/3 - epsilon Approximation for Maximum Weight Matching
MPI-I-2003-NWG2-002	F. Eisenbrand	Fast integer programming in fixed dimension
MPI-I-2003-4-005	T. Hangelbroek, G. Nrnberger, C. Roessl, H.S. Seidel, F. Zeilfelder	The Dimension of C^1 Splines of Arbitrary Degree on a Tetrahedral Partition
MPI-I-2003-4-003	R. Zayer, C. Roessl, H. Seidel	Convex Boundary Angle Based Flattening
MPI-I-2003-4-001	M. Tarini, H.P.A. Lensch, M. Goesele, H. Seidel	3D Acquisition of Mirroring Objects

MPI-I-2003-2-004	A. Podelski, A. Rybalchenko	Software Model Checking of Liveness Properties via Transition Invariants
MPI-I-2003-2-001	P. Maier	Compositional Circular Assume-Guarantee Rules Cannot Be Sound And Complete
MPI-I-2003-1-018	G. Schaefer	A Note on the Smoothed Complexity of the Single-Source Shortest Path Problem
MPI-I-2003-1-014	G. Schfer, L. Becchetti, S. Leonardi, A. Marchetti-Spaccamela, T. Vredeveld	Average Case and Smoothed Competitive Analysis of the Multi-Level Feedback Algorithm
MPI-I-2003-1-011	P. Krysta, A. Czumaj, B. Voecking	Selfish Traffic Allocation for Server Farms
MPI-I-2003-1-010	H. Tamaki	A linear time heuristic for the branch-decomposition of planar graphs
MPI-I-2003-1-009	B. Csaba	On the Bollobás – Eldridge conjecture for bipartite graphs
MPI-I-2003-1-007	H. Tamaki	Alternating cycles contribution: a strategy of tour-merging for the traveling salesman problem
MPI-I-2003-1-005	M. Dietzfelbinger, P. Woelfel	Almost Random Graphs with Simple Hash Functions
MPI-I-2003-1-004	E. Althaus, T. Polzin, S.V. Daneshmand	Improving Linear Programming Approaches for the Steiner Tree Problem
MPI-I-2003-1-003	R. Beier, B. Vcking	Random Knapsack in Expected Polynomial Time
MPI-I-2003-1-002	P. Krysta, P. Sanders, B. Vcking	Scheduling and Traffic Allocation for Tasks with Bounded Splittability
MPI-I-2003-1-001	P. Sanders, R. Dementiev	Asynchronous Parallel Disk Sorting
MPI-I-2002-4-002	F. Drago, W. Martens, K. Myszkowski, H. Seidel	Perceptual Evaluation of Tone Mapping Operators with Regard to Similarity and Preference
MPI-I-2002-4-001	M. Goesele, J. Kautz, J. Lang, H.P.A. Lensch, H. Seidel	Tutorial Notes ACM SM 02 A Framework for the Acquisition, Processing and Interactive Display of High Quality 3D Models
MPI-I-2002-2-008	W. Charatonik, J. Talbot	Atomic Set Constraints with Projection
MPI-I-2002-2-007	W. Charatonik, H. Ganzinger	Symposium on the Effectiveness of Logic in Computer Science in Honour of Moshe Vardi
MPI-I-2002-1-008	P. Sanders, J.L. Trff	The Factor Algorithm for All-to-all Communication on Clusters of SMP Nodes
MPI-I-2002-1-005	M. Hoefer	Performance of heuristic and approximation algorithms for the uncapacitated facility location problem
MPI-I-2002-1-004	S. Hert, T. Polzin, L. Kettner, G. Schfer	Exp Lab A Tool Set for Computational Experiments
MPI-I-2002-1-003	I. Katriel, P. Sanders, J.L. Trff	A Practical Minimum Scanning Tree Algorithm Using the Cycle Property
MPI-I-2002-1-001	T. Polzin, S. Vahdati	Using (sub)graphs of small width for solving the Steiner problem
MPI-I-2001-4-005	H.P.A. Lensch, M. Goesele, H. Seidel	A Framework for the Acquisition, Processing and Interactive Display of High Quality 3D Models
MPI-I-2001-4-004	S.W. Choi, H. Seidel	Linear One-sided Stability of MAT for Weakly Injective Domain
MPI-I-2001-4-003	K. Daubert, W. Heidrich, J. Kautz, J. Dischler, H. Seidel	Efficient Light Transport Using Precomputed Visibility
MPI-I-2001-4-002	H.P.A. Lensch, J. Kautz, M. Goesele, H. Seidel	A Framework for the Acquisition, Processing, Transmission, and Interactive Display of High Quality 3D Models on the Web
MPI-I-2001-4-001	H.P.A. Lensch, J. Kautz, M. Goesele, W. Heidrich, H. Seidel	Image-Based Reconstruction of Spatially Varying Materials
MPI-I-2001-2-006	H. de Nivelle, S. Schulz	Proceeding of the Second International Workshop of the Implementation of Logics
MPI-I-2001-2-005	V. Sofronie-Stokkermans	Resolution-based decision procedures for the universal theory of some classes of distributive lattices with operators