On using automated deduction techniques for loci computing in dynamic geometry environments.

Francisco Botana
e-mail: fbotana@uvigo.es
University of Vigo
Pontevedra, Spain

Abstract

This note lists some personal thoughts about the uses of automated deduction techniques in standard dynamic geometry environments. Examples of limitations of current approaches for dealing with loci in such environments are given, and an alternative for computing these objects is discussed, illustrating its strengths and drawbacks. The paper also deals with extending this approach to the 3D case.

1. Introduction

Defining dynamic geometry as models built by computer software that can be changed dynamically [10], Gao [6] lists five basic properties of dynamic models: dynamic transformation, dynamic measurement, free dragging, animation and locus generation. The last property deserves special attention in this paper since using automated deduction techniques locus objects can be integrated in dynamic geometry environments as the basic elements (points, lines, …) are. Section 2 sketches current ways for dealing with loci in standard environments and recalls a symbolic method using elimination for computing their equations.

2. Geometric Loci

Computing a locus in a standard environment involves two elements: the tracer and the driver objects. The tracer object depends on the driver one, and, in turn, the driver object must also lie on a predefined path. The strategy consists of sampling this path and computing the position of the tracer object for each sample item. In this way, the system gets a list of geometric objects that can be returned as the required locus (see [2] for an extended discussion on this subject). A simple and paradigmatic example is the construction of the ellipse through the gardener's method (see, for instance, http://nash.sip.ucm.es/LAD/LADucation4ggb/Example1.html).

Nevertheless, following this method we will get just half the ellipse in most environments. The reason behind getting half the ellipse is that, apart from the constraint of F being simultaneously on both circles, the system adds an extra constraint since it detects that there are two intersection points, despite the user idea of just constructing a point lying on both circles. This kind of ambiguities always arise whenever quadratic objects are used.

Another source of imprecision comes from the heuristics used when joining the positions of the locus tracer point. Most systems show the semi ellipse in Figure 1 as a continuous curve, obtained joining contiguous positions of the locus point when sampling the path of the driver element. Nevertheless, this strategy can lead to anomalous results for some border cases. For instance, we can consider tracing a conchoid of Nicomedes. Given a point O and another point P lying on a line, the conchoid of Nicomedes is the locus of points X such that X, O and P are collinear, and the distance between P and X is constant. A common answer searching for this locus is shown in Figure 2, but if O is placed very near of the line supporting P some systems will return an aberrant locus, as shown in Figure 3.

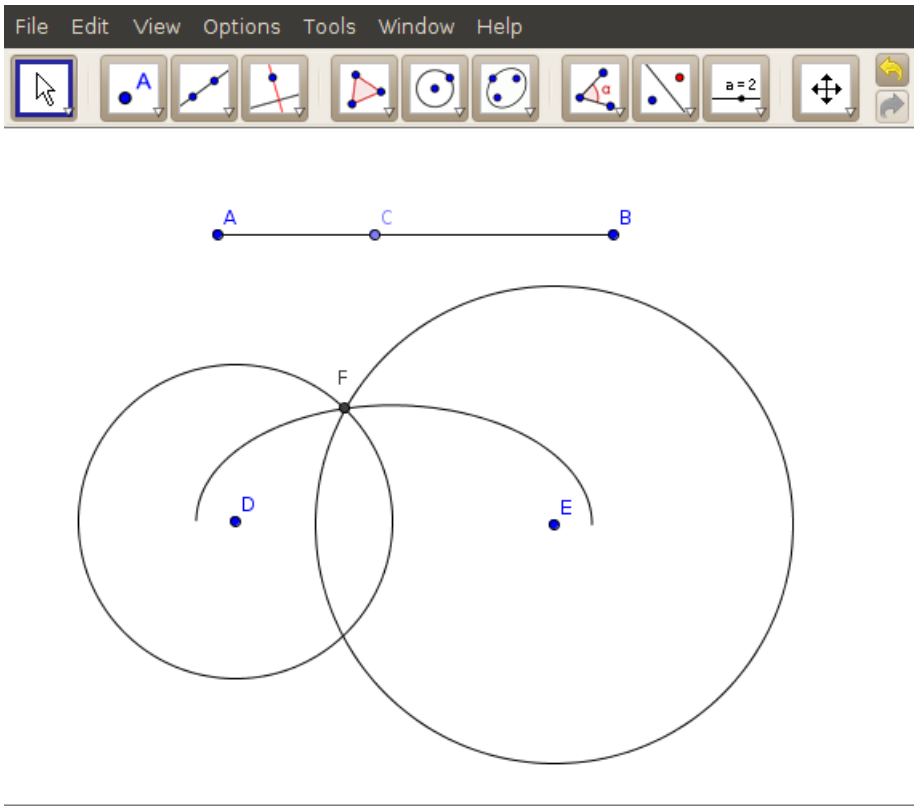

Figure 1: *A naif attempt to get an ellipse*

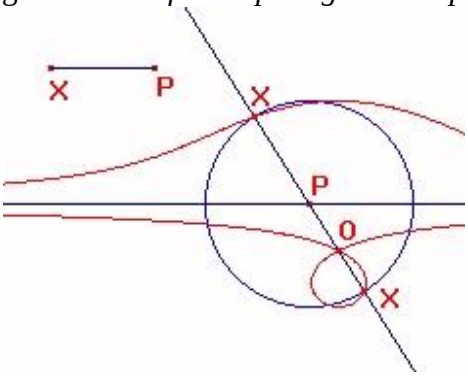

Figure 2: *An ordinary conchoid of Nicomedes*

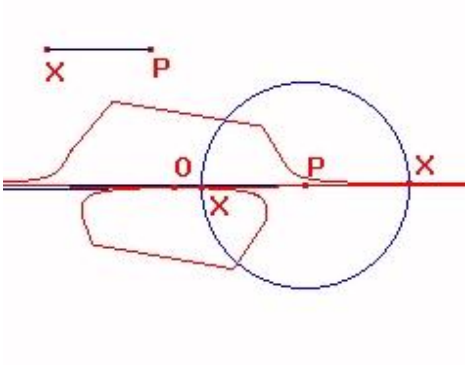

Figure 3: *The conchoid when O is very near of P support line*

A symbolic approach for computing the preceding locus uses a parametric description of the construction: since O is a free point we can assign it numeric coordinates (say (0,-1), for the sake of simplicity); similarly, $P(x_1,x_2)$ can be declared as a point on the x axis, that is, $x_2 = 0$. The point $X(u,v)$ lies on the line OP, so

$$(v+1) / u = 1 / x_1,$$

and on the circle centered in P and with radius $d = XP$,

$$(u-x_1)^2 + v^2 = d^2.$$

Eliminating x_1 and x_2 in the above three polynomial equations we get an implicit equation in u and v that describes the conchoid of Nicomedes (see [4] for a more technical specification). Note that the taken algebraic approach suppresses the ambiguity when choosing the intersections line-circle. Nevertheless, since we move on the realm of algebra, restricting P to a segment on the horizontal line would not modify the answer, whereas in such a case the locus shown in standard environments would be the corresponding part of a branch of the conchoid.

There exist various implementations of this approach. In [1] a remote add-on for dealing with Cabri, Cinderella, GeoGebra and The Geometer's Sketchpad constructions is described. Uploading a GeoGebra construction of the conchoid to http://nash.sip.ucm.es/LAD/LADucation4ggb/ the system returns a plot of the whole conchoid and its equation (Figure 4).

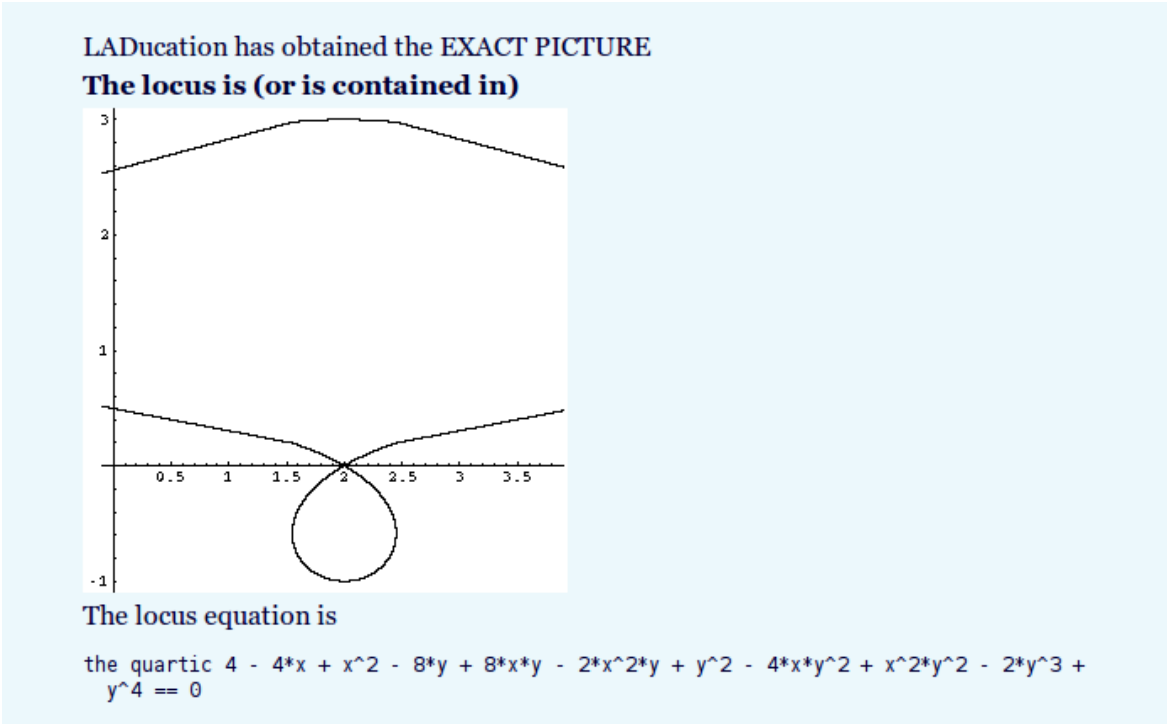

Figure 4: *A plot and the equation of a conchoid*

JSXGraph [9] is a JavaScript library for interactive geometry that also uses this symbolic approach for computing loci. Future versions of GeoGebra could also integrate this process (see [7] for ongoing work).

Although successful in computing the equations of loci, this symbolic method introduces new problems to solve. The first one involves degenerated conditions. A typical example is the limaçon of Pascal: given a circle with a fixed point *O* on it, and another circle point *P*, find the locus of points *Q* such that *O*, *P*, *Q* are aligned, and the distance between *P* and *Q* is constant. Figure 5 shows the locus returned by Cinderella and the one obtained via the symbolic method. The locus equations contain an extraneous factor, a circle. Although from a strict understanding of the locus conditions, it could be admissible (since for *O=P* the collinearity condition is undefined, only remaining the second condition, so giving the circle), this unexpected result can be confusing for users. Currently, there is no known solution to reject these spurious results in an automated and general way.

The second open problem is related with the magnitude of returned equations. Since accuracy is a goal in dynamic environments, it is usual to work internally with numbers of high precision. Since symbolic methods work in exact arithmetic, the user can find unmanageable expressions.

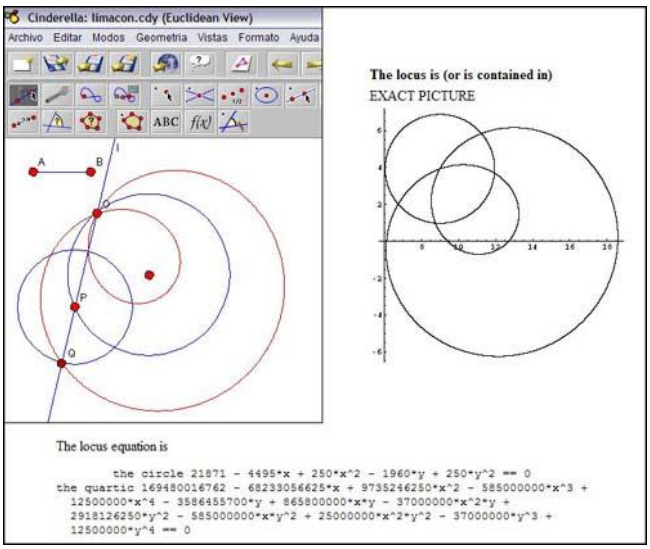

Figure 5: *A limaçon of Pascal*

Apart from the problems sketched above, there are constructions where the elimination strategy will fail due to its computational cost. The path described by the feet of a Jensen linkage [8], shown in Figure 6, can be easily traced in standard environments. Nevertheless, no path equation is returned in reasonable time since the involved elimination is a really heavy task. The above cited system LADucation4ggb uses an elimination procedure based in Groebner bases, so having a doubly exponential cost in the worst case. An approach using the method of Wu [11] seems to be more promising, given its lower computational cost. Nevertheless, this alternative approach should be balanced due to its stronger requirements for general applicability to geometric constructions, if having in mind an automatic procedure.

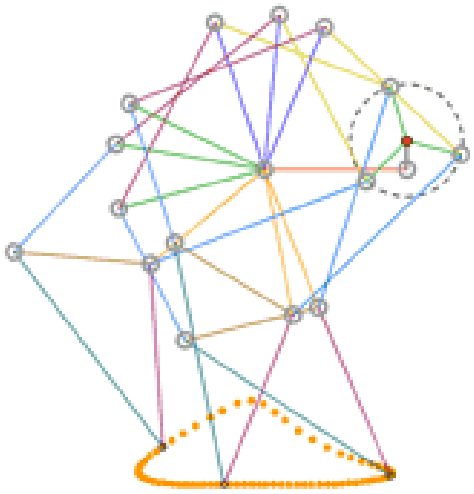

Figure 6: *A Jansen linkage*

Extending this symbolic approach to the 3D case does not introduce any significant new problem, apart from the above mentioned, undetected degenerated conditions and magnitude of returned equations. In [3] it is reported a system that computes the locus equations of 3D constructions specified in an ad-hoc grammar. For the sake of illustration, we study a simple generalization of the limaçon of Pascal to the spatial case: Given a sphere with a fixed point Pt_2 on it, and another sphere point Pt_4, find the locus of points Pt_8 such that Pt_2, Pt_4, Pt_8 are aligned, and the distance between Pt_4 and Pt_8 is constant. A textual description of this construction is

Range[-2.7,2.8,-4.4,2.8,-2.7,2.7];

FreePointD[Pt1,1,1,0];

FreePointD[Pt2,-31/49,16/9,0];

SphereD[Sp3,Pt1,Pt2];

PointOnSphere[Pt4,Sp3];

LineD[Ln5,Pt2,Pt4];

SphereD[Sp6,Pt4,Pt1];

IntersectionLineSphere[Pt8,Sp6,Ln5];

Locus[O10,Pt8,Pt4];

where the first line specifies the drawing box for the locus, and the remaining lines are self-explicative. Uploading this text to a server, the geometric predicates are translated into polynomials, an assignment of symbolic coordinates is performed, and, finally, after variable elimination, the user gets the answer as shown in Figure 7. Note that the system factors the locus in two components, namely a sphere

$$441\,x^2 + 558\,x + 441\,y^2 + 441\,z^2 - 1568\,y + 128\,z = 0,$$

corresponding, as in the standard limaçon, to a degenerated case ($Pt_2 = Pt_4$), and the quartic surface

85766121 *x*^4 + 171532242 $x^2\,y^2$ + 85766121 *y*^4 + 171532242 $x^2\,z^2$ + 171532242 $y^2\,z^2$ + 85766121 *z*^4 – 343064484 x^3 – 343064484 $x^2\,y$ – 343064484 $x\,y^2$ – 343064484 y^3 – 343064484 $x\,z^2$ – 343064484 $y\,z^2$ – 155363859 x^2 + 686128968 $x\,y$ – 155363859 y^2 – 498428343 z^2 + 80946126 x + 1433186300 y – 860340224 = 0.

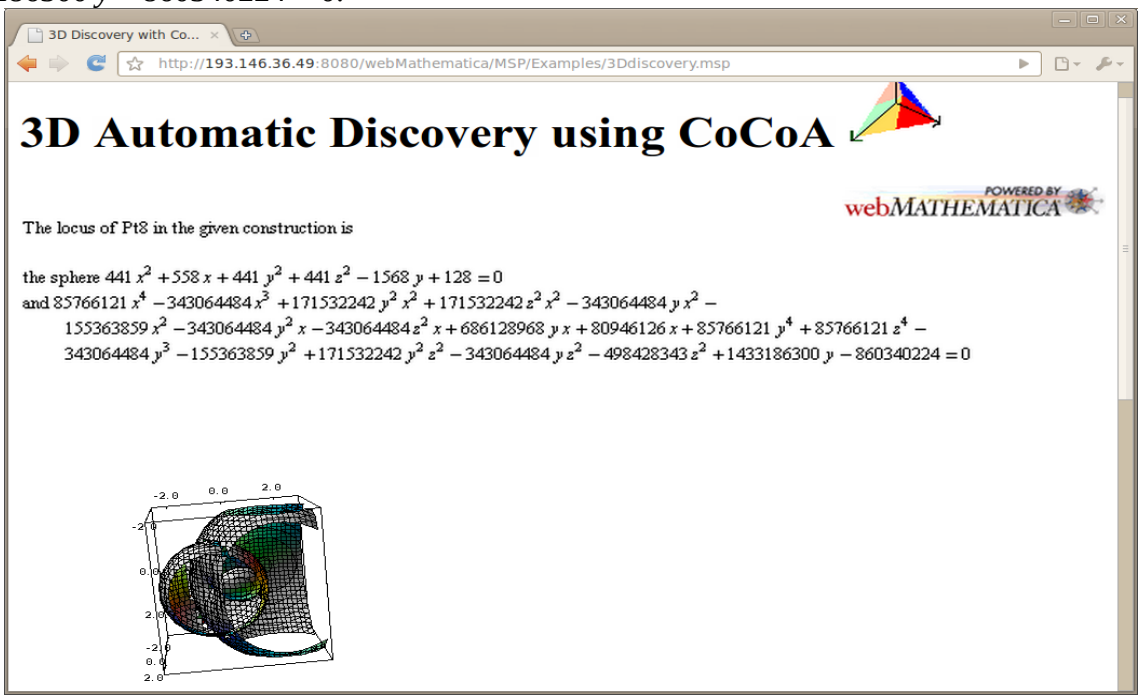

Figure 7: *A 3D limaçon of Pascal*

3. Conclusion

Some problems with the traditional approach for computing loci in standard dynamic environments are reviewed, and a symbolic method using polynomial elimination is used to illustrate how to compute the loci equations. The knowledge of these equations places loci objects to the same level that of basic elements. Nevertheless, incorporating this method in the internals of dynamic geometry software introduces new problems, such as dealing with degenerated factors and long expressions, which deserve special consideration.

Acknowledgements

The author was partially supported by research grant MTM2008-04699-C03-03/MTM from the Spanish MEC.

References

1. Abánades, M., Escribano, J., and Botana, F., *Remote symbolic computation of loci*, The International Journal of Technology in Mathematics Education, 17(3), 2010.

2. Botana, F., *Interactive versus symbolic approaches to plane loci generation in dynamic geometry environments*, Lecture Notes in Computer Science, 2330, Springer, 2002.
3. Botana, F., *Automatic determination of algebraic surfaces as loci of points*, Lecture Notes in Computer Science, 2657, Springer, 2003
4. Botana, F., and Valcarce, J., *A software tool for the investigation of plane loci*, Mathematics and Computers in Simulation, 61(2), 2003.
5. Botana, F., and Valcarce, J., *Automatic determination of envelopes and other derived curves within a graphic environment*, Mathematics and Computers in Simulation, 67(1-2), 2004.
6. Gao, X.-S., Automated geometry diagram construction and engineering geometry, Lecture Notes in Artificial Science, 1669, Springer, 1999.
7. GeoGebra Locus Line Equation, http://www.geogebra.org/trac/wiki/LocusLineEquation
8. Jansen, T., http://en.wikipedia.org/wiki/Theo_Jansen
9. JSXGraph, http://jsxgraph.uni-bayreuth.de/wp/
10. King, J., and Schattschneider, D., *Geometry Turned On*, The Mathematical Association of America, 1997.
11. Wu, W.-T., *Mechanical theorem proving in geometries*, Springer, 1994.