Complete, Exact, and Efficient Computations with Cubic Curves*

Arno Eigenwillig
Max-Planck-Institut für Informatik
66123 Saarbrücken, Germany
arno@mpi-sb.mpg.de

Lutz Kettner
Max-Planck-Institut für Informatik
66123 Saarbrücken, Germany
kettner@mpi-sb.mpg.de

Elmar Schömer
Johannes-Gutenberg-Universität
55099 Mainz, Germany
schoemer@uni-mainz.de

Nicola Wolpert
Max-Planck-Institut für Informatik
66123 Saarbrücken, Germany
nicola@mpi-sb.mpg.de

ABSTRACT

The Bentley-Ottmann sweep-line method can be used to compute the arrangement of planar curves provided a number of geometric primitives operating on the curves are available. We discuss the mathematics of the primitives for planar algebraic curves of degree three or less and derive efficient realizations. As a result, we obtain a *complete*, *exact*, and *efficient* algorithm for computing arrangements of cubic curves. Conics and cubic splines are special cases of cubic curves.

The algorithm is *complete* in that it handles all possible degeneracies including singularities. It is *exact* in that it provides the mathematically correct result. It is *efficient* in that it can handle hundreds of curves with a quarter million of segments in the final arrangement.

Categories and Subject Descriptors

I.3.5 [**Computer Graphics**]: Computational Geometry and Object Modeling—*geometric algorithms*; G.1.5 [**Numerical Analysis**]: Roots of Nonlinear Equations—*methods for polynomials*; D.m [**Software**]: Miscellaneous—*robust geometric computation*

General Terms

Algorithms, Performance, Reliability

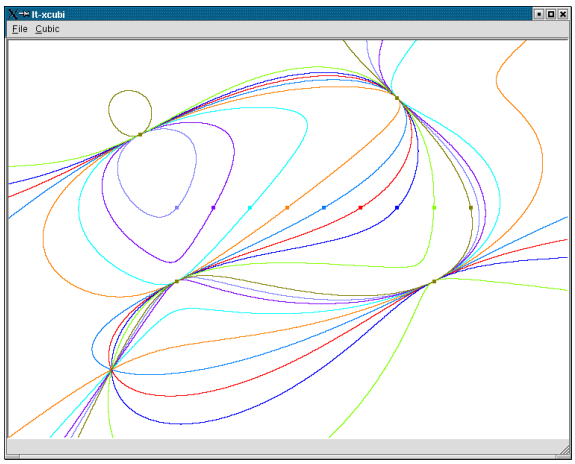
*Partially supported by the IST Programme of the European Union as a Shared-cost RTD (FET Open) Project under Contract No IST-2000-26473 (ECG – Effective Computational Geometry for Curves and Surfaces)

Keywords

Computational geometry, arrangements, algebraic curves, sweep-line algorithm, robustness, exact geometric computation

1. INTRODUCTION

The Bentley-Ottmann sweep-line method [3] can be used to compute the arrangement of planar curves. One *only* has to provide a number of geometric primitives (break the curve into x-monotone pieces, given two curves compute their intersections, compare two intersections or endpoints lexicographically, etc.). The "only" is the crux of the matter. In principle, the problem is simple, since the first-order theory of real closed fields is decidable by Cylindrical Algebraic Decomposition [9]. An efficient realization is another matter.



We discuss the mathematics of the primitives for cubic curves, i.e. planar algebraic curves of degree three (or less) and derive efficient realizations. Conics and cubic splines are special cases of cubic curves. We obtain a *complete* (it handles all possible degeneracies), *exact* (it provides the mathematically correct result), and *efficient* (it can handle hundreds of curves with a quarter million of resulting segments) sweep-line algorithm for computing planar arrangements of cubic curves.

Complete, exact, and efficient implementations for the linear case exist, e.g., in LEDA [24, ch. 10.7], and in the planar map [17] and Nef-polyhedron [28] classes of CGAL. However, existing implementations for curved objects are either incomplete, inexact, or not aimed at efficiency except for some recent work on circle and conic arcs (see below). For cubic curves we are not aware of any complete, exact, and efficient implementation.

With only minor modifications our geometric primitives can also be used to realize the randomized incremental approach. Our implementation can be easily extended to compute arrangements of cubic segments and to perform regularized boolean operations on polygons bounded by them.

2. PREVIOUS WORK

Our work is influenced by the work of three different communities: Computer Aided Design (CAD), computational geometry, and computer algebra. The problem of computing intersections of curves and more generally surfaces has a long history in CAD. The CAD community concentrated on approximate solutions by numerical methods. Exact solutions were never an issue, for example, it was never the goal to distinguish between a tangential intersection and two intersections lying very close together. Complete and exact implementations have been addressed only recently. MAPC [23] is a library for exact computation and manipulation of algebraic curves. It offers arrangements of planar curves but does not handle all degenerate situations.

Arrangements, mostly of linear objects, are also a major focus in computational geometry; see the survey articles of Halperin [21] and Agarwal/Sharir [2]. Many exact methods for curved objects, e.g. [1], have been formulated for the Real RAM model of computation [26], which allows unit-time operations on arbitrary algebraic numbers and conceals the high cost of exact arithmetic with algebraic numbers.

A more realistic view was taken by Sakkalis [27] and Hong [22]. Both analyze the topology of a single real algebraic curve. They do not consider the interaction between pairs of curves and the full algebraic machinery they develop is not necessary for cubic curves. Aspects of the crucial problem to capture behavior at irrational points by rational arithmetic were treated by Canny [8] (Gap Theorem) and Pedersen [25] (multivariate Sturm sequences). The running time of both methods is quite high.

Predicates for arrangements of circular arcs that reduce all computations to sign determination of polynomial expressions in the input data are treated by Devillers et al. [13]. Recent work by Emiris and Tsigardias [16] discusses some predicates on conics in this style; see also [15] in this volume. However, these approaches do not extend easily to more complicated curves.

Exact, efficient, and complete algorithms for planar arrangements have been published by Wein [30] and Berberich et al. [4] for conic segments, and by Wolpert [31] (see also [19]) for special quartic curves as part of a surface intersection algorithm. A generalization of Jacobi curves (used below for locating tangential intersections) is described by Wolpert [32].

3. OUR RESULTS

What are the difficulties in going from straight lines and straight line segments to conics and further on to cubic curves? The main distinction is the field of coordinates. For straight line segments with rational endpoints all vertices have rational coordinates. In the case of higher degree curves, the coordinates are, in general, irrational algebraic numbers.

The field of real root expressions (FRE) is the closure of the integers under the operations $+$, $-$, $*$, $/$, and $\sqrt[k]{\ }$ for arbitrary but fixed k; it is a subfield of the real algebraic numbers. Reasonably efficient methods [11, 6] are available to compute in FRE and to compare expressions in FRE. Since the $\sqrt[k]{\ }$ operation is restricted to real roots, general polynomial equations of degree ≥ 3 are not solvable in FRE.

The sweep-line algorithm works on x-monotone segments. Lines and line segments are x-monotone, conics need to be split at points of vertical tangent, and cubic curves need to be split at points of vertical tangent and at singularities. These terms are defined formally in Section 4. Computing and analyzing singularities is a challenging problem that first becomes relevant for algebraic curves of degree three. The coordinates of the split points are rational numbers in the case of line segments, are in FRE for conics, and are outside FRE for cubics. Thus comparisons between endpoints are more complex.

The x-monotone segments have parameterizations $y(x)$, according to the Implicit Function Theorem. In the case of line segments, y is a linear function of x, and for conics, $y(x)$ can be expressed using one square root. This allows for a simple comparison of the y-coordinates of different arcs within FRE. The x-monotone segments of cubics have *no* parameterization as functions of x within FRE and hence much of the machinery developed for conics does not carry over. Section 5 on choosing a generic coordinate system prepares the stage for performing the analysis of one cubic curve, which is the subject of Section 6.

We turn to pairs of curves. Two lines intersect in a single point with rational coordinates. Two conics intersect in up to four points. In the case of a tangential intersection, the x-coordinate of the intersection belongs to FRE and can even be written as an expression involving a single root. In other words, only the coordinates of transversal intersections are outside FRE. No such simplification holds for cubic curves and hence we need to work a lot harder. We deal with the analysis of a pair of curves in Section 7.

In Section 8 we put everything together and discuss high-level issues of the Bentley-Ottmann sweep as applied to cubic curves.

We conclude with a discussion of running time from a theoretical (Section 9) and especially an experimental (Section 10) point of view. There is a full implementation of our algorithm. For a detailed treatment, see [14].

In our exposition, we focus on curves of degree exactly 3. The extension to lines and conics is straightforward.

4. TERMINOLOGY

Let $f \in \mathbb{Q}[x,y]$ and $\textsc{Zero}(f) := \{(\alpha,\beta) \in \mathbb{R}^2 \mid f(\alpha,\beta) = 0\}$. The set of points $\textsc{Zero}(f)$ is called the *algebraic curve* defined by f. If the context is unambiguous, we will often identify the defining polynomial of an algebraic curve with its zero set. A *cubic curve* (or *cubic* for short) is a curve of degree 3 (or less). An algebraic curve can always be defined by a *square free* polynomial $f \in \mathbb{Q}[x,y]$, meaning that no non-constant square divides f. A polynomial $f = f_n(x) \cdot y^n + f_{n-1}(x) \cdot y^{n-1} + \cdots + f_0(x) \in \mathbb{Q}[x,y]$ is called *y-regular*

if $f_n(x)$ is a non-zero constant. This is a sufficient criterion for the absence of vertical asymptotes.

The *gradient vector* of an algebraic curve f is defined to be $\nabla f := (f_x, f_y)$, where subscripts denote partial derivatives. A point (α, β) of f is called *singular* if $(\nabla f)(\alpha, \beta) = (f_x(\alpha, \beta), f_y(\alpha, \beta)) = (0, 0)$, otherwise it is *non-singular*. At singular points, several arcs of one curve intersect. Beware that some or all of them may be complex and puncture the real plane only at this point.

We call a non-singular point $(\alpha, \beta) \in \mathbb{R}^2$ of f *vertical* if $f_y(\alpha, \beta) = 0$. Vertical points have a vertical tangent. A non-singular point $(\alpha, \beta) \in \mathbb{R}^2$ of f is named a *flex* if the curvature of f becomes zero in (α, β): $0 = (f_{xx} f_y^2 - 2 f_x f_y f_{xy} + f_{yy} f_x^2)(\alpha, \beta)$. A vertical point that is not a flex is called *extreme*, since its x-coordinate is minimal or maximal among the neighboring points on the curve.

Two curves f and g are *coprime* if they do not have a common non-constant factor. W.l.o.g., we assume that this is the case for every pair of curves f and g we consider. Coprimality can be tested and established by a bivariate gcd computation.

A point (α, β) is called an *intersection point* of f and g if it lies on f as well as on g. It is called a *tangential intersection point* of f and g if additionally the two gradient vectors are linearly dependent: $(f_x g_y - f_y g_x)(\alpha, \beta) = 0$. Otherwise one speaks of a *transversal intersection point*. Two intersection points $(\alpha_1, \beta_1) \neq (\alpha_2, \beta_2)$ are *covertical* if $\alpha_1 = \alpha_2$.

5. COORDINATE SYSTEM CONDITIONS

Some of the properties discussed above are inherent in the geometry of a curve and remain invariant under changes of coordinate systems (e.g. a point being a flex or a singularity), whereas others (such as verticality of a tangent) come from a specific choice of coordinates.

To simplify the analysis of curves and curve pairs, we impose conditions on the coordinate system that exclude certain degenerate choices. We check that the conditions are satisfied, and if not, we change coordinates suitably by shearing. Almost all choices of a coordinate system (a *generic* coordinate system) will do, as discussed in the Appendix. The algorithm remains complete, since there is no restriction of the geometric situations it can handle.

The conditions imposed on each curve f are these:

- The curve f is y-regular.
- There are no vertical flexes on f.
- All tangents at singularities are non-vertical.

The following conditions are imposed on every curve pair $\{f, g\}$ which is analyzed:

- No two (complex) intersection points of f and g are covertical.
- No two real intersections or real extreme and singular points of f and g are covertical.
- No intersection point is extreme on either curve.
- The curve $f_x g_y - f_y g_x$ is y-regular, and none of its extreme and singular points is covertical or equal to a non-singular intersection of f and g with multiplicity 2.
- If f is a product of three lines such that two of them are complex and intersect in a non-singular point of g, then the intersection of f_y and g in this point is transversal.

The meaning of the last two conditions will become clear in Section 7.

Note that the position conditions come from the analysis of curves, not from the carefully formulated version of the sweep-line algorithm we use [24, ch. 10.7]. Covertical intersection points are no problem for our method as long as they do not involve the same two curves. As far as the conditions relate to pairs of curves, they only affect pairs that interact geometrically and those needed while searching the Y-structure of the sweep algorithm.

W.l.o.g., we also require the curve's defining polynomial to be square free. If not, assuming f to be y-regular, we obtain the square free part of f as $f / \gcd(f, f_y)$. Remember that we also demanded coprimality for all pairs of input curves.

6. ANALYZING ONE CURVE

To perform the analysis of one curve f, we demand the underlying coordinate system to be generic with respect to f as described in Section 5.

With the goal of sweeping in mind, we wish to investigate the *behavior* of a single cubic curve f at any given x-coordinate x_0, meaning the number and relative position of its arcs along the line $x = x_0$. Algebraically this means that we are interested in the number and order of the real roots of $f(x_0, y) \in \mathbb{R}[y]$. Using the Implicit Function Theorem, one obtains that the arcs of f evolve smoothly as we vary x_0, except for the intersection points of f and f_y. Their x-coordinates are called the *critical points* of f. Since f and f_y are coprime algebraic curves, they intersect in a finite number of points (at most 6=deg(f)(deg(f) − 1) by Bézout's Theorem). In summary we obtain:

LEMMA 1. *Consider the behavior of a y-regular curve $f \in \mathbb{Q}[x, y]$ as x varies over $\mathbb{R}$. There is a finite number of critical points and of open intervals with constant behavior between them. Critical points can be found by intersecting f and f_y.*

In the absence of vertical flexes the critical points of f are the x-coordinates of extreme (Figure 1 (i)) and singular (Figure 1 (ii)) points. Collectively, we call them *one-curve event points*.

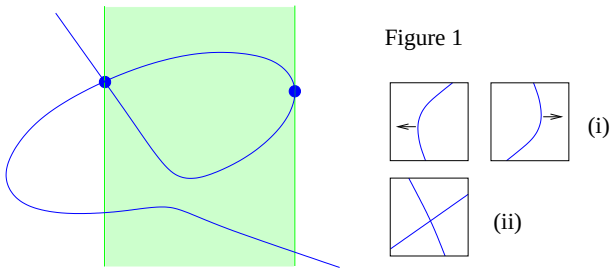


Figure 1

For degree reasons, a cubic curve f can never have two covertical one-curve event points:

LEMMA 2. *Let f be a y-regular cubic curve. No two intersections of f and f_y are covertical.*

PROOF. Any intersection point (α, β) is a double root of $f(\alpha, y) \in \mathbb{R}[y]$ which has degree < 4. □

To obtain the x-coordinates of the intersection points of f and f_y we eliminate the variable y using a well-known algebraic tool: the resultant. Given two curves $f, g \in \mathbb{Q}[x, y]$,

the *resultant* $\mathrm{res}(f,g,y) \in \mathbb{Q}[x]$ of f and g with respect to variable y has the following characteristic property [12, §3.5]:

PROPOSITION 3. *Let $f, g \in \mathbb{Q}[x,y]$ be y-regular curves. A number $\alpha \in \mathbb{C}$ is a root of $r := \mathrm{res}(f,g,y)$ if and only if there exists $\beta \in \mathbb{C}$ such that $f(\alpha,\beta) = g(\alpha,\beta) = 0$.*

COROLLARY 4. *If f and g have no covertical intersections in $\mathbb{C}^2$, then the correspondence between intersection points and roots of r is bijective, every real root of r corresponds to a real intersection point, and the multiplicity of a root is the multiplicity of the corresponding intersection.*

COROLLARY 5. *The critical points of a y-regular cubic curve f are precisely the real roots of* $\mathrm{res}(f, f_y, y)$.

We factor this resultant by multiplicities, i.e. $\mathrm{res}(f,f_y,y) = \prod_{i=1}^{k} r_i^i$ such that the $r_i \in \mathbb{Q}[x]$ are square free and pairwise coprime [18, ch. 8]. The roots of r_1 are the x-coordinates of extreme points, the roots of r_i for $i > 1$ those of singular points. By definition, every root α of a univariate polynomial $p(x) \in \mathbb{Q}[x]$ is an *algebraic number*. For $\deg(p) \geq 3$ there is no general way to express α within FRE. But we can determine an *isolating interval* for each real root α of p using Uspensky's method [10]. This yields rational bounds $a < \alpha < b$ such that α is the unique real root of p in $[a,b]$, and we use $(p,[a,b])$ to represent α. Two numbers of this form are compared along the following lines: Equality can be decided using the gcd of the defining polynomials. Unequal numbers can be compared by refining their intervals to disjointness.

Given another polynomial $q \in \mathbb{Q}[x]$, similar techniques allow us to refine the isolating interval of $(p,[a,b])$ such that it does not contain any root of q different from α.

To analyze the behavior of f at critical points, we need to extend every x-coordinate back to the event point by determining which arcs of f are involved. Let us begin with extreme points.

THEOREM 6. *Consider a locally x-minimal extreme point (α,β) of a y-regular cubic curve f with positive leading coefficient f_3. Let $a < \alpha$ be a rational number such that no zero of* $\mathrm{res}(f,f_y,y)$ *or* $\mathrm{res}(f_y,f_{yy},y)$ *lies in $[a,\alpha)$. Let $c \in \mathbb{Q}$ be the solution of the linear equation $f_{yy}(a,y) = 0$. Then (α,β) lies below/above the uninvolved arc of f if $f(a,c)$ is negative/positive, respectively.*

The case of a locally x-maximal point is symmetric.

PROOF. The univariate polynomial $f(\alpha,y) \in \mathbb{R}[y]$ has its root β in common with its first derivative $f_y(\alpha,y)$, but not with its second derivative $f_{yy}(\alpha,y)$ (because the curvature at (α,β) is non-zero). It follows that the roots of $f(\alpha,y)$ are β (twofold) and a third root $\beta' \in \mathbb{R}$ (simple). It follows further that $f_y(\alpha,y)$ has precisely two distinct simple real roots. One of them is β, and the other one, say γ, lies between β and β' by the Mean Value Theorem.

By choice of a, there is no change in behavior of f and f_y between a and α. Therefore, both roots of the quadratic polynomial $f_y(a,y) \in \mathbb{Q}[y]$ and thus also their midpoint c all lie below or above the unique zero of $f(a,y) \in \mathbb{Q}[y]$, and this reflects the relative position of arcs at $x = \alpha$. □

The x-coordinate α of an extreme point is represented by $(r_1,[a,b])$. To apply the proposition, it may be necessary to refine the interval $[a,b]$.

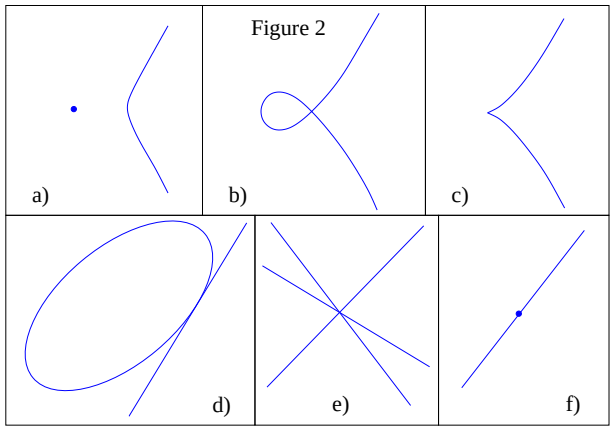
Now we turn to singularities. As before, we have to determine the arcs involved in each singularity, and we want to determine the type of the singularity (see below). The x-coordinates of the singularities are the roots of the resultant factors r_i for $i > 1$. The number of singular points bounds the degree of the polynomial defining their x-coordinate; in particular, a unique singularity is rational. The textbook classification of cubic curves [20] tells us that more than one singularity can arise only if f is a product of several *components* (non-constant factors) whose intersections then are the singularities of f.

PROPOSITION 7. *Let $f \in \mathbb{Q}[x,y]$ be a square free cubic curve with exactly s singular points in $\mathbb{C}^2$. Then $0 \leq s \leq 3$.*
If $s = 1$, then the unique singularity of f is rational.
If $s = 2$, then f is a product $f = gh$ of a line g and a conic h intersecting in two distinct points.
If $s = 3$, then $f = g_1 g_2 g_3$ is the product of three lines intersecting in three distinct points.

Given the x-coordinate α of a singularity (α,β), its y-coordinate β and the y-coordinate β' of the uninvolved arc (if any) can be obtained by factoring $f(\alpha,y)$ by multiplicities. If β' exists, we have to compare it to β to determine the involved arcs.

For rational singularities, arithmetic with α is no problem, and we can determine the type of singularity by inspecting the quadratic part $ay^2 + bxy + cx^2$ of the translated polynomial $f(x+\alpha, y+\beta)$. See Figure 2 for various types of rational singularities: A rational singularity can either be a) an acnode, or b) a crunode, or c) a cusp, or d) a tacnode, or e) a real triple point, or f) a complex triple point.

Figure 2



If a singularity is not known to be rational ($s > 1$), it is an acnode or crunode, which are easily distinguished, but to compare β and β' we resort to exact arithmetic over the extension field $\mathbb{Q}(\alpha)$. For $s = 2$, this requires to adjoin one square root, and the comparison of β and β' amounts to a comparison within FRE. If $s = 3$, we are in the very special case of three lines forming a triangle (which may have zero or two complex vertices). Conceptually, we have to compute and compare in $\mathbb{Q}(\alpha) \simeq \mathbb{Q}[x]/(r_2)$, and this is in general outside FRE. However, the necessary computation reduces to four bivariate polynomial divisions and one extended univariate polynomial gcd computation over the rationals.

Having performed the analysis of one curve, we know the number of points on it over any given x. We know where extreme points and singularities lie, which arcs they

involve, and (knowing the types of singularities) how arcs run through them.

7. ANALYZING TWO CURVES

We now turn to the behavior of a pair $\{f, g\}$ of coprime curves, subject to the conditions of Section 5. For all $x_0 \in \mathbb{R}$, we want to compute a *slice* of the pair, that is the sequence of intersections of f and g along the line $x = x_0$. Slices will form the basis of implementing the predicates needed by the sweep algorithm. Again, we have open intervals on the x-axis of constant behavior, and *critical points* on the x-axis where the sequence changes. The points in the plane that cause such a critical point are called *(two-curve) event points.*

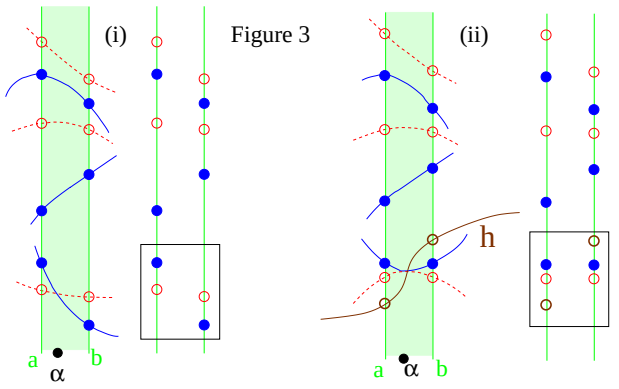
LEMMA 8. *Two curves f and g have finitely many two-curve event points. These are the one-curve event points and intersections of f and g.*

PROOF. $\text{ZERO}(f) \cup \text{ZERO}(g) = \text{ZERO}(fg)$ and $(fg)_y = f_y g + f g_y$. Thus the set of two-curve event points is given by $fg \cap (fg)_y = (f \cap f_y) \cup (g \cap g_y) \cup (f \cap g)$. □

For $x = x_0$ from an interval between two adjacent critical points, the slice is determined by substituting a rational x into f and g, solving for y by root isolation, and sorting the results. We also need to slice at critical points. Using the analysis of a single curve, one can extend slicing to x-coordinates at which just a one-curve event happens. The rest of this section describes how to slice at intersections.

We use a resultant $\text{res}(f, g, y)$ to project the intersection points onto their x-coordinates. Under the noncoverticality condition of Section 5, the multiplicity of a root α is the multiplicity of the corresponding intersection (α, β). Outside one-curve events, this reflects the degree up to which the Taylor expansions of the implicit functions of f and g around (α, β) agree.

We need to find out the respective arcs of f and g intersecting at a given zero α of $\text{res}(f, g, y)$. First consider the case that no singularity is involved in the intersection. That means exactly one arc of f intersects one arc of g. If the intersection multiplicity is odd the intersecting arcs can be determined the following way (see also Figure 3 (i)) [32]:



THEOREM 9. *Let f, g be two cubic curves in a generic coordinate system and let $(\alpha, \beta) \in \mathbb{R}^2$ be an intersection point of odd multiplicity. Let $a < \alpha < b$ be two rational numbers such that $[a, b]$ contains no root of $\text{res}(f, f_y, y)$, $\text{res}(g, g_y, y)$ and no root of $\text{res}(f, g, y)$ other than α. Then the intersecting arcs are precisely those whose order is transposed on the two vertical lines $x = a$ and $x = b$.*

PROOF. The odd intersection multiplicity entails the transposition of arcs locally around $x = \alpha$, and the conditions on a, b guarantee that no other event occurs over $[a, b]$. □

This technique can be seen as a simplified variant of box hit counting [23] [31].

If the intersection multiplicity is 2, there is no transposition of the arcs, so this technique fails. But we can consider an auxiliary curve of degree 4, the *Jacobi curve* $h = f_x g_y - f_y g_x$ of f and g. Wolpert [31] [32] shows:

THEOREM 10. *If two algebraic curves f and g have an intersection point of multiplicity 2, then h intersects f and g transversally in this point.*

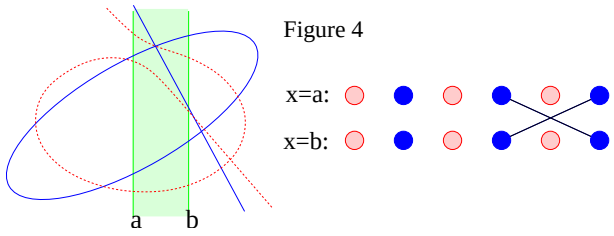
This reduces the analysis of non-singular intersections of multiplicity 2 to the previous method for transversal intersections (see Figure 3 (ii)) and motivates our condition from Section 5 on the absence of one-curve events of h which are covertical to twofold intersections.

For intersection points with even multiplicities > 2 we can employ exact arithmetic:

PROPOSITION 11. *Critical points of two cubic curves f and g corresponding to intersections of even multiplicity > 2 involve at most one square root.*

PROOF. We have $\deg(\text{res}(f, g, y)) = \deg(f) \cdot \deg(g) \leq 9$ so that factoring $\text{res}(f, g, y)$ by multiplicities produces a factor of degree at most 2 for multiplicity 4 or of degree at most 1 for multiplicities 6 and 8. □

Now consider the case that f intersects g at an x-coordinate $x = \alpha$ at which f has also a singularity but g does not. Under the conditions from Section 5, the intersection has to occur in the singularity of f, and it remains to detect the involved arc of g.



PROPOSITION 12. *Given two cubic curves f, g in a generic coordinate system such that the point (α, β) is singular on f and non-singular on g, we can detect the arc of g involved in this intersection by inspecting one pair $(f(x_0, y), g(x_0, y))$ or $(f_y(x_0, y), g(x_0, y))$ of univariate polynomials for no more than two rational values of x_0.*

PROOF. Recall our classification of singularities from Section 6. If α is known rational, we simply rely on exact arithmetic at $x_0 = \alpha$. Otherwise we know from the analysis of one curve whether we have a crunode or acnode. In case of a crunode, we will have a transposition due to the fact that g cannot be tangential to both arcs of f, see Figure 4. In case of an acnode, we additionally consider the curve f_y. We know which arc of f_y contains the singularity. This reduces the problem to intersecting two arcs, one of g and one of f_y, and the conditions on the coordinate system guarantee transversality. □

The final case of an intersection at an x-coordinate α where both curves are singular can only occur for rational α, due to the genericity and coprimality conditions.

PROPOSITION 13. *Let f, g be two coprime cubic curves in a generic coordinate system. Let (α, β) be a singular point of both f and g. Then α is known to be rational.*

PROOF. By genericity, there are no extreme points equal or covertical to intersections. Thus any root of the polynomial $d := \gcd(\mathrm{res}(f, f_y, y), \mathrm{res}(g, g_y, y), \mathrm{res}(f, g, y))$ corresponds to an intersection point which is singular on both f and g. If the square free part of d had degree > 1, then f and g would have a common component by Proposition 7, viz. the line joining the two common singular points, contradicting coprimality. □

8. PREDICATES

We show how to perform a Bentley-Ottmann-like sweep [3] of cubic curves, in the complete formulation from LEDA [24, ch. 10.7], based on the analysis of one and two curves. As input, we accept a set of cubic curves. As output, our method computes a planar map labelled with points (including auxiliary points like extreme points) and input curves, representing the arrangement.

In a preprocessing step, every input curve f is broken into sweepable *segments* such that each segment s has no one-curve events in its interior and such that all points in the interior of s have the same *arc number* i, meaning that β is the i-th real root of $f(\alpha, y)$ for every $(\alpha, \beta) \in \mathrm{int}(s)$.

A segment is represented by its endpoints, its supporting curve, and its respective arc numbers in the interior and at the endpoints. A point is represented by an x-coordinate, a supporting curve and an arc number.

Here is a summary of the sweep-line algorithm, indicating the realization of the necessary predicates:

- Extract next event from the X-structure and advance the sweep line to it. Find segments involved in the event by locating the event in the Y-structure, exploiting its order.
  Necessary predicate: Is a point p above, on, or below a given segment s? Decide this using the slice of the two supporting curves at the x-coordinate of p.
- Reorder the k intersecting segments according to intersection multiplicity in time $O(k)$, see [4].
  Necessary predicate: intersection multiplicity. This is the multiplicity of the corresponding root of $\mathrm{res}(f, g, y)$.
- Add starting segments to the Y-structure, obeying its ordering.
  Necessary predicate: comparison of segments right of a common point. Inspect the slice of the two supporting curves over the interval right of the intersection.
- Add intersections of newly adjacent segments to X-structure, obeying its ordering.
  Necessary construction: intersection of two segments; by analyzing the supporting curves.
  Necessary predicate: (x, y)-lexicographic comparison of points. If comparing x-coordinates does not break the tie, slice the two supporting curves at their common x-coordinate to compare the supporting arcs.

This use of predicates and the treatment of x-coordinates as objects in their own right reduces all geometric analyses to at most two curves at a time.

Running the algorithm and evaluating the predicates may necessitate to shear the whole scene (see Appendix). Once the algorithm terminates, the edges of the planar map are labelled with the original segments. This is no problem. However, the vertices are labelled with an implicit representation of points which is meaningless after shearing back because the notion of an i-th point on a curve at a given x is destroyed by shearing. Hence there is a post-processing step in which the point coordinates are computed and sheared back numerically within a user-defined error bound. As this happens only afterwards, the topology of the arrangement is always correct.

9. RUNTIME ANALYSIS

The runtime of the sweep in the Real RAM model remains the known $O((n + s)\log(n + m) + m)$ [24, ch. 10.7], where n is the number of curves, s the number of nodes, and m the number of edges in the resulting planar map. Here it is essential that we reorder all segments at an intersection point in linear time.

We consider the effect of shearing on the runtime. In the Appendix we show that we have only a constant number of forbidden directions for each analyzed pair of curves. We conclude that a random choice among quadratically many directions will lead to an expected constant number of shears. Since bit-complexity is of interest, we actually bias the choice towards directions of small bit size representations.

Besides the Real RAM model, the bit-complexity is of obvious importance here. However, a complete worst-case analysis is impractical, see the number of case distinctions, and furthermore, we expect no promising result from the known separations bounds that we would need to apply for the (cascaded) root isolations [7].

Instead, we emphasize that our approach is not tied to the worst case. Our methods benefit whenever a particular instance does not require the isolating intervals to approach the separation bounds limit but can stop earlier. Not only does the iteration stop earlier, e.g., in the Uspensky method, but also the bit complexity of the interval boundaries becomes smaller and subsequent steps are faster. Even more important, we do not use the separation bound approach to detect equality between our algebraic numbers. This is in contrast, e.g., to the LEDA real implementation, where the equality test is the most costly decision. Instead we detect equality of two algebraic numbers by finding a common factor of their defining polynomials with a root in the appropriate interval. This is much faster than refining the intervals to their separation bounds.

On top of this, we use modular arithmetic to quickly filter out gcd computations in cases of coprime defining polynomials. (We check whether their Bézout matrix has a vanishing determinant modulo a fixed-size prime.) The same modular filter helps to speed up factorization by multiplicities, because squarefreeness of a univariate polynomial p is equivalent to coprimality of p and p'.

In summary, we argue that the worst-case analysis of the bit complexity would be non-representative for our approach. Instead, we illustrate its effectiveness with the experiments in the next section.

An alternative is the randomized incremental construction because of its better asymptotic runtime. We can realize the necessary predicates with our approach. Since we do not simply determine signs of polynomial expressions, it is not

clear whether the lower degree of predicates for the linear case carries over to our setting.

10. EXPERIMENTS

We offer three series of benchmarks. Firstly, there is a series of *random* sets of n cubic curves. Each curve f is defined by interpolation through 9 points chosen uniformly at random from a set of $9n$ random points on the $\{-128,\ldots,127\}^2$ integer grid. Every interpolation point results in a homogeneous linear condition on the 10 unknown coefficients of f, so that generically 9 conditions determine the equation of a curve uniquely, up to a constant factor. For each input size n, we have generated an odd number of candidate input data sets and picked the one with median average running time for inclusion in our benchmark.

Secondly, there is a series of *degenerate* instances. It is obtained in a similar fashion, except that
1) There are only 54 interpolation points.
2) At the first interpolation point of each curve, we demand with probability 0.2 that not only f but also f_x and f_y vanish, making this point a rational singularity (yielding 2 additional linear conditions).
3) For each interpolation point p, we pick random values for slope m_p and curvature κ_p. Whenever a curve f is interpolated through p, we make its slope equal to m_p (yielding one additional linear condition) and, with probability $\sqrt{0.5}$, we also make its curvature equal to κ_p (yielding a further condition).

The result is an arrangement of curves with 1) vertices of high degree, 2) curves with singularities, 3) two- and three-fold intersections.

Thirdly, there is a series with *coefficient growth*: We take the $n = 60$ instances from the preceding *random* and *degenerate* series, scale each interpolation point p by a factor $s = 100$, 10 000, or 1 000 000, and then perturb it to $sp + \varepsilon$ with offset vector ε chosen randomly from $\{-10,\ldots,10\}^2$. (All occurrences of an interpolation point are mapped in the same way such as to preserve degeneracies.) This increases the bit size of the curves' coefficients but preserves the combinatorial structure of the arrangement (if s is sufficiently large), allowing us to measure the increased cost of arithmetic as a function of bit sizes.

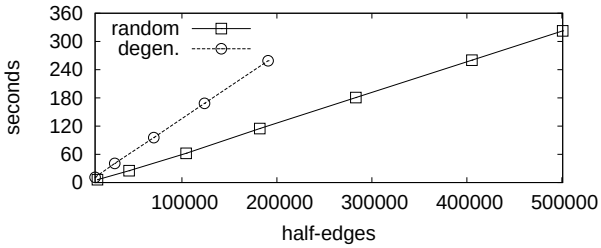
The tables below report average running times measured on a 1.2 GHz Pentium IIIM system with 512 KB of cache running Linux.

series	n	segs	nodes	h.edges	bits	time
random	30	226	2933	11038	99	6.1
random	60	454	11417	44440	99	25.1
random	90	680	26579	104474	100	62.2
random	120	940	46117	181978	100	114.8
random	150	1226	71594	283144	99	180.7
random	180	1460	102298	405312	101	260.2
random	200	1554	126278	500888	101	322.5

series	n	segs	nodes	h.edges	bits	time
degen.	30	243	2313	8604	116	11.1
degen.	60	534	7627	29284	116	40.8
degen.	90	722	17983	70378	120	95.6
degen.	120	1027	31504	123814	114	168.4
degen.	150	1292	48362	190698	116	258.9

Each row states number of input curves, total number of segments after splitting of input curves, number of nodes and half-edges in the resulting graph, the average bit length of a curve's longest coefficient, and the average running time in seconds. The fraction of time spent on the analysis of individual curves is well below one second even for the largest instances. The executable was compiled with `g++` 3.1. LEDA 4.4 was used for the exact number types and the internal data structures of our sweep code. All benchmark instances are computed in the original coordinate system; that is, they have not been transformed with a random shear.

The plot below shows the running times as a function of the number of computed half-edges (output complexity). In accordance with the theoretical analysis, the output complexity looks almost linear. However, the output size is quadratic in the number of curves, as for the straight-line case.



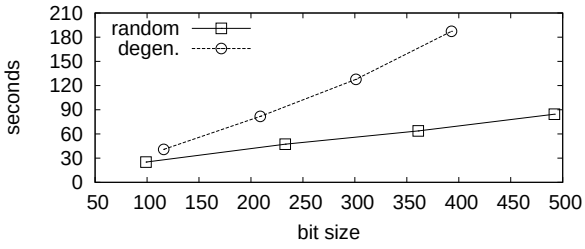
The modular filter for gcd computations (which are otherwise performed using the classical subresultant algorithm [18]) has been found to accelerate the "random 30" instance by a factor of 9. This filter and the caching of curve and curve pair analyses for repeated use in predicate evaluations are important sources of efficiency in our implementation.

As dominant reasons for slowdown in the degenerate instances we see: Multiple intersections of f, g cause multiple zeroes in $\mathrm{res}(f,g,y)$. High-degree vertices correspond to equality of event point x-coordinates. Both phenomena entail the computation of gcds that are avoided by the modular filter in the generic case. The analysis of a tangential intersection requires the consideration of a Jacobi curve, involving the refinement of an isolating interval against three additional resultants of degree 12.

The *coefficient growth* benchmark gives the following results.

random 60	segs	nodes	h.edges	bits	time
original	454	11417	44440	99	25.1
with $s = 10^2$	454	11437	44520	233	47.3
with $s = 10^4$	454	11417	44440	361	63.8
with $s = 10^6$	454	11417	44440	492	84.5

degenerate 60	segs	nodes	h.edges	bits	time
original	534	7627	29284	116	40.8
with $s = 10^2$	534	7639	29332	209	81.8
with $s = 10^4$	534	7627	29284	301	127.7
with $s = 10^6$	534	7627	29284	393	187.3



In this setup, the increase in running time is bounded by the growth of running time for the exact arithmetic, especially multiplication. Our experiments used LEDA with the $O(N^{\log_2 3})$ Karatsuba multiplication. This superlinear growth is well-visible for the degenerate instances, and indeed they invoke more symbolic computations (such as gcds which are otherwise avoided and additional resultants for Jacobi curves). For the random instances, the superlinear term is less pronounced, reflecting the fact that the coefficients of curves and resultants grow, whereas the interval boundaries in root isolation and comparison do not, so that in these parts of the algorithm only one of the two operands of multiplication grows.

11. REFERENCES

[1] S. Abhyankar and C. Bajaj. Computations with algebraic curves. In *ISSAC '88*, LNCS 358, pages 274–284. Springer, 1989.

[2] P. K. Agarwal and M. Sharir. Arrangements and their applications. In J.-R. Sack and J. Urrutia, editors, *Handbook of Computational Geometry*, pages 49–119. Elsevier, 2000.

[3] J. L. Bentley and T. A. Ottmann. Algorithms for reporting and counting geometric intersections. *IEEE Trans. Comput.*, C-28(9):643–647, Sept. 1979.

[4] E. Berberich, A. Eigenwillig, M. Hemmer, S. Hert, K. Mehlhorn, and E. Schömer. A computational basis for conic arcs and boolean operations on conic polygons. In *ESA'02*, LNCS 2461, pages 174–186. Springer, 2002.

[5] E. Brieskorn and H. Knörrer. *Plane Algebraic Curves*. Birkhäuser, Basel, 1986.

[6] C. Burnikel, R. Fleischer, K. Mehlhorn, and S. Schirra. A strong and easily computable separation bound for arithmetic expressions involving radicals. *Algorithmica*, 27(1):87–99, 2000.

[7] C. Burnikel, S. Funke, K. Mehlhorn, S. Schirra, and S. Schmitt. A separation bound for real algebraic expressions. In *ESA '01*, LNCS 2161, pages 254–265. Springer, 2001.

[8] J. Canny. *The Complexity of Robot Motion Planning*. ACM – MIT Press Doctoral Dissertation Award Series. MIT Press, Cambridge, MA, 1987.

[9] G. E. Collins. Quantifier elimination for real closed fields by cylindrical algebraic decomposition. In *Proc. 2nd GI Conf. on Automata Theory and Formal Languages*, LNCS 33, pages 134–183. Springer, 1975.

[10] G. E. Collins and R. Loos. Real zeros of polynomials. In B. Buchberger, G. E. Collins, and R. Loos, editors, *Computer Algebra: Symbolic and Algebraic Computation*, pages 83–94. Springer-Verlag, New York, NY, 1982.

[11] CORE library project. http://www.cs.nyu.edu/exact/core/.

[12] D. Cox, J. Little, and D. O'Shea. *Ideals, Varieties, and Algorithms*. Springer-Verlag, New York, 2nd edition, 1997.

[13] O. Devillers, A. Fronville, B. Mourrain, and M. Teillaud. Algebraic methods and arithmetic filtering for exact predicates on circle arcs. In *Proc. 16th Annu. ACM Symp. Comput. Geom.*, pages 139–147, 2000.

[14] A. Eigenwillig. Exact arrangement computation for cubic curves. Master's thesis, Saarland University, Saarbrücken, Germany, 2003.

[15] I. Emiris, A. Kakargias, S. Pion, M. Teillaud, and E. Tsigaridas. Towards an open curved kernel. In *Proc. 20th Annu. ACM Sympos. Comput. Geom.*, 2004.

[16] I. Z. Emiris and E. P. Tsigaridas. Comparison of fourth-degree algebraic numbers and applications to geometric predicates (revised version). Report ECG-TR-302206-03, Oct. 2003. http://www-sop.inria.fr/prisme/ECG/Results/.

[17] E. Flato, D. Halperin, I. Hanniel, O. Nechushtan, and E. Ezra. The design and implementation of planar maps in CGAL. *ACM J. of Exper. Algor.*, 5, 2000.

[18] K. O. Geddes, S. R. Czapor, and G. Labahn. *Algorithms for Computer Algebra*. Kluwer, 1992.

[19] N. Geismann, M. Hemmer, and E. Schömer. Computing a 3-dimensional cell in an arrangement of quadrics: Exactly and actually! In *Proc. 17th Annu. ACM Sympos. Comput. Geom.*, pages 264–271, 2001.

[20] C. G. Gibson. *Elementary Geometry of Algebraic Curves*. Cambridge University Press, 1998.

[21] D. Halperin. Arrangements. In J. Goodman and J. O'Rourke, editors, *Handbook of Discrete and Computational Geometry*, chapter 24. CRC Press LLC, Boca Raton, FL, 2nd edition, 2004.

[22] H. Hong. Efficient method for analyzing topology of plane real algebraic curves. In *Proceedings of IMACS-SC 93, (Lille, France)*, June 1993.

[23] J. Keyser, T. Culver, D. Manocha, and S. Krishnan. MAPC: A library for efficient and exact manipulation of algebraic points and curves. In *Proc. 15th Annu. ACM Sympos. Comput. Geom.*, pages 360–369, 1999.

[24] K. Mehlhorn and S. Näher. *LEDA: A Platform for Combinatorial and Geometric Computing*. Cambridge University Press, Cambridge, UK, 2000.

[25] P. Pedersen. Multivariate sturm theory. In *Proceedings of Applied Algebra, Algebraic Algorithms and Error-Correcting Codes*, pages 318–323, 1991.

[26] F. P. Preparata and M. I. Shamos. *Computational Geometry: An Introduction*. Springer, New York, NY, 1985.

[27] T. Sakkalis. The topological configuration of a real algebraic curve. *Bulletin of the Australian Mathematical Society*, 43:37–50, 1991.

[28] M. Seel. Implementation of planar Nef polyhedra. Report MPI-I-2001-1-003, MPI für Informatik, 66123 Saarbrücken, Germany, Aug. 2001.

[29] J. von zur Gathen and J. Gerhard. *Modern Computer Algebra*. Cambridge University Press, 1999.

[30] R. Wein. High level filtering for arrangements of conic arcs. In *ESA'02*, LNCS 2461, pages 884–895. Springer, 2002.

[31] N. Wolpert. *An Exact and Efficient Approach for Computing a Cell in an Arrangement of Quadrics*. PhD thesis, Saarland University, Saarbrücken, Germany, 2002.

[32] N. Wolpert. Jacobi curves: Computing the exact topology of arrangements of non-singular algebraic curves. In *ESA'03*, LNCS 2832, pages 532–543. Springer, 2003.

APPENDIX

We imposed certain position conditions that do not restrict the geometry of the arrangement but its algebraic representation, in particular the choice of the y-axis. In this Appendix, we will first derive a rough constant upper bound on the number of forbidden directions of the y-axis by rephrasing these conditions on the coordinate system as conditions that certain lines (which are well-defined without reference to a specific coordinate system) shall be non-vertical.

Afterwards, we discuss how to detect and remove violations of the conditions.

Let us first consider the conditions for a single curve f:

- A curve f is y-regular iff its highest-order terms (HOT) are not divisible by x. The HOT form a homogeneous polynomial of degree $\deg(f)$ which decomposes into linear factors over $\mathbb{C}$. These are the *complex asymptotes* of f. We obtain: f is y-regular iff none of its complex asymptotes is vertical. A cubic curve has at most 3 distinct complex asymptotes.
- A curve f has no vertical flexes iff the unique tangents in all flexes are non-vertical. A cubic curve has at most 9 flexes [20].
- The non-verticality of tangents in the singularities of a y-regular curve f excludes at most 2 further directions: For an irreducible cubic we demand the non-verticality of the two tangents in the singularity. For a cubic consisting of a conic and a line intersecting in two distinct singularities we demand non-verticality of the respective tangents to the conic component.

This yields an upper bound of **14** on the number of forbidden directions for a single curve.

Now consider a curve pair $\{f, g\}$.

- The noncoverticality of the ≤ 9 distinct intersection points is equivalent to the nonverticality of the $\leq \binom{9}{2} = 36$ lines joining any two of them.
- The noncoverticality of intersection points to one-curve events of f is equivalent to the nonverticality of any line ℓ that both contains an intersection point $p \in f \cap g$ and has a tangential intersection $q \neq p$ with f. This is because a tangential intersection of ℓ and f at q, according to the definition from Section 4, means that ℓ is a tangent to f at q, or that f is singular at q.
  The noncoverticality of one-curve events of f and one-curve events of g is equivalent to the nonverticality of any line ℓ which has a tangential intersection with both f and g, by the same argument.
- There are ≤ 9 non-singular intersection points, each with one unique tangent on f and on g, respectively. These points are not extreme iff their $\leq 2 \cdot 9 = 18$ tangents are non-vertical.
- The Jacobi curve $h = f_x g_y - f_y g_x = |\nabla f \ \nabla g|$ is well-defined independent of a specific choice of coordinates. It has $\deg(h) \leq 4$ complex asymptotes whose nonverticality is equivalent to y-regularity of h.
  There are ≤ 4 non-singular intersections of f and g with multiplicity 2, and no line through any of them that has a tangential intersection with h is allowed to be vertical (by the same argument as above).
- The transversal intersection of f_y and g in an acnode of a triangle f holds for all but at most one choice of a y-axis, because shearing the y-axis means for f_y just to replace it by another element of the pencil spanned by the two transversally intersecting curves f_y and f_x.

To obtain estimates on the number of lines ℓ fulfilling a condition of the form "ℓ has a tangential intersection with f (and we don't care where)", we lift the scene to the complex projective plane by homogenizing $f(x, y)$ to $F(x, y, z) = z^{\deg(f)} f(\frac{x}{z}, \frac{y}{z})$, and we consider the set of all complex-projective lines L that have a tangential intersection with F. Their duals L^* form an algebraic curve $\tilde{F}$ in dual space (unless F has a line component, but we leave out this special case for brevity) which has degree $\deg(F)(\deg(F) - 1)$ [5, pp. 252+]. For our setting, this means $\deg(\tilde{F}) \leq 6$. (Removing the line components from $\tilde{F}$, which reflect the intersections in singularities, gives the well-known *dual curve* F^* of F.)

With these notions, we can now dualize "a line L through p having a tangential intersection with curve F" to "an intersection point L^* of line p^* and curve $\tilde{F}$". This yields an upper bound of $\deg(\tilde{F})$ on the number of such lines L.

Let us proceed to bound the number of lines L tangent to F outside some fixed non-singular point p of F. Recall that, for degree reasons, a line can be tangent to a cubic at no more than one point. It follows that we can dualize "a line L through p having a tangential intersection with curve F outside p" to "an intersection point L^* of line p^* and curve $\tilde{F}$ distinct from T^*". It is known [5, p. 255] that the intersection of $\tilde{F}$ and p^* at T^* has multiplicity 2. Hence we have an upper bound of $\deg(\tilde{F}) - 2$ on the number of lines L.

Finally we can dualize "a line L having tangential intersections with curves F and G" to "an intersection point of $\tilde{F}$ and $\tilde{G}$". Bézout's Theorem implies that no more than $\deg(\tilde{F}) \deg(\tilde{G})$ such lines L exist.

Returning to the original question, we obtain the following bounds:

- Through each of the ≤ 9 intersection points p, there are $\leq 6 - 2 = 4$ lines that have a tangential intersection with f outside p; same for g. This forbids $\leq 2 \cdot 4 \cdot 9 = 72$ directions of lines.
- There are $\leq 6 \cdot 6 = 36$ lines that have a tangential intersection with both f and g.
- Through each of the ≤ 4 non-singular intersection points of f and g with multiplicity 2, there are $\leq 4 \cdot 3 = 12$ lines that have a tangential intersection with the Jacobi curve h. This forbids $\leq 4 \cdot 12 = 48$ directions of lines.

These three items forbid $\leq 72 + 36 + 48 = 156$ directions. Together with the $\leq 36+18+4+1 = 59$ forbidden directions from the preceding list, we obtain that no more than $156 + 59 = \mathbf{215}$ directions of a y-axis are forbidden per analyzed curve pair.

Since there are infinitely many possible y-axes, a random choice will pick one that is permissible for all curves and curve pairs analyzed during the algorithm with probability 1. Hence our strategy for finding a permissible, or generic, coordinate system is this: Shear the input scene with a random shearing parameter $r \in \mathbb{Q}$ and run the algorithm. Check the conditions along the way. If a violation is detected, pick a new r and restart.

A *shear* is an invertible linear map $S_r \colon (x, y) \mapsto (x + ry, y)$ for a fixed $r \in \mathbb{Q}$. It leaves the x-axis fixed and tilts the y-axis. Shearing a point p means replacing it by $S_r(p)$.

Shearing a curve f means replacing f by $f \circ S_r^{-1}$, because $f(p) = 0 \Leftrightarrow (f \circ S_r^{-1})(S_r(p)) = 0$.

From which range should the algorithm choose r? We do not recommend to compute an a priori bound on the number of forbidden directions and define a range larger than that, because most forbidden directions will lie outside that range, being irrational or rational with large denominator, and choosing from a large range comes at a price: A value r of binary encoding length s will increase the coefficient size of a curve f by a factor of $s \deg(f)$. Instead, start with a small range and increase its size with the number of past failures.

It remains to describe how we check the conditions during the curve and curve pair analyses.

Let us begin with the analysis of a curve f:

- Checking y-regularity is trivial.
- Vertical flexes can be detected easily, using their characterization as elements of $f \cap f_y \cap f_{yy} \setminus f_x$ and exploiting the fact that $\deg(\mathrm{res}(f_y, f_{yy}, y)) \leq 2$, which allows for explicit arithmetic in FRE.
- Vertical tangents at a rational singularity (α, β) are evident from $f(x + \alpha, y + \beta)$ having lowest-order terms that are not y-regular.
  Vertical tangents at irrational singularities can occur only for crunodes and are detected from the differing number of arcs over the incident intervals.

Now for curve pairs $\{f, g\}$:

- Let α be a root of $\mathrm{res}(f, g, y)$. If both curves have a singularity at $x = \alpha$, we are in the case of Proposition 13 and thus essentially done. For the remaining cases we extend the notion of a resultant. One can define the *first (scalar) subresultant* $\mathrm{sres}_1(f, g, y)$ of f and g (a.k.a. the first principal subresultant coefficient) such that $\mathrm{res}(f, g, y)(\alpha) = \mathrm{sres}_1(f, g, y)(\alpha) = 0$ implies $\deg(\gcd(f(\alpha, y), f(\alpha, y))) \geq 2$ [29, 6.10]. Hence our check reduces to testing whether α is also a root of $\mathrm{sres}_1(f, g, y)$.
- An intersection covertical to a singularity is detected directly for a known rational singularity. For an irrational singularity we first use the subresultant criterion to rule out covertical intersections. Then we verify the equality of the one intersection with the singular point by checking whether the expected transposition of arcs takes place. Absence of transposition also covers the transversality condition on f_y and g for an intersection in an acnode of a triangle f.
- An intersection covertical or equal to an extreme point of f is simply detected by the equality of the corresponding roots of $\mathrm{res}(f, g, y)$ and of the square-free factor of multiplicity 1 of $\mathrm{res}(f, f_y, y)$.
- The coverticality or equality of a non-singular intersection of f and g with multiplicity 2 and a one-curve event on their Jacobi curve h is detected analogously.