Easy (but exact) study of caustics of conics

Zoltán Kovács
zoltan@geogebra.org
The Priv. Univ. Coll. of Educ. of the Diocese of Linz
Salesianumweg 3
4020 Linz, Austria

Abstract

We give a concise summary on the technical history of visualizing implicit locus equations in the widely used dynamic geometry system GeoGebra. We mainly focus on the "draggable locus" equation, introduced in GeoGebra 5 (2014), and the dynamic envelopes. We discuss how user interface improvements (the LocusEquation and Envelope tools in GeoGebra Discovery), backend updates (the Giac subsystem) and algebraic experiments (to automatically exclude degeneracy) lead to easy study of caustics of a circle or a parabola in the classroom.

1 Introduction

At CADGME 2014 (Halle, Germany), Francisco Botana and the author presented a novel way of visualizing locus and envelope curves. The "draggable locus" feature was introduced in GeoGebra 5 – the authors showed how the **LocusEquation** and **Envelope** commands can be used to bring algebraic curves closer to the students.

As a closing remark in that past presentation, Sander Wildeman's remark was cited: "Once you have written an article about caustics you start to see them everywhere." Indeed – and this fact encouraged us to do further developments on the topic! Since 2014 we continuously enhance our approaches to support the study of locus and envelope equations for classroom situations as easily as possible. Our work consisted of

- user interface improvements (like the availability of the above commands as toolbar icons),
- backend updates (to speed up the underlying elimination process in the background)
- and algebraic experiments (to find the best suitable formula to express the tangents of conics in general, to obtain fast computational results).

Our research led finally to make it possible to obtain non-trivial results in a user friendly way. As an example of these improvements, we are going to study the caustics of a circle or a parabola in the second part of this paper. Despite our results are not new from the mathematical point of view, they are novel from the educational approach.

2 A brief history of "draggable loci"

We recall some of the milestones of the long road to achieve difficult algebro-geometric results in a simple way. The very beginnings include Reinhard Oldenburg's *Feli-X* and *FeliX* systems [15, 16]. Their first version was available in March 2004. It provided a simple user interface to work with planar geometric constructions. They could be expressed by direct algebraic inputs like the command `addEquation[yC[A]==0]` or geometric declarations like `B=addObject["point",0,4]`. The output as an algebraic equation was provided algebraically and also graphically. Feli-X used *Mathematica* as its backend but Oldenburg stated clearly that several schools could not afford Mathematica licenses – therefore a free alternative computer algebra system (CAS) should be considered for the long term.

In fact, an earlier work [5] (January 2003) published by Francisco Botana and José Luis Valcarce already described a similar system. The software tool *Lugares* was able to use a free CAS as its main backend: *CoCoA*. A later publication [6] by these authors (July 2004) described the software tool *GDI* which could compute and plot the envelope equation of a family of lines parametrized by the positions of a point on a geometric object, by providing a simple user interface.

Botana continued to work on making these concepts available for a wider audience. As of 2010 it turned out that the freely available *GeoGebra* system becomes very popular among end users in educational communities. Hence he published the *LADucation* system at CADGME 2010 (Hluboká nad Vltavou, Czech Republic[1]) – it was able to import GeoGebra files and manipulate them via a web service. LADucation internally used a CoCoA installation on the server.

A collaboration between a research group in algebraic geometry, led by Tomás Recio, and the GeoGebra Team, led by Markus Hohenwarter, seemed to be a fruitful cooperation at this point. Botana's concept on outsourcing the heavy computations to an external system remained an important part of the project. The first publicly available version was implemented by a student Sergio Arbeo in the frame of the Google Summer of Code (GSoC) 2010, advised by Miguel Á. Abánades, by using Heinz Kredel's Java Algebra System (*JAS*) on the top of GeoGebra 4.0. This first version already provided the **LocusEquation** command. It heavily relied on another GSoC project that aimed at contour and implicit plotting, programmed by Philipp Birklbauer and advised by Darko Drakulic, in the same year.

The first implementation had some difficulties with speed. It turned out that the implementation of Gröbner basis computations is the main bottleneck. Finally, several changes in GeoGebra's CAS backend (Reduce in 4.2, 2011; SingularWS [3] and Giac in 4.4, 2013; Giac in 5.0 [10], 2014) and some substantial speedups of Giac's internal command `eliminate` led to major visual improvements that made fast animations possible when the inputs were changed via a dragging move [9].

Meanwhile the **Envelope** command was added to GeoGebra 5.0 by Botana and the author [4]. It heavily exploited Giac's fast elimination algorithm. Also, a general geometric prover subsystem was developed by Simon Weitzhofer and the author in 2011, and supported by some other researchers in the following years [2]. To avoid implementing similar ideas multiple times, the commands **LocusEquation**, **Envelope** and **Prove** were unified in 2017 by the author. As a result, GeoGebra was extended with a general geometric prover subsystem called *GeoGebra Automated Reasoning Tools* [12]. This subsystem was thoroughly tested, documented and further improved by additional re-

[1]The slides for Botana's talk are available at http://home.pf.jcu.cz/~cadgme2010/proceedings/botana/Francisco_Botana_talk.pdf.

searchers, among others: M. Pilar Vélez, Pavel Pech, Roman Hašek and Thierry Dana-Picard. This fruitful collaboration among several researchers yielded to further improvements like

- automated removal of some degenerated components ([11], 2019),
- toolbar access of the **LocusEquation** and **Envelope** commands (2020),
- attaching points to arbitrary algebraic curves that are defined by the user via a polynomial equation ([7], 2020).

In fact, some of the latest improvements are not yet official parts of GeoGebra. Instead, an experimental version called *GeoGebra Discovery*[2] is launched to make it possible for the researchers to try out these novel techniques in a preliminary version that can be considered "unstable" for the every day use. After being stable enough and confirmed by the GeoGebra Team, some of these new features may be supported officially by the mainstream version of GeoGebra as well.

Now in 2022, the locus equation and envelope algorithms are ready to play a substantial role in helping students to experiment with algebraic curves, up to eventually a degree of 20 for the studied polynomials. In the second part of the paper some simple ways will be presented on how to get the caustics of a circle or a parabola. This can be fruitful not just for mathematics classes but for STEAM projects (for example, with optical applications in physics). Also, the underlying mathematical and technological details will be explained at some extent. Finally, as future plan, the easy study of caustics of an ellipse or hyperbola will be mentioned – even if these tasks are too difficult yet for the current stage of the project.

3 Locus and its equation

The "draggable locus" feature appeared in certain software systems after the millecentenary. It is based on locus *equations*, and it therefore has a *symbolic* property. On the other hand, the appearance of geometric locus in computer programs is an older feature, and it has a *numerical* property from the computational view.

The first exhaustive work on presenting the benefits of geometric loci is perhaps the doctoral dissertation [8] by Ulrich Kortenkamp. Its chapter 10.4 "Exploring Geometry with Loci" lists several applications of drawing a geometric locus. In a later work [18] by Kortenkamp and Jürgen Richter-Gebert we find the following *definition of a locus*:

> A locus is the trace of a point under the movement of another point. A locus is defined by three objects:
>
> - The *mover*, a free element whose movement drives the generation of the locus.
> - The *road*, an element incident to the mover. The mover will be moved along the road.
> - The *tracer*, the element whose trace is calculated and presented as a geometric locus.

[2]Available at https://github.com/kovzol/geogebra-discovery

In some software systems (like GeoGebra) the road is called *path*.

When computing the equation of a locus, GeoGebra uses computational algebraic geometry to manipulate the equations that describe the coordinates of the points in the figure. The obtained output is always an algebraic curve (or the union of discrete points) which is computed by—roughly speaking—eliminating some variables in the algebraic translation of the geometric construction.

The process to get the locus equation is performed by denoting the locus point by (x, y), and the other points of the construction by (v_i, v_{i+1}) for some i, and the relationships among the points are described by polynomial equations $f_j(v_1, v_2, \ldots, v_n, x, y) = 0$. Now the goal is to eliminate all variables but x and y from the ideal $\langle f_j \rangle$.

In fact, similar steps can be used to compute an envelope of a family of curves. We start with the same procedure, but we define the locus point virtually by putting it on the object which is tangent to the envelope. Now by computing the determinant of the Jacobian matrix

$$D = \begin{vmatrix} \frac{\partial f_1}{\partial v_1} & \frac{\partial f_2}{\partial v_1} & \cdots & \frac{\partial f_n}{\partial v_1} \\ \frac{\partial f_1}{\partial v_2} & \frac{\partial f_2}{\partial v_2} & \cdots & \frac{\partial f_n}{\partial v_2} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial f_1}{\partial v_n} & \frac{\partial f_2}{\partial v_n} & \cdots & \frac{\partial f_n}{\partial v_n} \end{vmatrix}$$

an extra polynomial $f_{n+1} = D$ will be obtained. As a final step, we eliminate all variables but x and y from the ideal $\langle f_j \rangle$.

GeoGebra (since 2016) also supports the computation of *implicit locus* curves [1]. This means the algebraic discovery of the path curve which is prescribed by some expected geometric property of the figure. In this case the sought mover point is denoted by (x, y), and the other points of the construction by $(v_i, v_i + 1)$. The relationships among the points are described by the polynomial equations $f_j = 0$: they are the *hypotheses*. The algebraic translation of the assumed implicit condition (the *thesis*) is also added to the polynomials. Now by eliminating all variables but x and y from the ideal $\langle f_j \rangle$ the implicit locus will be obtained.

In GeoGebra, technically, dynamic plotting of a locus or envelope equation is performed as a mixture of symbolic and numerical computations that are shown in Figure 1. The computationally heavy part is to eliminate the variables from the ideal obtained during the algebraization process. If the user drags one of the free points during the use of a dynamic applet, a number of consecutive eliminations must be performed. When using an efficient algorithm, it is possible to achieve a high number of symbolic computations within a second. This leads to a continuous movement on the screen – unless the input construction is too complicated and it results in a heavy computation. We recall that elimination can be in worst case doubly exponential in the number of variables [14] because the underlying algorithm uses Gröbner bases in the background.

Finally, here we refer to the paper [13] by Peter Lebmeir and Richter-Gebert that proposes a different way to compute the locus equation. This approach starts with a collection of sample points that are graphically constructed when moving the mover point. The algorithm then looks for a curve, that goes through the sample points, of bounded degree, to establish a rigorous bound on the degree of the locus equation. This sophisticated method, however, does not seem to be widespread in practice.

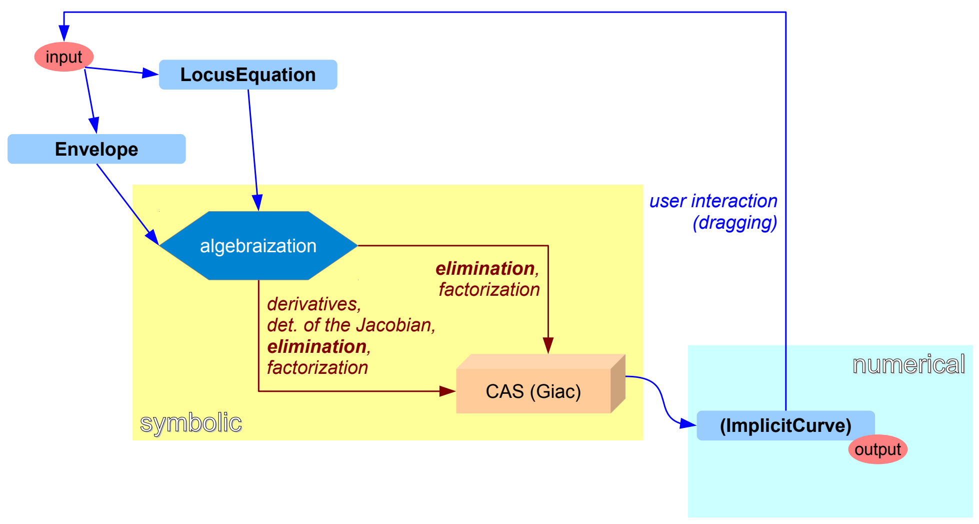


Figure 1: Workflow of plotting a locus or envelope equation in GeoGebra

4 Caustics of a conic

4.1 The case of the circle

According to `mathcurve.com`, the caustics by reflection of circles are the envelopes of the reflection of rays, emitted by a light source placed at finite or infinite distance, by a circle. It is well known how to classify these curves:

- in the finite cases we usually get a sextic curve,
  - if the light source is outside the circle, then it is a nephroid curve, and
  - inside a circle it is still a sextic but it looks differently (Fig. 2).
  - If the light source is on the circle, we get a cardioid of degree 4 (Fig. 3).
- Otherwise, if the light source is placed at infinite distance, we get a nephroid again (Fig. 4).

We emphasize that the method we use in GeoGebra differs from the usual parametric way. We use just elimination to get the algebraic output before the plotting step. This explains why the case of the cardioid results in an extra degenerate linear component that is tangent to the circle and the cardioid at the point of light source (Fig. 3).

Figures 2, 3 and 4 use similar notations which will be summarized here quickly. The circle is denoted by c and defined with midpoint M and circumpoint C. The light source S is explicitly given in Figures 2 and 3, but it is created dynamically in Figure 4 by setting up the direction of the light first via a vector $\vec{v} = \overrightarrow{OV}$. In each figure we use a tangent point T that is attached to circle c and designates the mirror m accordingly. In Figure 4 we create point S to have $\overrightarrow{ST} = \vec{v}$. Now we can get point S' as the reflection of S about m in all three setups. By reflecting S' about T we get the technical point S'' so that it is possible to graphically present the segment ST and then the ray TS'' which together show the beam of the light.

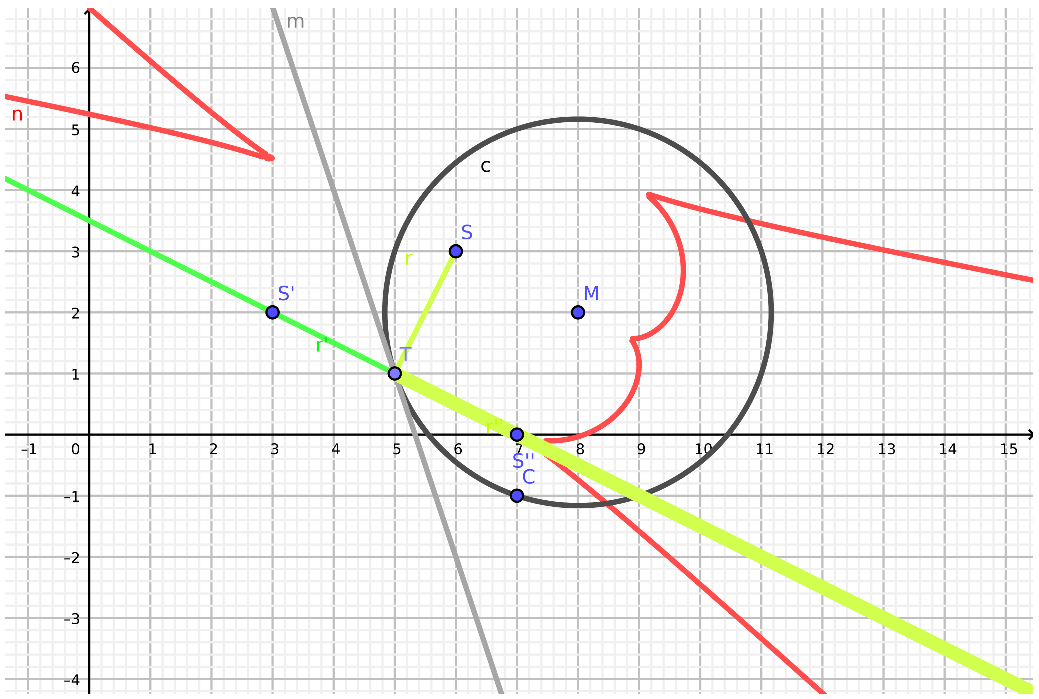


Figure 2: A caustic of a circle, the light source is placed inside

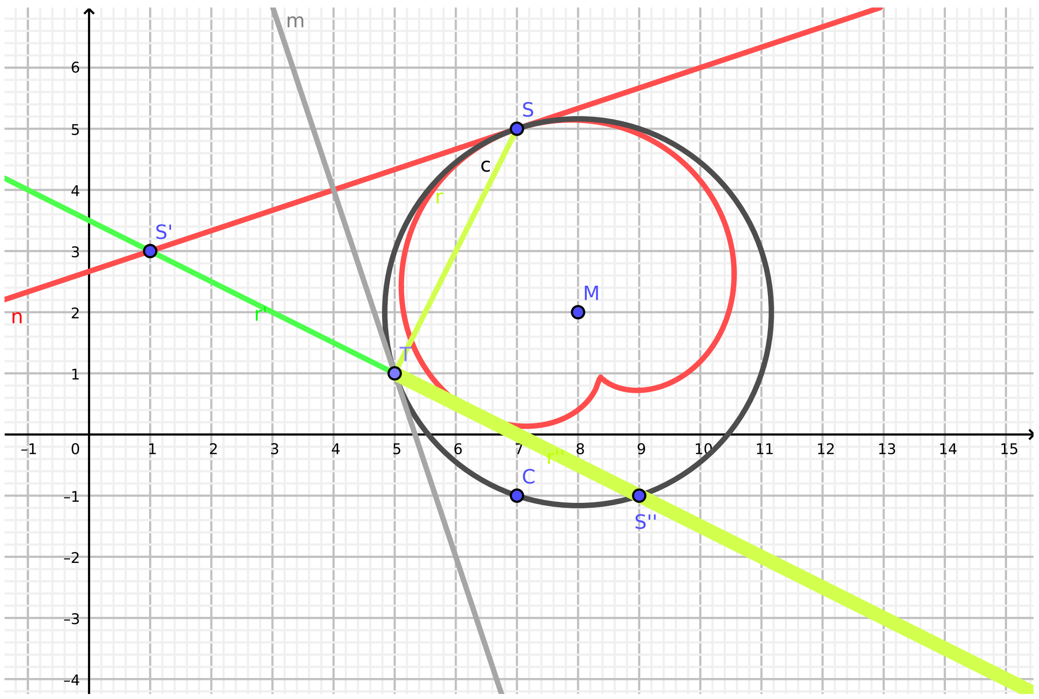


Figure 3: A caustic of a circle, the light source is placed on the circle

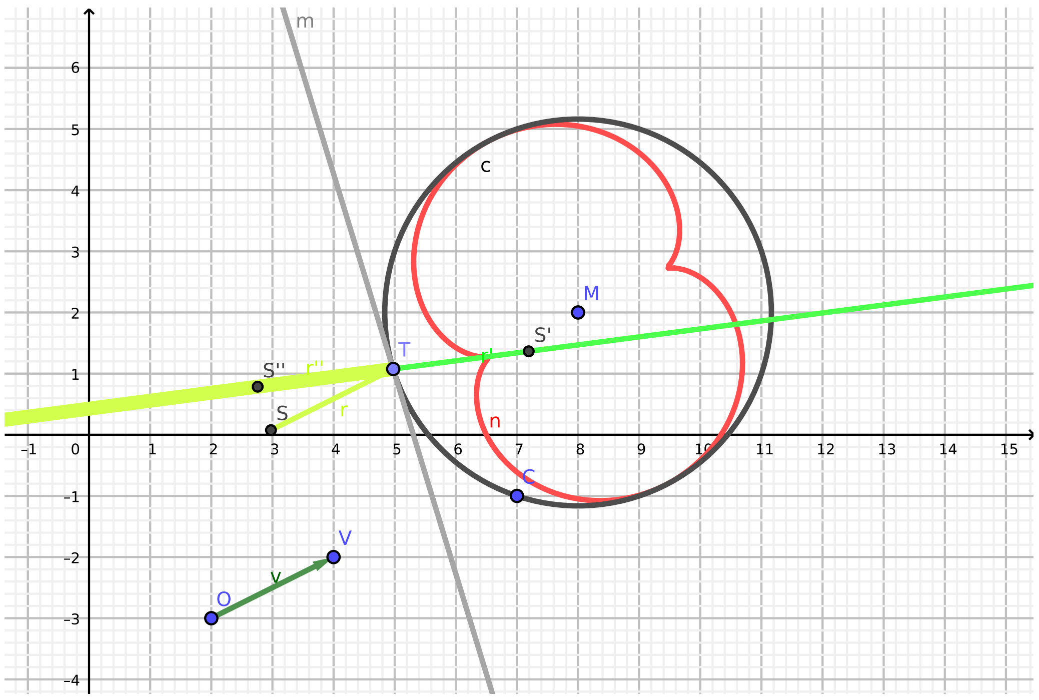


Figure 4: A caustic of a circle, the light source is placed at infinite distance

In the mathematical sense we simply use the family of lines (or rays[3]) TS' (which is the same as the family of lines $S''T$) and compute the envelope n tangent to these lines. This makes us to compute and plot the caustics of a circle in a very simple way in GeoGebra, see Table 2. In fact, only steps 1–9 are required to plot the caustic, the remaining steps are just explaining the optical details. Step 9 as a toolbar icon is supported only in GeoGebra Discovery – the corresponding *command* must be used in the mainstream version of GeoGebra. We highlight that the whole construction can be built by using only the mouse – no keyboard input is required!

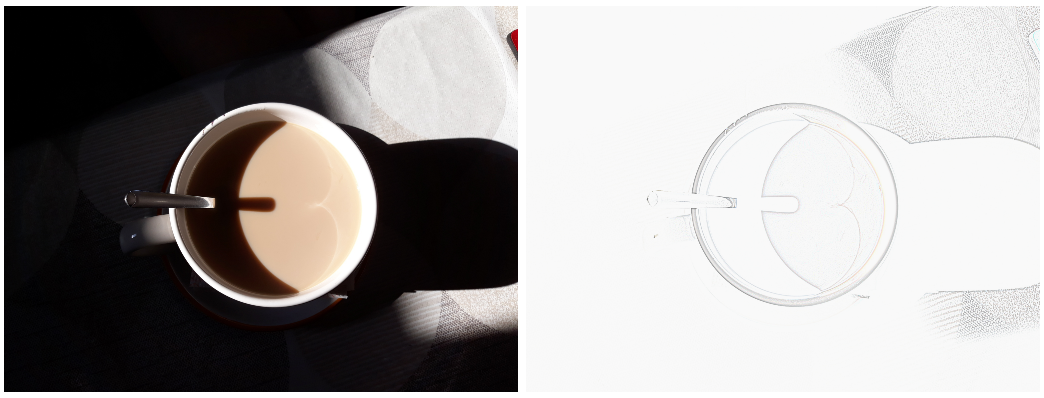
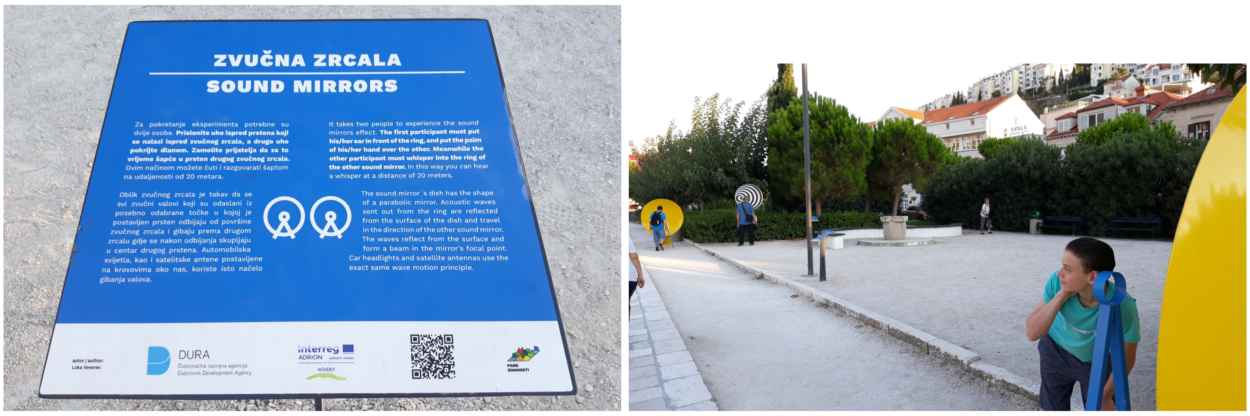


Figure 5: A caustic of a circle in a cup of coffee – the right figure emphasizes the appearing curves after an automatic edge-detection

No.	Name	Toolbar Icon	Description
1	Point M	•A	
2	Point C	•A	
3	Circle c	⊙	Circle through C with center M
4	Point S	•A	
5	Point T	•A	Point on c
6	Line m	⟋○	Tangent to c through T
7	Point S'	⟍	S mirrored at m
8	Ray r'	⟋	Ray through T, S'
9	Implicit Curve n	⤫	Envelope(r', T)
10	Segment r	⟋	Segment S, T
11	Point S''	⋰	S' mirrored at T
12	Ray r''	⟋	Ray through T, S''

Table 2: Easy construction of caustics of a circle, the light source is at finite point



ZVUČNA ZRCALA

SOUND MIRRORS

Za pokretanje eksperimenta potrebne su dvije osobe. **Prislonite uho ispred prstena koji se nalazi ispred zvučnog zrcala, a drugo uho pokrijte dlanom. Zamolite prijatelja da za to vrijeme šapće u prsten drugog zvučnog zrcala.** Ovim načinom možete čuti i razgovarati šaptom na udaljenosti od 20 metara.

Oblik zvučnog zrcala je takav da se svi zvučni valovi koji su odaslani iz posebno odabrane točke u kojoj je postavljen prsten odbijaju od površine zvučnog zrcala i gibaju prema drugom zrcalu gdje se nakon odbijanja skupljaju u centar drugog prstena. Automobilska svijetla, kao i satelitske antene postavljene na krovovima oko nas, koriste isto načelo gibanja valova.

It takes two people to experience the sound mirrors effect. **The first participant must put his/her ear in front of the ring, and put the palm of his/her hand over the other. Meanwhile the other participant must whisper into the ring of the other sound mirror.** In this way you can hear a whisper at a distance of 20 meters.

The sound mirror´s dish has the shape of a parabolic mirror. Acoustic waves sent out from the ring are reflected from the surface of the dish and travel in the direction of the other sound mirror. The waves reflect from the surface and form a beam in the mirror's focal point. Car headlights and satellite antennas use the exact same wave motion principle.

autor / author: Luka Veverec

DURA
Dubrovačka razvojna agencija
Dubrovnik Development Agency

Interreg ADRION ADRIATIC-IONIAN
WONDER

PARK ZNANOSTI

Figure 6: An outdoor activity in Dubrovnik, Croatia, demonstrating sound transmission via two parabolic mirrors: the mirrors are made of yellow metal disks and their foci are highlighted with blue circular rings

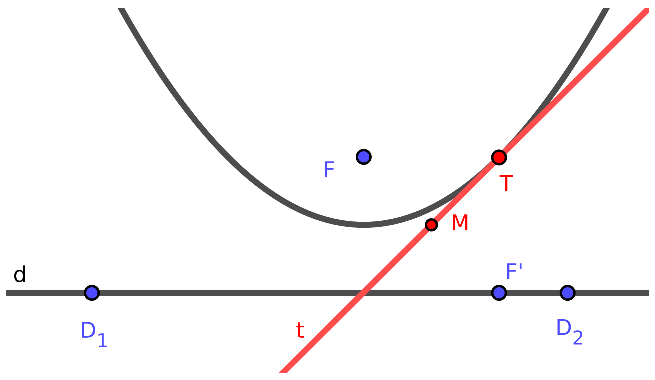


Figure 7: Geometric setup for the tangent of a parabola

Application of the obtained results is very common in every day life. Objects that have a circular cross-section can usually be observed that they behave as mirroring surfaces. Inside them, eventually on the bottom of a cup or a bucket, or in a liquid material (maybe in the water that fills the bucket), the caustic can be seen, sometimes very sharply. Figure 5 shows one photo taken recently by the author – the half of a nephroid curve can be detected easily.

We omit the equation system that was used to create the envelope curves. We emphasize, on the other hand, that a smooth animation of the curves can be achieved on a recent PC: on an Intel(R) Core(TM) i7-4770 CPU @ 3.40GHz (Ubuntu Linux 22.04) we managed to get 14.3 frames per second (FPS) for the finite point case and 11.5 FPS for the infinite one.

4.2 The case of the parabola

It is well known that in the case of the parabola we can have a unique behavior of reflection when the light source is put in the focus. In this case the reflected rays are all parallel lines. An important application of this fact is data transmission (or broadcasting) via parabolic mirrors: the sender (and/or the receiver) needs to be in the focus of a parabola so that the direction of transmission will be orthogonal to the directrix (Fig. 6).

Now we continue with a general position of the light source (Fig. 7). We assume that the parabola is given with focus F and directrix d (which is defined with line points D_1 and D_2). Let T be a tangent point of the parabola. We are searching for a point M such that MT is the tangent *line* of the parabola.

We give a list of geometric properties that should be fulfilled to describe the setup geometrically:

1. D_1, D_2 and F' are collinear.
2. $F'T$ and D_1D_2 are orthogonal.
3. $FT = F'T$.
4. M is the midpoint of FF'.

[3]Lines are *algebraic closures* of rays. In *algebraic geometry* we deal with polynomial equations, so we cannot distinguish between a line or its infinite subsets. Similarly, when removing a point from a conic, we cannot distinguish between the full conic and the object we obtain after removing that point.

5. F, F' and T should not be collinear.

After several attempts, the following equations seem to be a fruitful list to describe the setup algebraically:

$$\det\begin{pmatrix} d_{1x} & d_{1y} & 1 \\ d_{2x} & d_{2y} & 1 \\ f'_x & f'_y & 1 \end{pmatrix} = 0, \tag{1}$$

$$(f'_x - t_x)(d_{1x} - d_{2x}) + (f'_y - t_y)(d_{1y} - d_{2y}) = 0, \tag{2}$$

$$(f_x - t_x)^2 + (f_y - t_y)^2 = (f'_x - t_x)^2 + (f'_y - t_y)^2, \tag{3}$$

$$m_x = \frac{f_x + f'_x}{2}, m_y = \frac{f_y + f'_y}{2}, \tag{4}$$

$$t \cdot \det\begin{pmatrix} f_x & f_y & 1 \\ f'_x & f'_y & 1 \\ t_x & t_y & 1 \end{pmatrix} = 1. \tag{5}$$

Here (4) consists of two equations for the two coordinates of M. Also, in (5) we deny the equality by introducing a new variable: this technique is well-known in algebraic geometry to express certain inequalities (it is also known as *Rabinowitsch's trick* [17]).

We highlight that this set of equations is not the only way to describe the setup geometrically or algebraically. But this seems to be a useful way, because it allows us to obtain the envelope curve when the tangent line MT is changing dynamically. In fact, the most important equation to explain is (5): if we omit it, we allow T to be the vertex of the parabola, and this is to be avoided, because this would imply $M = T$ and no line MT would be exactly defined: each line going through the vertex would be allowed. Therefore, when (5) is not explicitly stated, an infinite amount of lines would be introduced and this would prevent us from getting the correct envelope equation! On the other hand, even if we exclude the presence of the vertex in the study, we still get the correct envelope, because it corresponds to an algebraically closed set. In other words: by considering the input without the vertex, we get a geometric output that may lack of some points, but the algebraic closure of that output is still the correct, full envelope.

Figures 8, 9 and 10 show the rich geometry of different cases when the light source is

- inside the parabola,
- on it, or
- outside it.

In the first and third cases we get a sextic curve, while in the second one a quartic curve will be obtained (together with an extra degenerate linear component that is tangent to the parabola and the quartic).

In the case of infinite distance we get that the output is cubic – unless the direction of parallel lines is orthogonal to the directrix (in this case we obtain the focus F as a single point of the envelope, as expected). See Figures 11 and 12. In fact, we obtain a Tschirnhausen cubic in the general case [19].

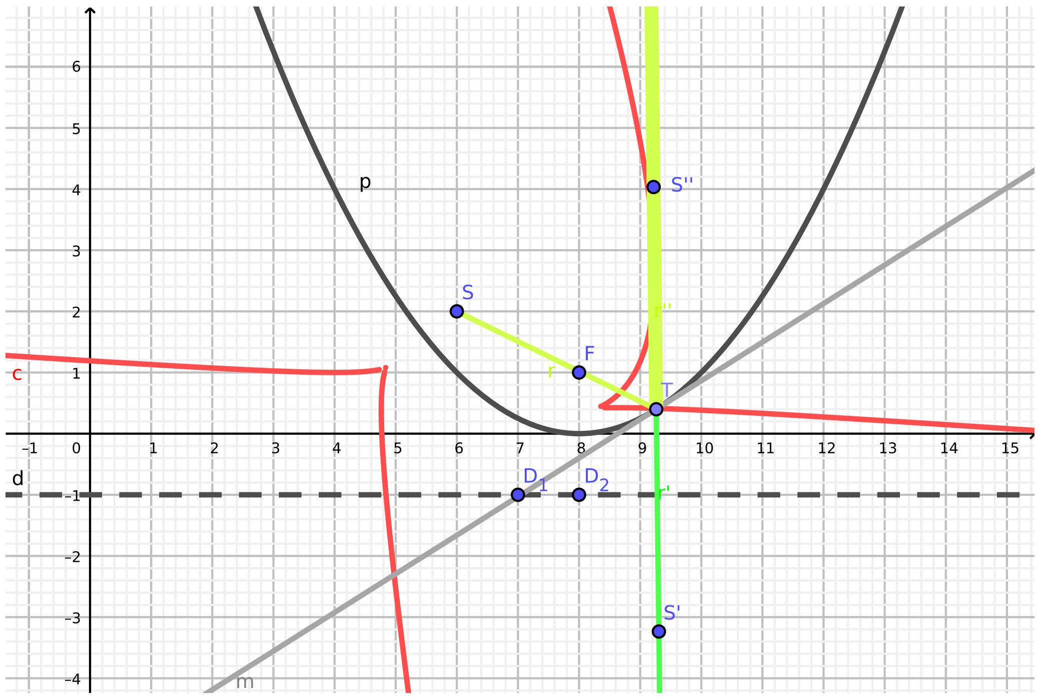


Figure 8: A caustic of a parabola, the light source is placed inside

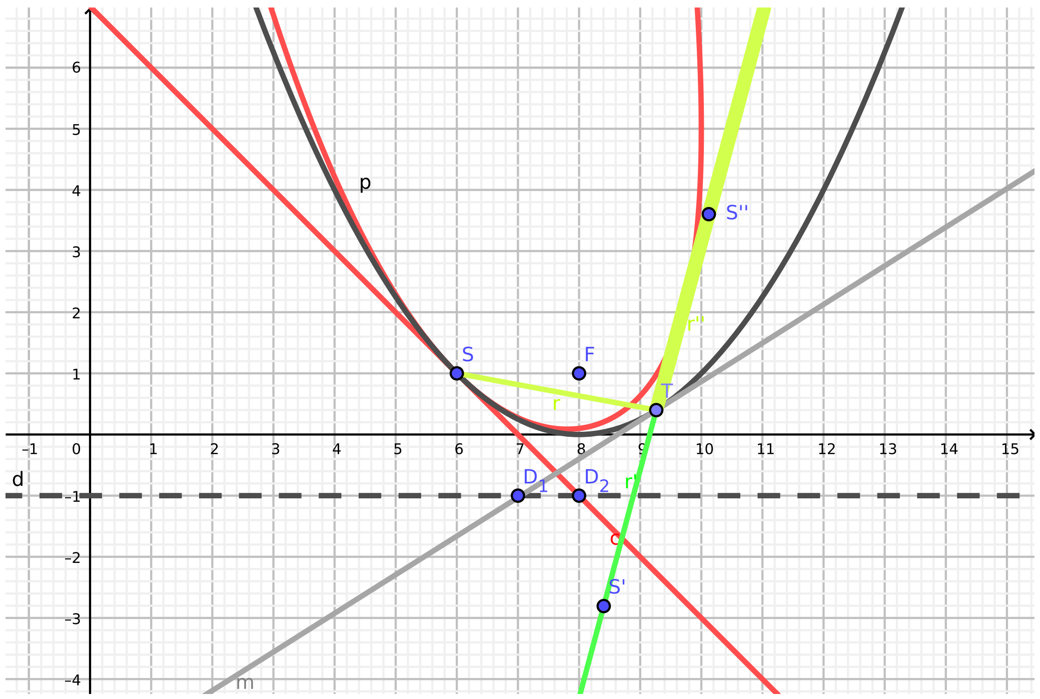


Figure 9: A caustic of a parabola, the light source is placed on the parabola

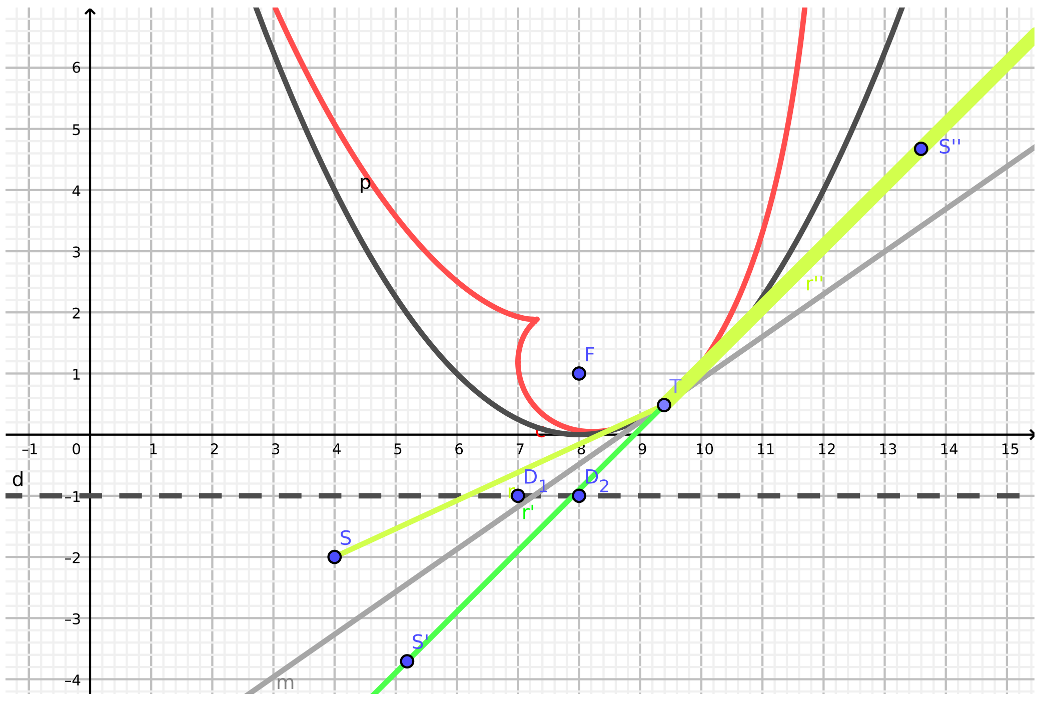


Figure 10: A caustic of a parabola, the light source is placed outside

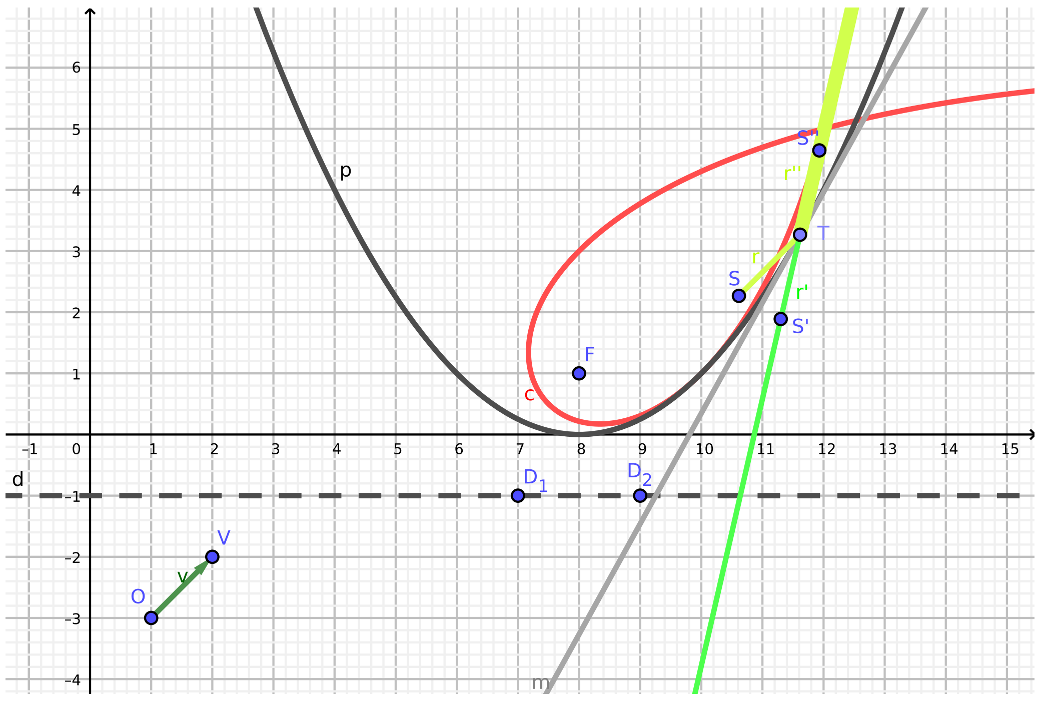


Figure 11: A caustic of a parabola, the light source is at infinite distance, general case

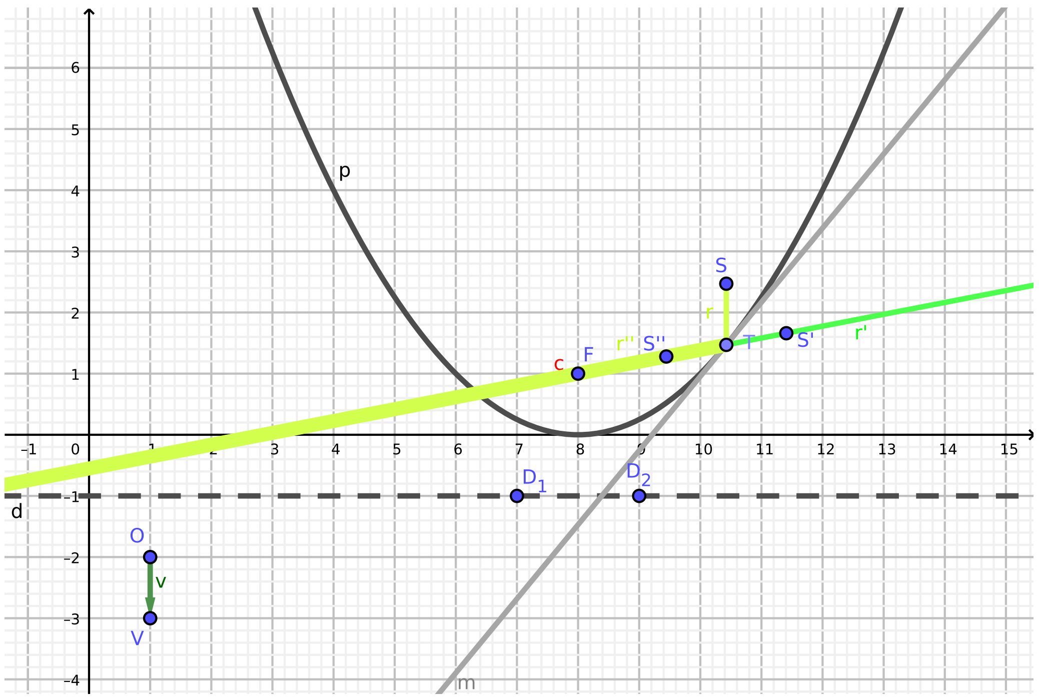


Figure 12: A caustic of a parabola, the light source is at infinite distance, "orthogonal" case

Finally we provide here the construction steps how the last two cases were built by using GeoGebra Discovery. In Table 4 a list of 17 steps are given (in fact, the last three steps are just explaining the optical details). Only one of them (step 10) requires the use of the keyboard. This only step can be worked around by using a line o (given with free points O_1 and O_2) and attaching the point S to it, then projecting S orthogonally to the parabola to get the tangent point T (see Figure 13) – in this case, however, the output contains *two* degenerate linear components and the animation is somewhat slower (see Table 5 for a speed comparison).

4.3 The case of the ellipse and the hyperbola

Table 5 already prognosed the bad news about the case of ellipses and hyperbolas. Indeed, an algebraic geometry approach was in our research work not yet possible because of the too high amount of computations.

We give a brief overview about how we tried to handle the difficulties. First of all, ellipses and hyperbolas have the same equations if they are defined by their foci A and B and a circumpoint C. We assume this setup because GeoGebra's user interface supports this kind of synthetic construction of conics (Fig. 14).

No.	Name	Toolbar Icon	Description
1	Point O	●	
2	Point V	●	
3	Vector v	↗	Vector(O, V)
4	Point D_1	●	
5	Point D_2	●	
6	Line d	↗	Line D_1, D_2
7	Point F	●	
8	Parabola p	⌒	Parabola with focus F and directrix d
9	Point T	●	Point on p
10	Point S		$T - v$
11	Line m	○	Tangent to p through T
12	Point S'	╲	S mirrored at m
13	Ray r'	↗	Ray through T, S'
14	Implicit Curve c	✳	Envelope(r', T)
15	Segment r	↗	Segment S, T
16	Point S''	●	S' mirrored at T
17	Ray r''	↗	Ray through T, S''

Table 4: Easy construction of caustics of a parabola, the light source is at infinite point

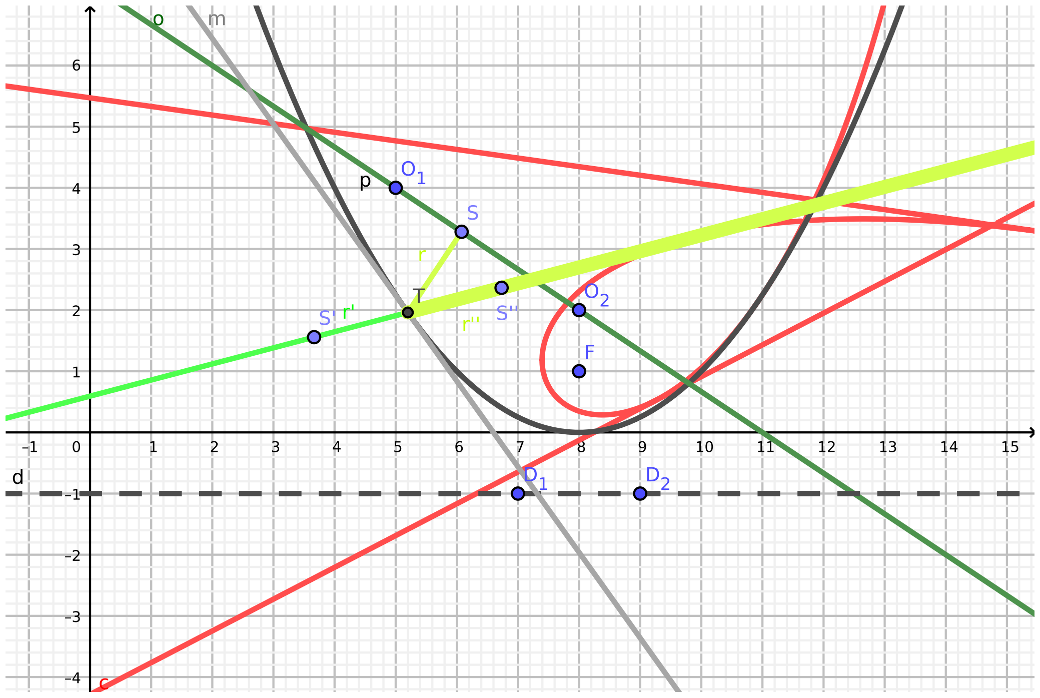


Figure 13: A caustic of a parabola, the light source is at infinite distance (“mouse only” version)

Conic	*Finite*	*Infinite* (“mouse only”)	*Infinite* (via vector)
Circle	14.3	4.0	11.5
Parabola	11.6	1.6	7.2
Hyperbola/Ellipse	–	–	–

Table 5: Benchmarking animation speed (FPS) of caustics of conics, according to the distance of light source

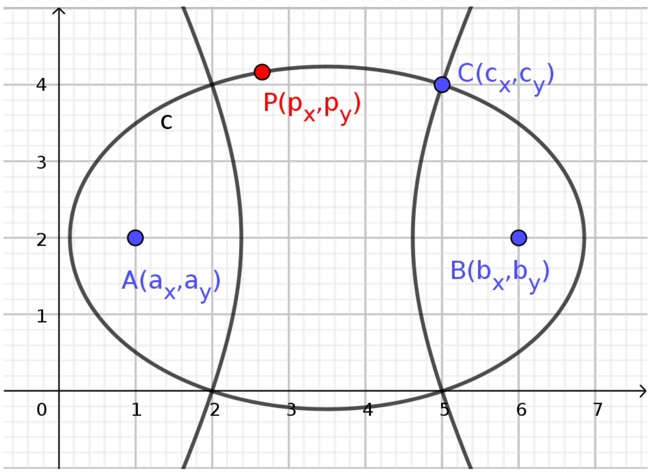


Figure 14: Ellipses and hyperbolas may be indistinguishable in the algebraic geometry approach

We would like to algebraically express the coordinates of P. We tried two strategies to describe the coordinates p_x and p_y:

1. Set up an equation system that contains all required constraints. Use these equations in the complete equation system also, when additional constraints are added (for example, the ones that describe the tangents and mirror images).
2. Set up the same system, but use elimination to obtain *one* equation (a union of the two conics, that is, a *quartic*). Remove the other constraints, and keep only the quartic one which is obtained after elimination. Use only the quartic, and add eventually additional constraints for the "rest" of construction (for example, the ones that describe the tangents and mirror images).

These methods are mathematically equivalent, but in the further computation steps, they may be of different complexity.

In fact, the geometric setup can be expressed by using the following equations:

$$\begin{aligned}
AC^2 &= (a_x - c_x)^2 + (a_y - c_y)^2,\\
BC^2 &= (b_x - c_x)^2 + (b_y - c_y)^2,\\
AP^2 &= (a_x - p_x)^2 + (a_y - p_y)^2,\\
BP^2 &= (b_x - p_x)^2 + (b_y - p_y)^2,\\
AP \pm BP &= AC \pm BC,
\end{aligned}$$

where the last operations decide if an ellipse $(+)$ or a hyperbola $(-)$ is observed. But, as stated above, these operations will have no effect on the obtained output. So we will use "$+$" from now on.

After several attempts with the first strategy, it can be learned that the number of variables may be too high for the underlying CAS. So we tried to use the second strategy by writing a short program in Giac that obtains the quartic in parameters of a_x, a_y, b_x, b_y, c_x and c_y. We needed to eliminate all helper variables from the equation system by using this code:

```
>> poly:=eliminate([ac^2=(a_x-c_x)^2+(a_y-c_y)^2,
        bc^2=(b_x-c_x)^2+(b_y-c_y)^2,
        ap^2=(a_x-p_x)^2+(a_y-p_y)^2,
        bp^2=(b_x-p_x)^2+(b_y-p_y)^2,
        ap+bp=ac+bc],
        [ac,bc,ap,bp])
>> print(poly)
```

Giac gave the following answer (reasonably quickly):

```
poly:[c_x^4*a_y^4
     -4*a_x*c_x^3*a_y^3*c_y
     +6*a_x^2*c_x^2*a_y^2*c_y^2
     -4*a_x^3*c_x*a_y*c_y^3
     +a_x^4*c_y^4
     ...
     -4*c_y*b_x^2*b_y*p_y^4
     -4*a_x^2*b_y^2*p_y^4
     +8*a_x*c_x*b_y^2*p_y^4
     -4*c_x^2*b_y^2*p_y^4]
```

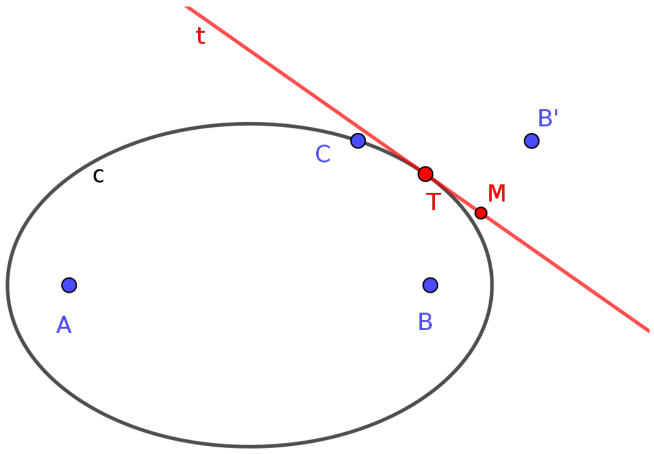


Figure 15: Geometric setup for the tangent of an ellipse/hyperbola

The answer consisted of 1171 terms and more than 20000 characters, so we omit showing the output here.[4] We translated Giac's output into Java and added it explicitly in GeoGebra's source code.

Now we are ready to formalize the tangent of an ellipse (or hyperbola) in a similar way as shown in equations (1)–(5) for the case of a parabola, that is, the "rest". See Figure 15.

The following properties need to be set up:

1. A, T and B' are collinear.
2. M is the midpoint of BB'.
3. $BT = B'T$.
4. A, B and T should not be collinear.

Similarly, we can set up 5 equations (*two* for the midpoint), similarly to (1)–(5). Here we emphasize again that the last property is required to avoid the degenerate case $M = T$.

As mentioned above, the equation system is still too heavy for elimination, if we add some other equations that describe the reflection. So we need to give up studying the caustics of an ellipse/hyperbola by using algebraic equations at this point.

Nevertheless, at least, we are able to obtain some beautiful curves that are created by an arbitrary free point E and its mirror image E' about the family of tangents t of a given ellipse/hyperbola.

Figure 16 shows a union of two curves that are algebraically one curve, a degree 8 curve. This consist of a larger red geometric component (according to the ellipse) and a smaller yellow one (according to the hyperbola). In fact, GeoGebra Discovery computes an additional algebraic component as well, a degree 12 curve (see Figure 17). Despite the high degrees, the relatively fast computa-

[4]The full output can be found in https://github.com/kovzol/geogebra/commit/8654b0e17446c53c3fc2f51aba606d1f87eb7174, in the file `common/src/main/java/org/geogebra/common/kernel/algos/AlgoEllipseHyperbolaFociPoint.java`.

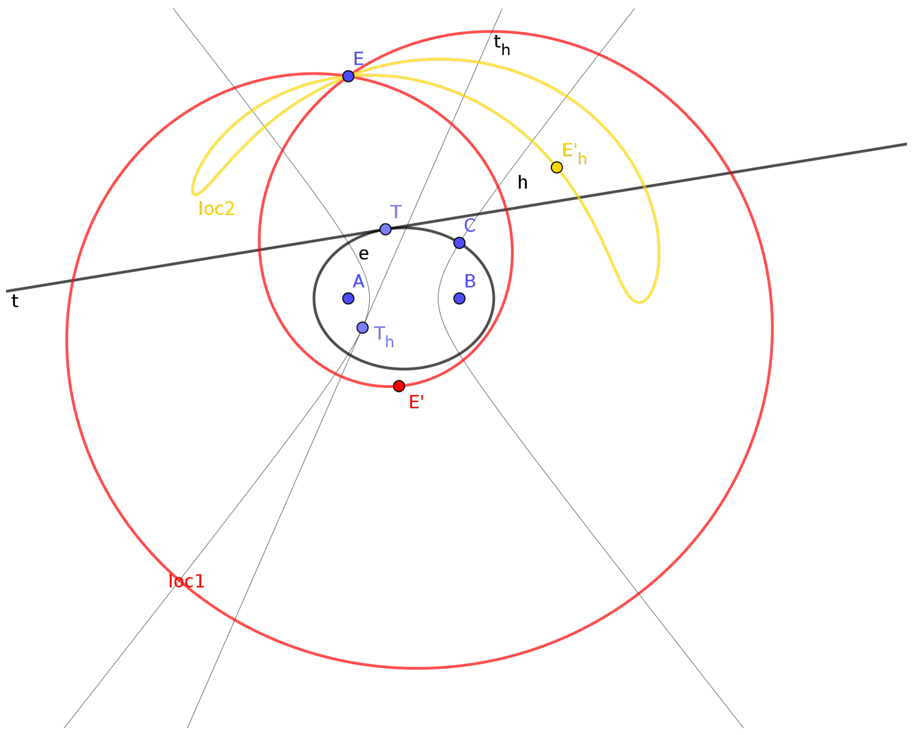


Figure 16: A union of two curves

tion time (animation is between 3.43 and 4.38 FPS[5]) may give some hope that this method has not yet found its extents and some further improvement could allow us to visualize the caustics of an ellipse/hyperbola in a similar way.

5 Conclusions

First, we give some technical remarks. We learned that fast computations may require suitable geometric definitions and a "lucky" way of algebraization. Maybe it is surprising, but avoiding degeneracy can be crucial in certain envelope computations. Since elimination is slow in general (double exponential in the number of variables), a simplification of the figure may play an important role. Finally, exploiting important properties from elementary geometry may also be helpful.

From the educational point of view, we strongly believe that visualization of optics can be a fruitful direction in STEAM projects. Unfortunately, algebraic curves of higher degrees are sometimes completely ignored or underrated in schools. It is clear that several polynomial curves of higher degrees "live among us", and their computer based visualization strengthens the validity to find them a place in the school curriculum as well. Even more so, if the visualization process consists of just a couple of mouse clicks in a popular free software tool!

Higher degree curves have a rich geometry. The same definition with different parameters may re-

[5]See `https://prover-test.geogebra.org/job/GeoGebra_Discovery-art-plottertest/105/artifact/fork/geogebra/test/scripts/benchmark/art-plotter/html/all.html` for a recent benchmark on a set of test cases – our latest example is called `ellipse-point-mirror-tangent*.ggb`.

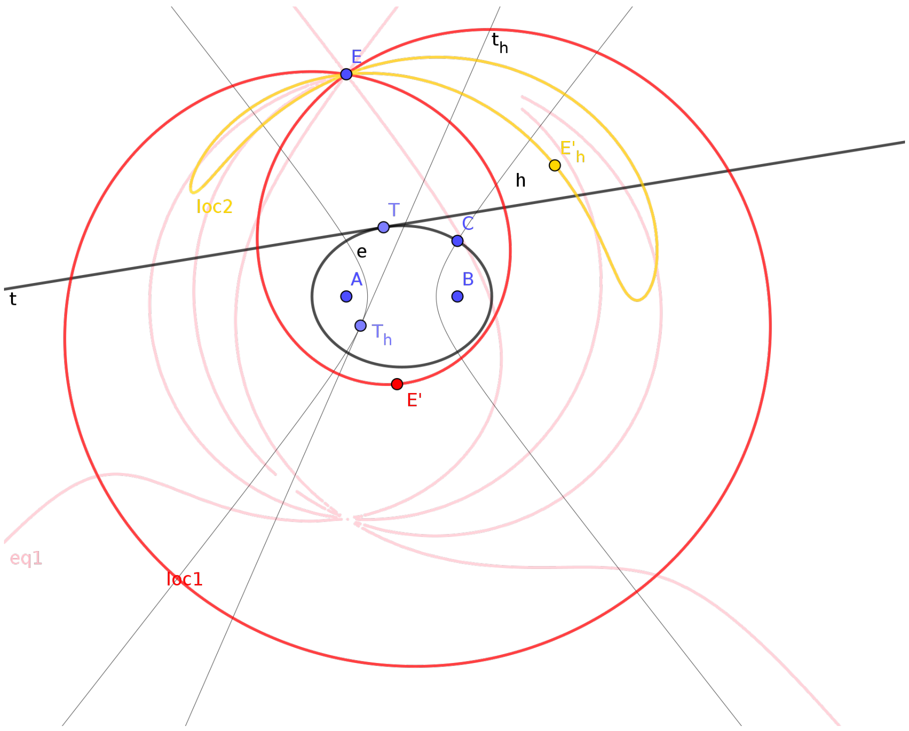


Figure 17: A union of two curves with the extra component

sult in completely different curves. This may encourage young learners to perform self-explorations, on the one hand, because of seeking for beauty, and on the other hand, to discover the challenging (and non-trivial) world of algebraic geometry. Here further steps could be, for example, factorization of the obtained polynomials, removing degenerate components, or self-studying geometric properties.

At CADGME 2010 Botana closed his plenary talk with the rhetorical question after showing the huge and exotic coefficients of a mechanically obtained algebraic curve:

> "Should we hide this (kind of) result to students?"

An appropriate answer today in 2022 may be the following: "We cannot." Science is proving more complicated than we ever thought. But is not complexity beautiful and fascinating?

6 Acknowledgment

The author was supported by the grant PID2020-113192GB-I00 from the Spanish MICINN.

Gerda and Domonkos Kovács kindly helped in the preparation of Figure 6.

Thanks to Csaba Sárvári for his motivation in writing this paper.

References

[1] Miguel Abánades, Francisco Botana, Zoltán Kovács, Tomás Recio, and Csilla Sólyom-Gecse. Development of automatic reasoning tools in GeoGebra. *ACM Commun. Comput. Algebra*, 50(3):85–88, 11 2016.

[2] Francisco Botana, Markus Hohenwarter, Predrag Janičić, Zoltán Kovács, Ivan Petrović, Tomás Recio, and Simon Weitzhofer. Automated theorem proving in GeoGebra: Current achievements. *Journal of Automated Reasoning*, 55(1):39–59, 2015.

[3] Francisco Botana and Zoltán Kovács. A Singular web service for geometric computations. *Annals of Mathematics and Artificial Intelligence*, pages 1–12, November 2014.

[4] Francisco Botana and Zoltán Kovács. New tools in GeoGebra offering novel opportunities to teach loci and envelopes. *CoRR*, abs/1605.09153, 2016.

[5] Francisco Botana and José Luis Valcarce. A software tool for the investigation of plane loci. *Mathematics and Computers in Simulation*, 61(2):139–152, January 2003.

[6] Francisco Botana and José Luis Valcarce. Automatic determination of envelopes and other derived curves within a graphic environment. *Mathematics and Computers in Simulation*, 67:3–13, July 2004.

[7] Thierry Dana-Picard and Zoltán Kovács. Offsets of a regular trifolium (curvas paralelas a un trifolium regular). *Boletín de la Soc. Puig Adam*, 112:63–81, 2021.

[8] Ulrich Kortenkamp. *Foundations of Dynamic Geometry*. PhD thesis, ETH Zürich, 1999.

[9] Zoltán Kovács. Real-time animated dynamic geometry in the classrooms by using fast Gröbner basis computations. *Mathematics in Computer Science*, 11(3-4):351–361, 3 2017.

[10] Zoltán Kovács and Bernard Parisse. Giac and GeoGebra – improved Gröbner basis computations. In Jaime Gutierrez, Josef Schicho, and Martin Weimann, editors, *Computer Algebra and Polynomials*, Lecture Notes in Computer Science, pages 126–138. Springer, 2015.

[11] Zoltán Kovács and Pavel Pech. Experiments on automatic inclusion of some non-degeneracy conditions among the hypotheses in locus equation computations. In Cezary Kaliszyk, Edwin Brady, Andrea Kohlhase, and Claudio Sacerdoti Coen, editors, *Intelligent Computer Mathematics*, volume 11617 of *Lecture Notes in Artificial Intelligence*, pages 140–154. Springer International Publishing, July 2019.

[12] Zoltán Kovács, Tomás Recio, and M. Pilar Vélez. Using automated reasoning tools in GeoGebra in the teaching and learning of proving in geometry. *International Journal of Technology in Mathematics Education*, 25(2):33–50, 2018.

[13] Peter Lebmeir and Jürgen Richter-Gebert. Recognition of computationally constructed loci. In Francisco Botana and Tomas Recio, editors, *Automated Deduction in Geometry*, pages 52–67, Berlin, Heidelberg, 2007. Springer Berlin Heidelberg.

[14] Ernst W. Mayr and Albert R. Meyer. The complexity of the word problem for commutative semigroups and polynomial ideals. *Advances in Mathematics*, 46:305–329, 1982.

[15] Reinhard Oldenburg. Feli-X – ein Computeralgebra-gestütztes dynamisches Geometrieprogramm. *Computeralgebra Rundbrief*, March 2004.

[16] Reinhard Oldenburg. FeliX – mit Algebra Geometrie machen. *Computeralgebra Rundbrief, Sonderheft zum Jahr der Mathematik*, April 2008.

[17] J.L. Rabinowitsch. Zum Hilbertschen Nullstellensatz. *Math. Ann.*, 102(1):520, 1929.

[18] Jürgen Richter-Gebert and Ulrich H. Kortenkamp. *The Cinderella.2 Manual. Working with The Interactive Geometry Software*. Springer Berlin, Heidelberg, 2012.

[19] Eric W. Weisstein. "Tschirnhausen cubic." from MathWorld–a Wolfram web resource, 2022. http://mathworld.wolfram.com/TschirnhausenCubic.html.