DYNAMIC ASPECTS IN COMPUTATIONAL GEOMETRY

Jürgen Richter-Gebert Ulrich H. Kortenkamp

Abstract

Computational Geometry very often focuses on static problems, like computing the convex hull or Voronoi complex of a given set of points. Fundamentally new questions arise when the objects under consideration are no longer static, but may move around with respect to certain constraints. This scenario is not unusual, for instance every mechanism can be considered as a dynamic geometric entity.

We here focus on the new areas of problems that arise from genuinely dynamic effects. Constructions from elementary geometry play a crucial role in this context, since they form first natural instances of non-trivial examples where it is reasonable to study the dynamic behavior. One of the most fundamental problems arises when one considers *one* point of an intersection of a line and a circle and allows the line to move around. Since the point of intersection is not unique, a computer program has to decide for every "discrete snapshot" which of the two intersection points is meant. If this decision is not made correctly a "path-jumping" of the point may occur. A careful analysis of the situation shows that for a satisfactory resolution of the problem one has to embed the configuration in an ambient complex projective space. One even has to take monodromy effects and underlying Riemann surfaces into account.

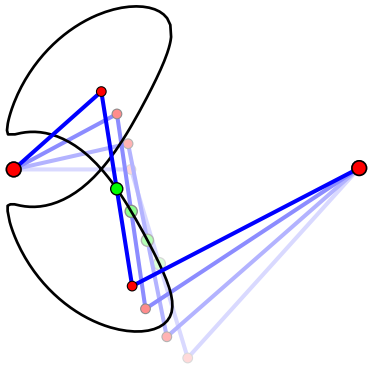
We will develop a theory for the study of these phenomena. After this, we will investigate the algorithmic complexity of "making the right decisions". It will turn out that even in very weak versions this problem is NP-hard. In some stronger versions it is PSPACE hard or even undecidable.

Introduction

Imagine any construction of elementary geometry (say a ruler and compass construction of the midpoint of two points A and B). It consists of certain *free* elements (the points A and B) and certain *dependent* elements whose position is determined by the position of the free elements. Each specific picture of such a construction is a snapshot taken out off the whole continuum of all possible drawings for all possible locations of the free elements. If we move the free elements we can walk continuously from one instance of the construction to another one. During such a walk a continuous motion of the free elements should be reflected by in a continuous movement of the dependent elements.

This article deals with those effects and problems that genuinely arise from such a dynamic setup of geometry. The research that led to the results presented in this paper was motivated by the desire (and the actual work) of implementing a concrete software package for doing dynamic geometry on a computer [11, 12]. With such a program one

should be able to do constructions of elementary geometry with a few mouse clicks. After a construction is made one should be able to pick the free elements with the mouse, drag them around, while the whole construction follows accordingly.



Geometric locus under the motion of a "three bar linkage". The use of complex path tracing generates reliably complete real branches of algebraic curves. The orientation heuristics that are usually used by other software can only generate partial loci.

The innocent looking requirement of *continuity of dependent elements* turned out to be fundamentally hard to fulfill. In fact one has to rely on notions of complex function theory and Riemann surfaces to get a mathematically sound treatment of these effects [6, 13]. Here we report on these problems from a conceptual point of view and from a complexity theoretic point of view. We explain how a complexified setup can be used to obtain continuity. However we will also explain that actually computing the correct picture that results from a movement can be algorithmically hard. The complexity classes in most algorithmic questions related to that context range from NP-hard problems via PSPACE-hard problems up to even undecidable problems. In particular one can prove that ...

... it is in general PSPACE-hard to decide whether two instances of the same construction can be continuously deformed into each other by moving the base elements along a real path.

... it is NP-hard to calculate the position of the dependent elements after a specific move of a free element.

... it is undecidable whether two instances of a construction involving "wheels" can be continuously deformed into each other by moving the base elements.

Detailed proves of these results can be found in [13].

Although the results of this article arose from the study of configuration spaces of elementary geometric constructions they are naturally related to many other setups in the area of geometry. Among those are the study of configuration spaces of mechanical linkages [2, 5], realization spaces of oriented matroids [8, 1, 9, 15] and polytopes [10], and the piano movers problem (with possibly many pianos) [4, 14]. Our complexity results are partially generalizations and strengthenings of known complexity results in these areas. Besides the narrow

context of dynamic geometry our results are relevant for all areas where geometric objects are moved around under certain geometric constraints, like robotics, parametric CAD [3], virtual reality, or computational kinematics. Our results imply that many problems of these areas are computationally difficult (like the *persistent naming problem* of parametric CAD [3] or the *navigation problem* of computational kinematics.)

1 Geometric straight line programs

We restrict ourselves to the following particularly simple scenario, which arises in the context of interactive geometry software: *a dynamic setup for elementary geometry.* Nonetheless, we want to emphasize the the underlying methods apply also to much more general contexts.

Large parts of elementary geometry are based on the theory of *ruler and compass constructions.* Such constructions are usually done by first drawing a set of "free points" in the plane and then proceeding by adding new objects with operations like: "join of two points", "intersection", "circle given by midpoint and perimeter point". We formalize constructions that use these operations by the concept of *geometric straight line programs.*

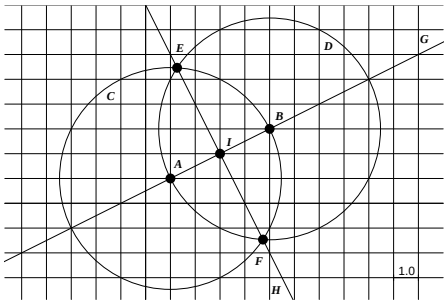
We assume that the objects are given by suitable parameters (coordinates). A geometric straight line program (GSLP) is a sequence of program statements, where each statement describes the position of a new elementary object. The operations we allow are:

`P=FreePoint(x,y)`	Point `P` is at position (x,y)
`L=Join(P1,P2)`	Line `L` is the join of points `P1` and `P2`
`C=Circle(M,P)`	Circle `P` is the a circle with center `M` through `P`
`P=Meet(L1,L2)`	Point `P` is the meet of lines `L1` and `L2`
`P=IntersectionCL(C,L)`	Point `P` is the intersection of line `L` and circle `C`
`P=IntersectionCC(C1,C2)`	Point `P` is the intersection of the circles `C1` and `C2`

The first four of these operations produce a unique element. The last two operations have an intrinsic ambiguity: A line and a circle, or two circles, can have more than one intersection. It can even happen that a line and a circle do not intersect at all. So, in general a GSLP does not describe a *unique* geometric situation. One can even get stuck during the execution of a GSLP if an intersection does not exist. However, for a given geometric configuration of points, lines and circles it is easy to check whether it is compatible the definition of a given GSLP. Such a configuration is then called an *instance* of the GSLP. It is clear that for every GSLP (with concrete values for the coordinates of the free points) there are at most finitely many possible compatible instances, since this number is bounded by 2^n where n is the number of ambiguous choices.

Example: The following GSLP encodes a construction of the midpoint of two points $A = (1, 1)$ and $B = (5, 3)$.

```
1:  A=FreePoint(1,1);
2:  B=FreePoint(5,3);
3:  C=Circle(A,B);
4:  D=Circle(B,A);
5:  E=IntersectionCC(C,D);
6:  F=IntersectionCC(C,D);
7:  G=Join(A,B);
8:  H=Join(E,F);
9:  I=Meet(G,H);
```



In this GSLP points E and F have the same definition as being the intersection of the two circles. Only for the "right" choices you will obtain an actual instance of the GSLP in which the final point is indeed the midpoint of A and B.

2 The problem of continuity

We now study a GSLP in a dynamic setup. For this consider the coordinates of the free points parametrized by $\lambda \in [0, 1]$. We are interested in a reasonable behavior of the dependent elements of the construction. Or more intuitively: assume you constructed the above picture with an interactive geometry program, how should the dependent elements behave, when you move A or B?

It is clear that the only freedom for the choice of dependent elements comes from the ambiguities of circle intersections. A desirable behavior would be the following:

> *"While the free points move continuously all dependent objects move continuously as well." In other words: "The coordinates of all elements are continuous functions in λ."*

At first sight it is not clear whether this requirement is satisfiable at all (compare [7]). In fact, all geometry systems and programs for parametric CAD that are currently available suffer from non-continuous behavior of dependent elements; while you move a free point it may happen that parts of the construction jump from one place to another. In particular, we must find a way to deal with the problem of vanishing intersections.

3 Complex projective geometry

To fulfill the continuity requirements we first need to fix a topology. For this consider the Euclidean plane as a subset of the projective plane. This gives us "points at infinity" and the desired topology is induced by the topology of the manifold structure that underlies the projective plane. In particular *point can move to infinity and can "continuously" come back from the opposite side of the (embedded euclidean) plane.*

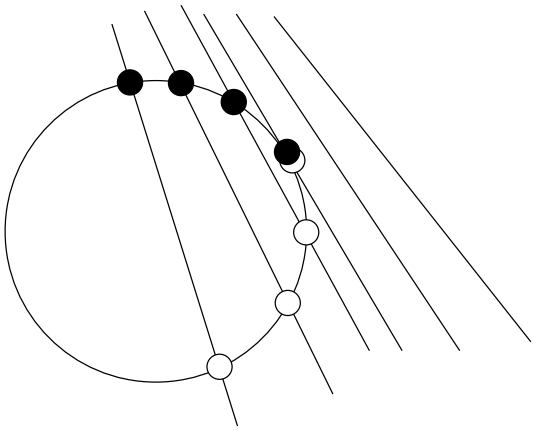
The next and more important enlargement of the setup comes from embedding the whole situation in complex space. For this we just assume that the coordinates of our objects may be also given by complex numbers and study *complex two-dimensional projective geometry.* Since every complex number can be described by two real numbers this space has real dimension four. Nevertheless, the Euclidean plane can still be found as a substructure of this space.

Compared to the real setup complex calculations have a great advantage: Intersections never (!) vanish. Even if a line and a circle do not intersect in the real Euclidean plane, it is still reasonable to speak about intersection points with complex coordinates, since their coordinates are just solutions of suitable quadratic equations.

4 Moving in complex spaces

Assuming that the free points perform a continuous motion there is an easy strategy of dealing with multiple intersections: *Consider both intersections as individual objects and trace their paths through complex space.* As long as the intersection points do not coincide one can easily tell them apart and (in a continuous model of computation) trace them as individual objects.

In the Euclidean plane there can be one, two or none intersections of a line and a circle. In complex space these intersections never vanish. If we can avoid "singular situations" it is always possible to trace the two intersections individually.



But how can we avoid the situations where the intersection points coincide? There we again take advantage from the complex setup. Let us call an instance of a GSLP *non-singular* if no "double intersections" occur. If we now move from one non-singular instance of a GSLP to another non-singular instance of the GSLP there is always a path through complex space that avoids all singular situations. This is a consequence of the fact that a non-constant analytic function can have no accumulation points as its set of roots. This means that we can always take a "complex detour" that allows us to individually trace all dependent objects of a construction. For such paths we obtain perfect continuity. In fact for an analytic path that avoids all singularities the coordinates of the dependent objects are analytic functions in the input parameter.

5 Real benefits

The approach above may sound far too complicated to resolve the original problem of Euclidean geometry. However, it can be proved that as soon as we want to have continuous behavior of the dependent elements, there is only one way to make the decisions and that this choice coincides with our solution. Although the setup uses complex numbers we have several benefits in the real case. Here is a list of keywords of what becomes possible under this setup:

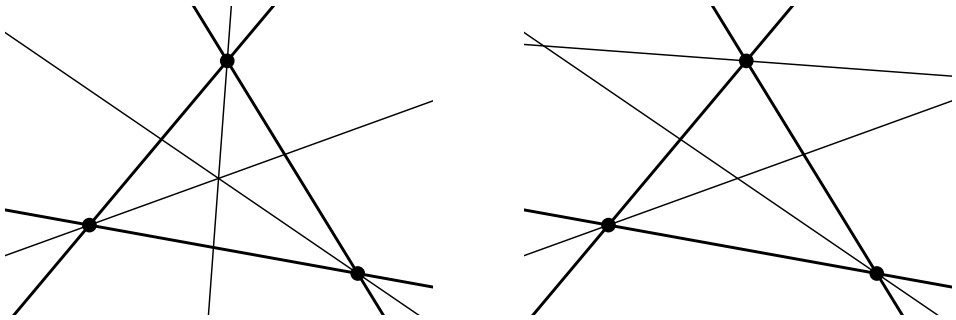
- All derived elements behave **analytically**.
- The solution is **unique**.
- The behavior is **globally consistent**.
- There are **no jumping elements**.
- Geometric theorems are true **once-and-forever**.
- **Randomized proving** works.
- **Self exploring Loci**.
- We have generic tools for **computational kinematics**.

6 Randomized proving and continuity

To keep its own data structures clean, our program needs consistent information about incidences and equalities that occur in configurations. Such incidences may either be trivial consequences of the construction or arise from geometric theorems like the altitudes of a triangle meeting in a point. We actually get this information by a *randomized theorem checking* technique. Enough random instances of a configuration are generated and for each of them the conjectured incidence is checked. This is done until the program either accepts the theorem with a certain high probability or it rejects it, if a counterexample was found.

To be really reliable, the randomized theorem checking engine needs enough (!) random (!) examples. Again there arises a theoretical problem which originates from ambiguities in geometric constructions. Consider the theorem stating that *the angular bisectors of the sides of a triangle meet in a point.* Due to the intrinsic ambiguity of the angular bisector operation this sentence stated as such is not true. Consider the drawing in the picture below. It shows two valid instances of the construction: Take three points — form the three joins of any pair of them — draw the three angular bisectors of any pair of lines. In one of the drawings the chosen angular bisectors meet in the other they do not.

For the theorem checking the program does a "random walk" that stays always in the desired component of the configuration space. Staying in the correct component during this random walk again depends on consistent and continuous behavior of dependent elements.



Depending on the choices the angular bisectors of a triangle can intersect or not.

7 Complexity issues

Let us briefly sketch the issues of algorithmic complexity that arise in this context (details and proofs can be found in [13]). In fact, two fundamental questions can be formulated that capture the main algorithmic questions of dynamic geometry:

- **Reachability problem:** Is it possible to move the free points such that a first instance is smoothly deformed into a specific second one?
- **Tracing problem:** How can a dynamic geometry program decide after a move what instance to draw for the new position of the free elements?

In fact it makes an essential difference whether one allows in the reachability problem only real coordinates of the free elements or also complex paths. For the real version it can be proved that (after suitable formalization) the reachability problem is in general PSPACE-hard. It is still NP-hard if one restricts oneself to constructions that only use *join*, *meet*, and *angular bisector* operations. The Tracing problem turns out to be (at least) NP-hard. We briefly sketch how the above results can be achieved. We first focus on the following result:

Theorem 7.1. *Let $\mathcal{C}$ be a geometric straight line program that uses at most three* angular bisector *operations and except of this only* join *and* meet *operations. Furthermore let I_1 and I_2 be two instances of $\mathcal{C}$ that differ only in the choice of one angular bisector. It is NP-hard to decide whether I_1 can be moved continuously into I_2 by a real continuous motion of the free points of $\mathcal{C}$.*

In order to sketch a proof of this theorem we describe how a reduction of the well known 3-SAT problem to the real reachability problem can be achieved. Our reduction proceeds

in several steps. The first step consists of transforming 3-SAT to an algebraic setup. For this let us first formally state the 3-SAT decision problem:

3-SAT: *Let $B = (b_1, \ldots, b_n)$ be boolean variables, and let the literals over B be $\widetilde{B} = (b_1, \ldots, b_n, \neg b_1, \ldots, \neg b_n)$. Furthermore let $C_1, \ldots, C_k$ be clauses formed by disjunction of three literals from $\widetilde{B}$. Decide whether there is a truth assignment for B that satisfies all clauses $C_1, \ldots, C_k$ simultaneously.*

We may w.l.o.g. assume that each variable occurs at most once in each clause. We give a (polynomial time) procedure that transfers each instance of 3-SAT into a corresponding problem concerning the roots of a multivariate polynomial. Let $b_1, \ldots, b_n$ be the boolean variables and let $C_1, \ldots, C_k$ be the clauses of a concrete 3-SAT S. To each b_i we assign a formal variable x_i. For a literal $l_i \in \{b_i, \neg b_i\}$ we set

$$f(x_i) := \begin{cases} x_i & \text{if } l_i = b_i, \\ 1 - x_i & \text{if } l_i = \neg b_i, \end{cases}$$

Assume that for each $j = 1, \ldots, k$ the clause C_j is of the form $l_r^j \vee l_s^j \vee l_t^j$ where the literal l_i^j is either b_i or $\neg b_i$. We set

$$F_j := f(l_r^j) \cdot f(l_s^j) \cdot f(l_t^j)$$

Finally we set

$$F_S = \sum_{j=1}^{k} F_j.$$

By this translation for instance the 3-SAT formula $(b_1 \vee \neg b_3 \vee b_5) \wedge (\neg b_2 \vee b_4 \vee \neg b_5)$ is translated to $(x_1 \cdot (1 - x_3) \cdot x_5) + ((1 - x_2) \cdot x_4 \cdot (1 - x_5))$. The satisfying truth assignments for S and the roots of $F(S)$ in $[0, 1]^n$ are related by the following lemma (here $[0, 1]$ denotes the closed interval between 0 and 1).

Lemma 7.2. *S has a satisfying truth assignment if and only if there are $(x_1, \ldots, x_n) \in [0, 1]^n$ with $F_S(x_1, \ldots, x_n) = 0$.*

Proof. If S has a concrete satisfying truth assignment $(b_1, \ldots, b_n) \in \{\textsc{True}, \textsc{False}\}^n$ we set

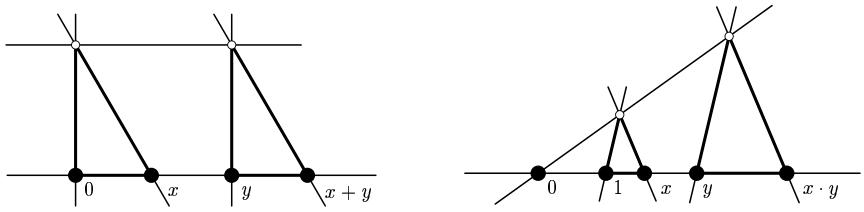
$$x_i := \begin{cases} 0 & \text{if } b_i = \textsc{True}, \\ 1 & \text{if } b_i = \textsc{False}, \end{cases}$$

Since every clause contains at least one true literal we the get that all $f_1, \ldots, f_k$ are zero. This yields that F_S is zero as well. Conversely, assume that there are values $(x_1, \ldots, x_n) \in [0, 1]^n$ such that $F_S(x_1, \ldots, x_n) = 0$. If the x_i are chosen in the interval $[0, 1]$ all f_j are non-negative. Thus if $\sum_{j=1}^{k} f_j = 0$ implies that all f_j are zero. However each f_i can only be zero if at least one of its factors is zero. By setting

$$b_i := \begin{cases} \textsc{True} & \text{if } x_i = 0, \\ \textsc{False} & \text{if } x_i \neq 0, \end{cases}$$

We get a satisfying truth assignment for S. □

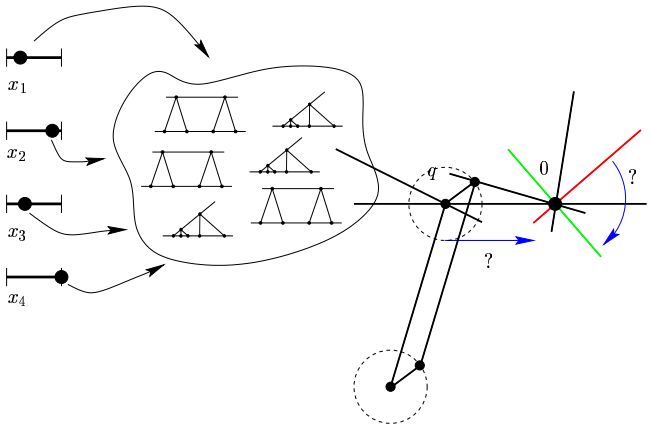
In the next step of our reduction we simulate the algebraic computation of the polynomial F_S by an elementary geometric construction. For this we take a line on which we fix positions of the points 0 and 1 to define a scale of measurement. Each point on the line corresponds then to a certain value. Multiplication and addition of values on the line can be performed by the classical *von Staudt constructions* (see picture below). (The parallelisms that occur in these construction can – after fixing a line at infinity – entirely expressed by joins and meets.)



Von Staudt constructions for addition and multiplication.

The calculation of the polynomial F_S can be decomposed into elementary arithmetic operations. The entire construction of the geometric analogon to F_S contains n free points $x_1, \ldots, x_n$ as input variables and one dependent point q as output variable. If we restrict the position of the input points to the interval $[0, 1]$ on the computation line, we see that the point q can only be moved to the origin if the original 3-SAT problem S was satisfiable. Restricting the input points to the interval can be done by a small geometric gadget that uses Thales Theorem.

Finally, we construct a semi-free point that can move only on a small circle around the output point. We can detect whether this point can circle around the origin by an angular bisector construction. For this we join this point to the origin and form the three times iterated angular bisector with our calculation line. Schematically the construction is shown in the following picture.



Construction of a "geometric locker".

The whole construction forms a kind of geometric locker. Opening the locker corresponds to interchanging

We associate the following reachability problem to this construction. Is it possible from an arbitrary position of the input points to move in a real path such that the final angular bisector is rotated by an angle of 90^o and all free points reached their initial position again. The whole construction forms a kind of geometric locker. Opening the locker corresponds to achieve the desired position of the angular bisector, but for this one has to know the correct positions of the code dials (the input variables). Opening the locker proceeds by first moving the dials to the right position, changing the the angular bisectors position and finally moving the dials back to the original position.

It is not difficult to prove that the corresponding reachability question is equivalent to finding a satisfying truth assignment for our original 3-SAT problem S. The argument for this goes as follows:

- The only way that the final angular bisector can make this 90^o turn is that the line through the origin and through the point that is restricted to the circle around q makes a full turn.
- This is only possible if p and the origin get so close such that the circle around p contains the origin.
- This is only possible if the input points x_i can be moved to a position that corresponds to a satisfying truth assignment of S.

Thus changing the position of the final angular bisector requires that we know a satisfying truth assignment for S. This finishes our sketch of the proof of Theorem 7.1.

The other main Theorem one can proof is the following.

Theorem 7.3. *Given a geometric straight line Program $\mathcal{P}$ that contains exactly one free point p. Furthermore given two instances A and B such that p is at position a in A and p is at position b in B. Let $p(t) : [0,1] \to [a,b]$ be a concrete (straight) movement of p with $p(0) = a$ and $p(1) = b$. It is NP-hard to decide whether a continuous evaluation of $\mathcal{P}$ under this movement that starts at instance A ends up at the instance B.*

We very briefly sketch idea behind the proof of this theorem. We combine our previous construction with a kind of "automatic safe cracker" that while moving the point p explores systematically all positions of the input variables, and for every position tries to change the location of the angular bisector. If the original 3-SAT instance S had a satisfying truth assignment, then the final angular bisector will have changed its position when p reached its end situation $p(1)$. Hence from the final instance of the configuration we can read of whether S had a satisfying truth assignment.

8 Remarks

More information about the software and the underlying mathematics can be found on the Cinderella Website at `http://www.cinderella.de`.

References

[1] H. Günzel, *The universal partition theorem for oriented matroids,* Discrete Comput. Geom., **19**, (1998), 521–551.

[2] J. Hopcroft, D. Joseph & S. Whitesides, *Movement problems for 2-dimensional Linkages,* SIAM J. Comput., **13**, (1984), 610–629.

[3] Ch. M. Hoffmann, *Solid Modeling,* in: J.E. Goodman & J. O'Rourke. (eds.): *Handbook of Discrete and Computational Geometry*, Lecture Notes in Mathematics **1346**, CRC Press, Boca Raton, New York, 1997, 863–880.

[4] J. Hopcroft, J.T. Schwarz & M. Sharir, *On the Complexity of Motion Planning for multiple Independent Objects; PSPACE-Herdness of the "Warehouseman's Problem",* Intern. J. Robotics Research **3**, (1984), 76–87.

[5] D. Jordan & M. Steiner, *Configuration Spaces of Mechanical Linkages,* Discrete Comput. Geom., **22**, (1999), 297–315.

[6] U. Kortenkamp, *Foundations of dynamic geometry,* PhD-thesis, ETH Zürich, 1999, `http://www.inf.fu-berlin.de/~kortenka/Papers/diss.pdf`.

[7] J.-M. Laborde, *Exploring non-euclidean geometry in a dynamic geometry environment like Cabri-géomètre*, Geometry Turned On (James King and Doris Schattschneider, eds.), MAA Notes, no. 41, The Mathematical Association of America, 1997, 185–191.

[8] N.E. Mnëv, *The universality theorems on the classification problem of configuration varieties and convex polytopes varieties,* in: Viro, O.Ya. (ed.): *Topology and Geometry — Rohlin Seminar*, Lecture Notes in Mathematics **1346**, Springer, Heidelberg 1988, 527–544.

[9] J. Richter-Gebert, *The Universality theorems for oriented matroids and polytopes,* Contemporary Mathematics **223**, (1999), 269–292.

[10] J. Richter-Gebert, *Realization Spaces of Polytopes,* Lecture Notes in Mathematics **1643**, Springer, Heidelberg 1996.

[11] J. Richter-Gebert & U. Kortenkamp, *Cinderella - The interactive geometry software*, Springer 1999; see also `http://www.cinderella.de`.

[12] J. Richter-Gebert & U. Kortenkamp, *Cinderella - Die interaktive Geometriesoftware*, HEUREKA Klett, 2000.

[13] J. Richter-Gebert & U. Kortenkamp, *Complexity issues in dynamic geometry*, in preparation, manuskript available on request.

[14] J. Reif, *Complexity of the movers' problem and generalizations,* Proc. 20th IEEE conf. on Foundations of Comp. Sci., Long beach, Calif.: IEEE Computer Society, 1979, 421–427.

[15] P. Shor, *Stretchability of pseudolines is NP–hard,* in: *Applied Geometry and Discrete Mathematics – The Victor Klee Festschrift* (P. Gritzmann, B. Sturmfels, eds.), DIMACS Series in Discrete Mathematics and Theoretical Computer Science, Amer. Math. Soc., Providence, RI, **4**, (1991), 531–554.