Hardware Assistance for Constrained Circle Constructions II: Cluster Merging Problems

Ching-Shoei Chiang,[1] Christoph M. Hoffmann[2] and Paul Rosen[3]

[1]Soochow University, Taipei, R.O.C., chiang@scu.edu.tw
[2]Purdue University, West Lafayette, IN, USA, cmh@cs.purdue.edu
[3]Purdue University, West Lafayette, IN, USA, rosen@purdue.edu

ABSTRACT

In geometric constraint solving, constructing circles with indeterminate radius is an important sub problem. Such constructions are both *sequential*, meaning that we seek a circle tangent to three known geometric entities, as well as *simultaneous*, when several sets of entities, among them variable-radius circles, must be determined together. In Part I of our investigation, we considered sequential constructions in which a single circle had to be constructed from required tangencies to known geometric entities. In Part II, we develop hardware-assisted techniques to solve the simultaneous construction problems of variable-radius circles. We utilize the graphics hardware to determine the solutions. For Part I, this could be accomplished relying fundamentally on constructing distance maps. Here, we need to construct and sample surfaces from configuration space.

Keywords: geometric constraint solving, variable-radius circles, cluster merging, hardware acceleration, GPU.
DOI: 10.3722/cadaps.2010.33-44

1 INTRODUCTION

Geometric constraint solving plays a pivotal role in computer-aided design (CAD). Practical solvers include graph-theoretic solvers in which the constraint problem is decomposed into sub problems of known structure, those sub problems subsequently are solved, and then the solution fragments are assembled into a solution of the original problem. Specialized and general decomposition algorithms have been published and implemented; e.g., [1],[9],[10],[13],[16]. The subsequent construction phase solves small systems of algebraic equations, and for 2D solvers with a focused scope these systems are easy to solve and numerically stable. However, the construction phase becomes demanding when one considers including variable-radius circles and extends the geometric vocabulary to include entities such as PH curves or conics; e.g., [2],[3],[6],[7],[8],[11]. For such constructions, the dominant algebraic approach delivers systems of equations that may require substantial computations as well as generating many solutions, not all of them of geometric significance. In response, commercial solvers will either give up or else employ numerical approximations; e.g. [16]. There is an extensive literature on geometric constraint solving, PH curves, and on Bézier curves. We refer the reader to [19] for a recent survey.

In this paper, we revisit the problem of constructing variable-radius circles, that is, of circles whose radius is not prescribed but must be discovered as part of the solution. In Part I of our study, [4], we advocated using graphics hardware to solve sequential circle constructions. That is, we considered the problem of finding a circle tangent or incident to three entities, where the entities are chosen from among known circles, lines and PH-curves. The equation systems that describe such constructions and must be solved may have a high algebraic degree, motivating the hardware assisted approach. Moreover, a high algebraic degree implies the existence of many solutions, most of them not relevant to the application. For example, in the case of a PH-curve the tangency should be in the [0,1] parameter domain.

Here, we consider *cluster merging* where one of the clusters is a variable-radius circle. That is, we consider how to construct circles from four constraints arising from two partially solved sub problems that relate to each other by an unknown translation or rotation. A simple example illustrates the situation: In Fig. 1, the given constraints are as set forth in the caption. Having fixed the blue segments (AB) and (BC), note that the red segments, (AD) and (DE), can rigidly rotate around point A. Conversely, after fixing the red segments, we may rigidly rotate the blue segments around A. The circle is defined by four distance constraints from the blue and the red sides indicated by the arrows. A solution of this problem should determine the circle's radius and center, as well as the angle at A.

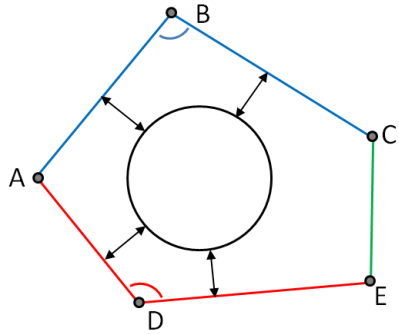

Fig. 1: Rotational cluster merge with a variable-radius circle from four constraints: Lengths (A,B), (B,C), (A,D), and (D,E) and angles (A,B,C) and (E,D,A) are given. The circle is to have a prescribed distance from the blue and the red sides. No symmetry is required.

As shown in [7],[8], the equation systems that arise for these types of constraint problems can be factored using cyclographic maps [15]. Even so, the problems become algebraically more complex. We show how to solve the systems using graphics hardware, transforming the algebra into geometry that is easily and rapidly solved.

Solving geometric constraint problems using the GPU appears to be new, but other GPU algorithms have flourished in recent years in other application areas, including graphics, computational geometry, and linear algebra. We refer the reader to the survey [17]. From a technical perspective, the most relevant prior work is on Voronoi and distance field computations; e.g., [12],[18].

The remainder of the paper is structured as follows. In Section 2 we define the problems we consider and review prior results needed to accomplish our task. Section 3 considers cluster merges where the third cluster is the undefined circle. Section 4 discusses our implementation and performance achieved, as well as accuracy.

For simpler problem instances, such as the one of Fig. 1, the algebraic approach is quite adequate. However, for more demanding situations, such as those discussed in Section 3, we advocate the geometric approach that makes use of the graphics hardware of workstations and laptops. That approach achieves greater simplicity and better performance, and avoids having to solve systems of high algebraic degree.

2 DEFINITIONS AND THEOREMS

2.1 Constraint Solving and Problem Description

We refer the reader to [1] for basic definitions and recall that a geometric constraint problem can be conceptualized as a graph whose vertices are the geometric elements (points, lines, circles) and whose edges represent constraints among them (angle, distance, tangency, concentricity, etc). A solved sub problem is an induced sub graph where all geometric elements can be positioned such that all constraints on them are satisfied. We restrict to structurally well-constrained sub problems which we call *clusters*; [5]. In triangle solvers, three clusters are considered recursively and are such that each pair of clusters shares a vertex. Such clusters can be merged into a larger cluster by determining the relative position of the three shared elements. We consider how to solve three clusters, K_1, K_2, K_3, to be merged where K_3 is a circle of unknown radius and position. Note that circles in K_1 and K_2 must have known radii since they are fully determined relative to the other elements in the cluster.

The shared geometric element between K_1 and K_2 is restricted to a line, a fixed-radius circle, or a point, so that the relative position of the two clusters is determined up to a translation or a rotation. Each cluster contains two elements that are constraining the circle K_3. Those elements can be lines or points, and a simple example is given above in Fig. 1 where K_1 consists of the points A, D, E and the orange lines, and K_2 consists of the points A, B, C and the blue lines. The shared geometry is the point A. Note that elements in K_1 and K_2 can include circles of fixed radius, since they can be replaced with points after adjusting the constraints upon them. For this type of construction, the possible cases have been mapped out in detail in [2],[7],[8] and the algebraic systems formulated and solved. The most complicated case, involving only circles in the clusters K_1 and K_2 gives rise to a system with the algebraic degree 64. Rather than approaching the problem algebraically, we develop a geometric approach and show how it can be solved easily using hardware acceleration.

As in [7], we denote the type of problem by $S(G_1G_2\,, G_3G_4)$, where S denotes the shared geometry, G_1 and G_2 denote the two geometric elements constraining the variable radius circle in cluster K_1, and G_3 and G_4 denote the two geometric elements constraining the variable radius circle in cluster K_2. The G_j may be circles or lines, denoted by C or L, as may be S. Thus, the problem type of Fig. 1 is, in this notation, a $C(LL, LL)$ problem – considering the point A a circle of radius zero. In Section 3, we consider the cases $L(CL, CC)$ and $C(CL, CC)$. Once we know how to solve these cases, the remaining cases are readily solved analogously. We also consider problems in which constraints are included on the center of the unknown circle. Analyzed algebraically in [2], we consider here the cases $L(CC', CL)$ and $C(CC', CC)$.

2.2 Cyclographic or γ-Maps and τ-Maps

We plan to factor the problems into sub problems, and a good way to do so is to consider oriented circles and oriented lines. An oriented circle is called a *cycle*. The orientation may be positive (counter-clockwise) or negative (clockwise), and is expressed by the signed radius. The *cyclographic map* [15],[3],[7] of a cycle with center (a, b) and radius r is the cone $(x-a)^2+(y-b)^2-(z-r)^2=0$. The vertex of the cone is above the *xy*-plane for positive cycles and below the *xy*-plane for negative ones. A point is considered a cycle of zero radius and has both positive and negative orientation.

The cyclographic map of the oriented line with the equation $ax+by+d=0$ is a plane through the line inclined 45 degrees against the *xy*-plane. The line is oriented by the normal (a, b) using a left-hand rule. Following the line in its orientation, the normal is to the left, and the plane is positive also to the left.

For the oriented entity E in the *xy*-plane, its cyclographic map is also denoted by $\gamma(E)$ in the literature; e.g., [2]. Consider the point (x, y, z) on $\gamma(E)$. Then the point corresponds to a cycle with center (x, y) and signed radius z that is tangent to E in a consistent orientation. The cyclographic map $\gamma(E)$ can be defined analogously for any oriented plane curve E and is a ruled surface. The intersection of two cyclographic maps is of interest since it projects to a bisector of the underlying geometric entities. Since the cone's generators are inclined by 45 degrees to the *xy*-plane, the intersection of the cyclographic maps of two circles is a conic section plus a component at infinity (which is not of interest). This fact is easily verified by subtracting the implicit cone equations, yielding the equation of a plane that contains the (finite) intersection of the two cones.

Let C be a circle in the xy-plane. The τ-map of C, denoted $\tau(C)$, is a cylinder through C with axis, and generators, perpendicular to the xy-plane. Likewise, the τ-map of a line L in the xy-plane, denoted $\tau(L)$, is a plane through L that is perpendicular to the xy-plane; e.g. [2]. The τ-map will be used to express constraints on the center of a variable-radius circle geometrically.

2.3 Conic Sections

Given the implicit equations of a cone and a plane, we will use a parametric representation of its intersection, a conic. We restrict to the cones that are cyclographic maps of cycles. Assume that the coordinate system has been chosen such that the vertex of the cone C is at the origin and the z-axis is the axis of rotation. The equation of the cone is then $C: x^2 + y^2 - z^2 = 0$.

Let P' be the plane with equation $[u, v, w, K] \cdot [x, y, z, 1] = 0$ and assume that $u^2 + v^2 = 1$. Here the vector $[u, v, w, K]$ is the coefficient vector of the implicit plane equation. Rotate the coordinate system around the z-axis such that the rotated plane $P'' = [\pm 1, 0, w, K]$ intersects the xy-plane in a line parallel to the y-axis. Since $u^2 + v^2 = 1$, the rotation matrix to be applied to the plane normal is

$$R = \begin{bmatrix} u & v & 0 \\ -v & u & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{1}$$

The resulting plane equation is $[\pm 1, 0, w, K'] \cdot [x, y, z, 1] = 0$. The cone is invariant under this rotation.

If $K' = 0$, the conic is degenerate consisting of two intersecting lines ($|w| < 1$), a double line ($|w| = 1$), or an isolated point ($|w| > 1$). For the nondegenerate cases we have $K' \neq 0$. Then, if $|w| = 1$, the intersection is a parabola. If $|w| < 1$, the intersection is an ellipse, and if $|w| > 1$, the intersection is a hyperbola. The parametric representation of nondegenerate intersections is then, by algebra:

$$\begin{aligned} x(t) &= z(t)\cos(t) \\ y(t) &= z(t)\sin(t) \\ z(t) &= -K'/(\cos(t) + w) \end{aligned} \tag{2}$$

In the elliptic case, the angle t ranges from $-\pi$ to $+\pi$. The major axis of the ellipse is at 0 and π, and the minor axis at $\pm\pi/2$. In the hyperbolic case, the major axis is at 0 and π. When $\cos(t) = -w$, then $z(t)$ is undefined, separating the parametric representation of the two branches of the hyperbola. The midpoint of the hyperbola is

$$M = [K'/(w^2 - 1), 0, -wK'/(w^2 - 1)] \tag{3}$$

and the two vertices of the hyperbola are

$$A = [K'/(w - 1), 0, -K'/(w - 1)], \quad B = [-K'/(w + 1), 0, -K'/(w + 1)] \tag{4}$$

The cyclographic maps of two cycles intersect in a conic section (and at infinity). If the two cones are given by $(x - x_1)^2 + (y - y_1)^2 - (z - r_1)^2 = 0$ and $(x - x_2)^2 + (y - y_2)^2 - (z - r_2)^2 = 0$, then the conic lies in a plane whose equation is obtained by subtracting. After a suitable translation and rotation around the z-axis the intersection conic is parameterized as described before.

2.4 Sweeping and Revolving

To account for the degree of freedom between the two clusters, we will construct from the parametric conic a surface. If the clusters have a translation degree of freedom T, then we sweep the conic in the direction of $\pm T$. If the degree of freedom is a revolution, then we revolve the conic around the axis of revolution. The details are routine.

3 VARIABLE-RADIUS CIRCLES AS CLUSTERS

The constructions of [7],[8] solve all cases in which a circle is sought subject to four constraints on the circle circumference. The extra constraint is necessary to fix a translational or rotational degree of freedom between two clusters that are involved in the constraints on the circle. The cases that arise vary in algebraic complexity. We will explain how to map the algebraic steps of the solutions to a hardware-assisted approach and illustrate the method by solving two of the hardest problems, $L(CL, CC)$ and $C(CL, CC)$. The moving cluster will be the pair of circles, and the circle-line configuration will be kept fixed.

The constraints considered in the earlier papers were on the sought circle's circumference and not explicitly on the circle's center. Cases in which the center of the sought circle is constrained have been considered in [2] using an algebraic approach. Again, a number of cases were distinguished and solved with help of cyclographic maps and with τ-maps. Those cases, and their solutions, can equally be approached using hardware-assisted rendering methods. Here, we consider the (easy) case $L(CL,CC)$, of degree 4, and the harder cases $L(CC',CL)$, of degree 8, and $C(CC',CC)$, of degree 32. The notation C' and L' means that the center of the variable-radius circle should lie on a circle C or on a line L. We will explain what additional steps will be needed to solve this problem extension.

3.1 L(CL, CC)

For the case $L(CL,CC)$ we are given two clusters K_1 and K_2 and a vector T. K_2 can be translated against K_1 in the direction $\pm T$. The variable radius circle K_3 is constrained by a line and three circles. Without loss of generality we assume that those are tangency constraints.

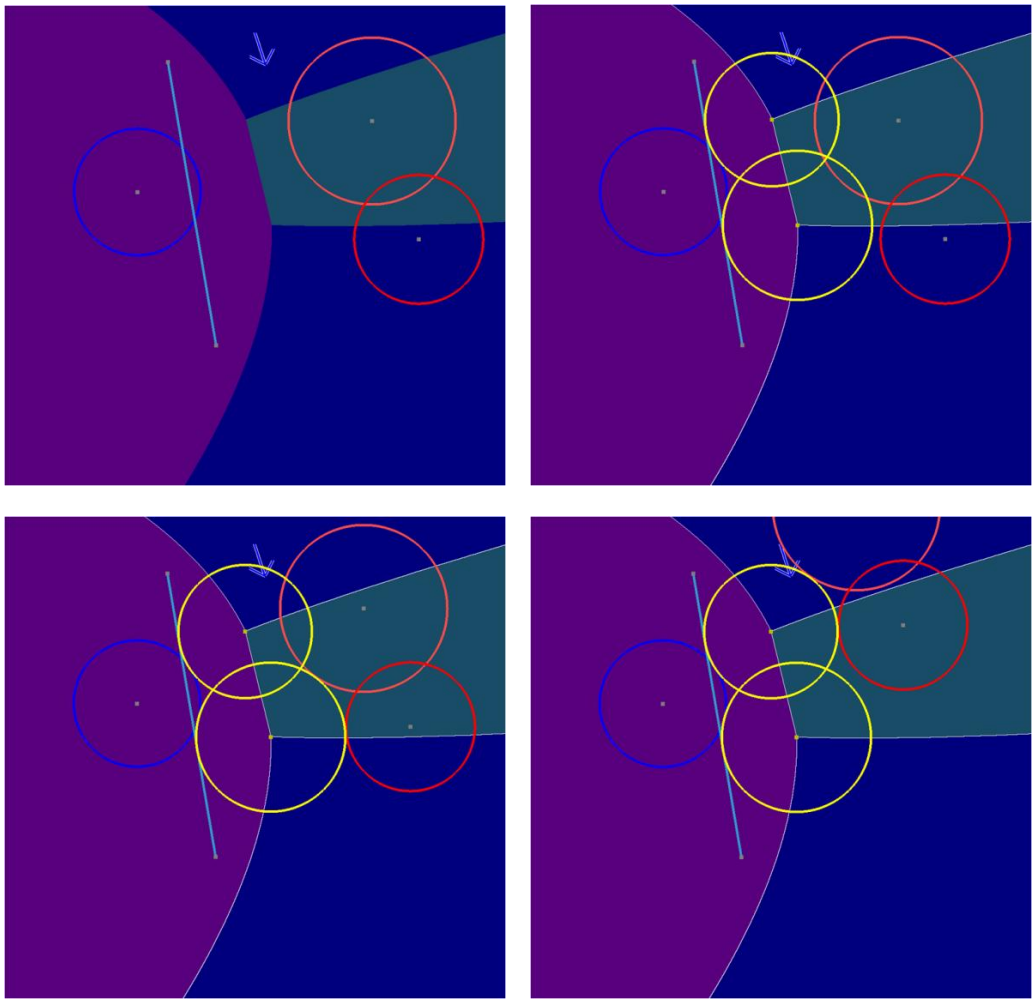

Fig. 2: L(CL, CC) example. *Top left*: Fixed cluster in blue, moving cluster in red, arrow shows translation direction. *Top right*: Two solution found by intersecting the cyclographic maps of the fixed-cluster objects with the purple surface obtained by sweeping the conic intersection of the two cyclographic maps of the moving cluster circles. *Bottom*: the two solutions with the moving cluster translated as required.

We proceed as follows:

1. Determine the cyclographic maps of the circle and the line in K_1 and call them G and P respectively. Here G is a cone and P is a plane. The intersection of the two maps is a conic H_1 represented by the quadratic and a linear equations of G and P.
2. Intersect the cyclographic maps of the two circles in K_2 obtaining a conic H in parametric form. Note that the remaining quadratic intersection is at infinity as mentioned in Section 2.3.
3. Sweep H in the direction T, so obtaining the cylinder H_2 with a quadratic equation.
4. Render the surfaces G, P and H_2 obtaining the intersection of the cylinder H_2 with the conic H_1.

The method is correct because the cylinder H_2 is the map of all cycles tangent to the two given cycles in K_2 for all translated positions. Likewise, the conic H_1 is the map of all cycles tangent to the line and the cycle in the fixed cluster K_1. So the intersection of G, P and H_2 is the map of the sought variable-radius circle. An example of the method is shown in Fig. 2.

The figure shows the fixed cluster objects, a line segment and a circle, in blue. Their cyclographic maps are rendered in dark green (line segment) and dark blue (blue circle). They are the surfaces G and P described in Step 1 above. The two red circles constitute the moving cluster. The blue arrow indicates the direction of translation, T, allowed for the cluster. The two cyclographic maps of the red circles are two cones that intersect in the hyperbola H (Step 2 above). Sweeping this hyperbola in the direction of translation we obtain the third surface to be rendered, the hyperbolic cylinder H_2 of Step 3 above. Two intersections of the three surfaces lie in the domain, and their height above the $z = 0$ plane is the required radius of the sought circle. The solutions are rendered by the yellow circles centered at the projection onto the $z = 0$ plane, of the intersections in 3D. We can demonstrate the correctness of the solutions by translating the two red circles rigidly in the direction of $\pm T$ as shown in Fig. 2, bottom.

The surfaces in the domain are rendered by orthographic projection from the negative z direction, hence we have to render the positive half-space z^+ and the negative half-space z^- separately. Also, because of the assumed orientations, we have to render four combinations.

3.2 C(CL, CC)

The case $C(CL, CC)$ is algebraically the most complex case due to the fact that we have to generate a surface for the moving cluster by rotating the conic intersection of the two cones that are the distance maps of the two cycles. However, it is easy to approach this case geometrically, proceeding as in the $L(CL, CC)$ case. We render the cyclographic maps of the cycle and the directed line segment in the fixed cluster K_1 and determine the conic H that is the intersection of the cones associated with the two cycles in K_2. Instead of extruding H by a vector, however, we now have to rotate the conic H_1 around the center of the circle shared by clusters K_1 and K_2. The axis of rotation is perpendicular to the *xy*-plane. An example is shown in Fig. 3.

As described in [8], the dark blue surface is a hyperbola rotated around a skew axis, with estimated algebraic degree of 8. While an implicit equation for this surface may be difficult to find, it is easy to parameterize the conic and generate a piecewise-linear approximation. As an example, we show the surfaces involved in 3D in Fig. 4.

3.3 Constraints Involving the Center of the Variable-Radius Circle

We consider now the case where one or more of the four constraints is on the center of the circle constituting the cluster K_3. As explained in [2], if E is the geometric entity in K_1 or K_2 that constrains the *center* of K_3, then we must use the τ-map $\tau(E)$ instead of the cyclographic map $\gamma(E)$ to express the constraint. The τ-map is a cylinder if E is a circle and is a vertical plane if E is a line. This fact both simplifies and complicates the geometric approach. Briefly, the simplification would be that two τ-maps in the same cluster intersect in one or two vertical lines, either fixing the center outright, or else each generating a sweep that is a vertical plane or a right-circular cylinder. The complications are as follows:

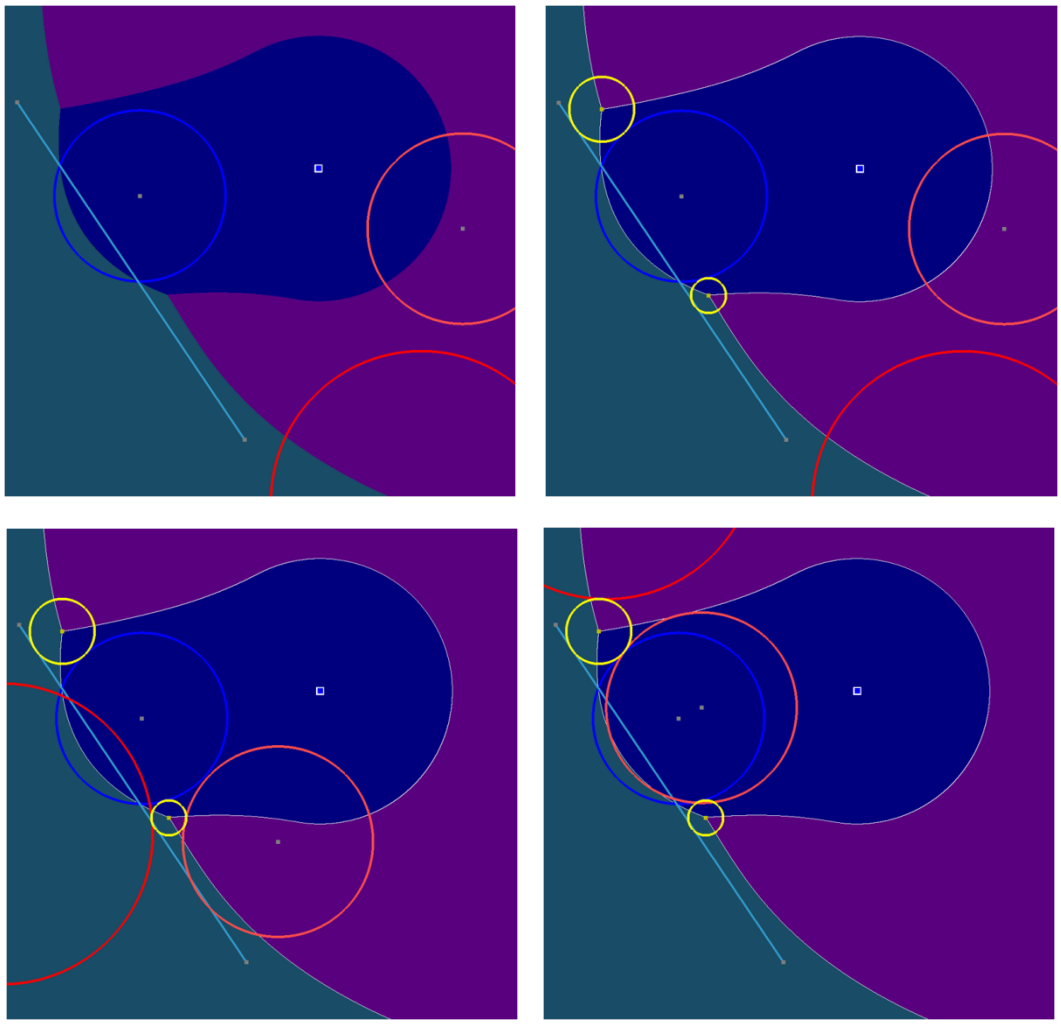

Fig. 3: $C(CL, CC)$ example. *Top left*: Initial configuration and rendered surfaces, blue line and cycle are fixed. Rotation of the moving cluster is around the blue square point of revolution. *Top right*: The rotated conic H generates the dark blue surface. It intersects the cyclographic maps of the line segment and the blue circle in two points in the domain, generating two solutions for K_3 shown in yellow. *Bottom*: Red circles rotated rigidly around the pivot point demonstrating the two solutions of the problem.

(1) The intersection of a cylinder and a cone is in general a space curve of degree 4. In order to deal with τ –maps in the moving cluster, therefore, the determination of the intersection, in parametric form, is an issue. We will deal with this situation by a two-stage process in which the intersection is extracted from the rendered cylinder and cone, with help of the depth buffer. The extracted curve will be a point sequence that naturally allows us to represent it by a piecewise linear approximation and facilitates tessellating the sweep.

(2) A τ-map $\tau(E)$ to be rendered by orthographic projection onto the *xy*-plane is seen edge-on. Since the surface has a vertical ruling, its orthographic projection to the *xy*-plane is precisely the curve E. This means that the surface is not rendered by the graphics hardware and that therefore

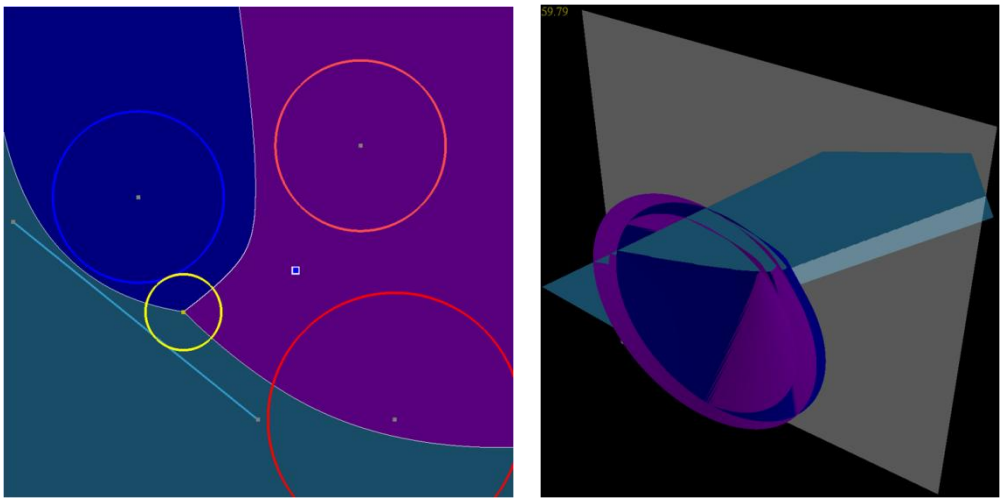

Fig. 4: The surfaces in 3D. The fixed cluster surfaces are the dark blue cone and the lighter blue plane. The purple surface is the swept hyperbola that is the intersection of the two cones that are the cyclographic maps of the two red circles in the moving cluster. The gray plane is the $z = 0$ plane.

extracting pixels adjacent to three differently colored raster areas will not work. Instead, it will be rendered as a curve and so intersected with the intersection curves of the other two surfaces.

(3) When $\tau(E)$ in the moving cluster is a plane, the intersection with a cone or an inclined plane is computed using the methods of Section 2. When the moving cluster requires the intersection of two τ-maps, we obtain one or two vertical lines, so the swept surface has a vertical ruling. This case is dealt with as in (2).

The overall algorithm is thus as follows:

1. Render the maps of the fixed cluster elements E_1 and E_2 in K_1; call the surfaces G_1 and G_2.
2. Compute the intersection curve of the maps of the moving cluster K_2 and call it H. Sweep H and call the surface H'.
   a. If both surfaces of K_2 are τ-maps, render H' as a curve.
   b. If we have to intersect a cone and a cylinder in the moving cluster, we first extract the intersection by rendering the moving cluster maps in the given position and extract the curve as a sequence of points in 3D.
3. Determine the solution; the following cases arise:
   a. If four τ-maps are involved, the circle center is (over)constrained and the circle perimeter cannot be determined.
   b. If three τ-maps are involved, we may assume that two of them are in the moving cluster, so the swept surface H' is either a pair of planes or a pair of cylinders and thus is easy to tesselate.
   c. If two τ-maps are involved, both in the same cluster, we assume they are in the moving cluster and proceed as in Step b.
   d. If two τ-maps are involved, one in K_1 and the other in K_2, then we render one of them as curve.

e. If one τ-map is involved, we assume that the map is in the fixed cluster and proceed as in Step d.

We have implemented several cases involving τ-maps. In Fig. 5, we show the case $L(CC', CL)$ in which there is one constraint on the circle center. This constraint, distance from a point, can be represented by the dark blue circle.

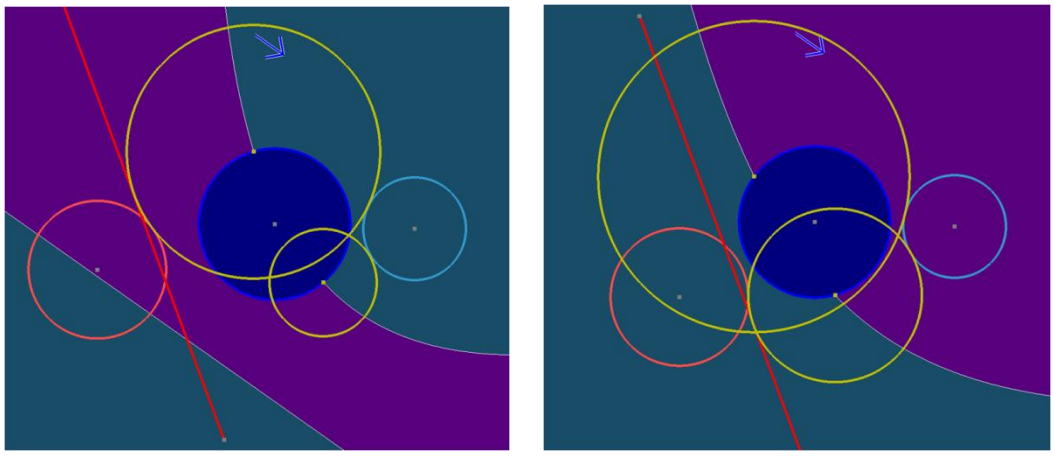

Fig. 5: $L(CC', CL)$ case. The dark blue circle represents the constraint on the center of circle that constitutes the cluster K_3. The light blue cycle, with positive radius, is also part of the fixed cluster K_1. The red cycle (positive radius) and the red line constitute the moving cluster K_2. The left raster shows two of the four solutions, the right the remaining two, in yellow. The position of the moving cluster has been translated in the arrow direction to show how the solution satisfies the constraints.

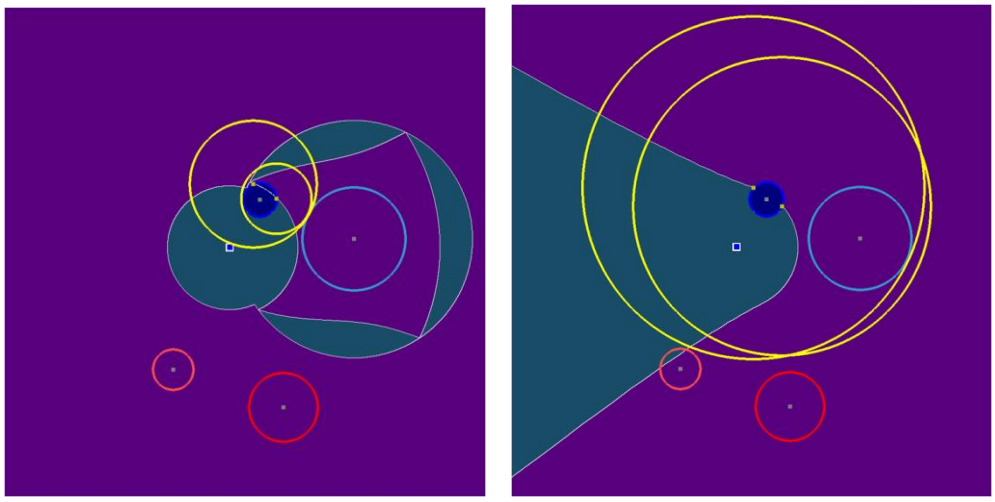

Fig. 6: $C(CC', CC)$ case. The circle is to be tangent to the two red circles of the moving cluster and to the light blue circle of the fixed cluster. The dark blue circle in the fixed cluster represents the constraint on the center of the sought circle, a required distance from a point. The square box indicates the pivot point. Two solutions (left) are found rendering the z^+ half space viewing from below, the other two (right) are found by viewing the same configuration from above, reversing the depth buffer. Additional solutions are obtained by changing cycle orientations.

Four solutions are found, but not all four are visible from below, so some solutions are found by reversing the depth buffer, a computation that is essentially free. In Fig. 6, we show the case $C(CC', CC)$ in which there is also only one constraint of the circle center.

The dark blue circle C_1 in the fixed cluster K_1, with radius r, represents a distance constraint on the center of the sought circle K_3: the center of K_3 is to lie at distance r from the center of C_1. Thus, it must lie on the circle C_1.

4 IMPLEMENTATION AND PERFORMANCE

4.1 Faceting Surfaces

As in [4], we approximate the cyclographic map by a tessellation with an appropriate accuracy. For a raster of 1000x1000 pixels, circular cones are tessellated with approximately 100 facets, yielding a sub-pixel accuracy. With flat shading and orthographic projection, intersections are then extracted by hardware from the raster and the intersection points communicated back. Those points can then be used directly, or else we can use them as starting points for further refinement such as Newton iteration or other suitable numerical procedures.

Because of the geometry of the cyclographic map, tessellation is quite straightforward and error bounds are not difficult to obtain for its accuracy. For instance, the cyclographic map of a cycle is a right-circular cone. We determine a piecewise linear approximation of the circle and then erect over each line segment a planar quadrilateral going outward. Gaps are closed by triangles. Alternatively, we can use the method of Hoff and erect trapezoids over each cycle segment subdivided into two triangles. See also Section 4.1 of Part I.

The swept surfaces, arising from the intersection of the cyclographic maps and the relative motion of the cluster K_2 are algebraically complex, thus a direct tessellation based on the intersection curve is advantageous. Two approaches can be used: We can compute a parametric representation of the intersecting surfaces, and we do so for the sweeps involving conics. A second approach is to extract the intersection from the depth buffer and use a sequence-of-points representation as basis for tessellating sweeps. In the latter case, less accuracy is achieved, but the approach is useful when intersecting τ-maps with γ-maps since their intersection is a space curve of degree 4, so side-stepping the algebra.

4.2 Performance

The performances quoted here were measured on a PC running Windows Vista (32bit) with the following configuration: Intel Xeon X5460 CPU at 3.16GHz, 4GB main memory, and an nVidia GeForce GTX 285 graphics card driving a display with 2560x1600 pixels. The program was run in debug mode alongside routine applications such as Outlook and Word.

A first implementation achieved the disappointing performance of only 20 frames per second (fps) or 50 msec. Here, the circle extraction operation was done in the CPU; that is, the graphics card was strictly used to display. We then programmed the graphics card to do the circle extraction which roughly doubled the speed. Finally, we also computed the vertex positions for the sweep surface on the GPU. The resulting performance was as follows:

For the translational cluster problem $L(CC', CL)$, moving the geometric elements with the mouse or changing their size with the mouse, the performance achieved was between 900fps and 1100fps, around 1 μsec, without extracting and rendering the solution circles. When solution circles were added to the display, performance dropped to about 210fps to 230fps, slightly faster than 5 μsec, depending on the mouse speed. For the rotational cluster problem $C(CC', CC)$ performance was around 180fps and 200fps without circle extraction, and 110fps to 120fps, about 8 μsec, with circle extraction. The performance discrepancy without circle extraction is due to the higher complexity of the tessellation in the rotational case. In the translational case, the extruded cylinders are faceted by long quadrilaterals along the generators. In the rotational case we facet a doubly-curved surface. For instance, in the case $C(CC', CC)$ the rotated surface rendered has 500,000 triangles.

All surfaces were recomputed for every frame. This means that additional speedups are possible if the program first analyzes which surfaces have to be recomputed and reuses the surfaces.

4.3 Hardware Vs. Algebra

Algebraic constraint solvers seek to decompose problems into small systems of algebraic equations which are then solved. When the sub-problem involves circles, a further decomposition is achieved by employing cyclographic maps, distinguishing circle orientations, and τ-maps. This is a highly effective strategy, but it becomes less attractive as the complexity of the sub-problem grows, as is the case when considering higher-order shape elements such as Bézier curves or simultaneous problems. As discussed in [4] and re-affirmed here, the highly parallel sampling approach using the rendering hardware becomes attractive in such situations. As a sampling approach, the GPU computation has limited accuracy. It can be iterated, as pointed out in the first part of our study, but a deft combination with numerical techniques is perhaps a better route. As one of the referees pointed out, sophisticated optimization packages such as *realpaver* [20] can solve some of the equations in part 1 with an impressive speed of 0.06sec. We would speculate that key parts of such packages may already benefit from GPU-based parallelism.

What limitations does the hardware approach have? In applications such as Voronoi diagram determination [12] or sequential constructions of circles, the problem essence is to construct the distance function. For parametric curves, conics and lines this is very straightforward because of the simplicity of the γ-map. In particular, the restriction to the function part of the γ-map is easily accomplished by rendering orthographically the positive half space z^+ and viewing it from the negative z-direction. The constructions considered here only partially fall into this category. That is, the maps constructed and rendered for the elements of the fixed cluster are just to determine the distance functions, but the swept intersections of the cyclographic maps of the moving cluster are structures from *configuration space*. Any point on the swept surface now determines a distance from the elements of the moving cluster *after translating the cluster a particular distance or rotating it by a certain angle*. Therefore, it is no longer enough to simply render all surfaces from below and seek out the common intersections. Instead, we have to look for *all* intersections, and for the surfaces that arise in our constructions this means also looking from above for additional solutions. Fortunately, this can be done by depth buffer reversal, a graphics hardware operation that is as efficient as viewing from below. However, if swept surfaces arise for which a vertical line has more than two intersections, then depth buffer reversal is not enough and the surfaces have to be clipped in some way. That issue needs to be investigated further.

Moreover, we have considered only the distance functions of 2D structures which are 3D surfaces and thus well-suited to the GPU. The distance function of a 3D structure is, however, a manifold in 4-space and thus cannot be directly translated into GPU-based graphics operations. Some work in motion planning in this higher-dimensional setting has been reported in, e.g., [14]. More work will be required to find optimal approaches that do well for constraint solving.

5 SUMMARY

The algebraic approach to solving cluster merging problems where one of the clusters is a variable-radius circle has been considered in [7],[8]. Here, we have replaced solving the algebraic equations that arise by a geometric construction that exploits the capability of commodity graphics hardware. The advantage of doing so is that the solutions are easy and extremely fast to obtain. The speeds of sequential constructions were found in [4] to reach 500fps, that is, within 0.002 sec. The solutions here are more complex, owing to the fact that we have to compute a parametric representation of plane/cone intersections and construct from the resulting curves extrusions or surfaces of revolution. Even with that additional burden, we achieve attractive frame rates of 110fps or better. Most of the performance difference is due to rotational sweeps whose faceting is now more complicated and generates many more facets. Translational sweeps are much faster since the generators are straight lines, thus we need far fewer facets. Accordingly, the performance in the translational case is much higher. Also, there are optimizations possible that can further accelerate the computation that we have not implemented.

The hardware-assisted approach is very attractive because it directly uses the geometric intuition underlying the algebra and can find approximate solutions so much faster. However, as constructions reach beyond distance function computations, the geometric complexity of the surfaces rendered impacts the overall effort to identify all relevant solutions.

ACKNOWLEDGEMENTS

This work has been supported in part by NSC Grant NSC 97-2212-E-031-002, NSC-98-2918-I-031-004, NSF CPATH CCF-0722210, DOE DE-FG52-06NA26290, and a gift by Intel Corp.

REFERENCES

[1] Bouma, W.; Fudos, I.; Hoffmann, C.; Cai, J.; Paige, R.: A Geometric Constraint Solver, CAD 27, 1995, 487-501.
[2] Chiang, C.-S.; Joan-Arinyo, R.: Revisiting Variable Radius Circles in Constructive Geometric Constraint Solving, CAD 21(4), 2004, 371-399.
[3] Chiang, C.-S.; Lin, S.-Y.; Chen, J.C.: The Cyclographic Maps for Bezier Curve, 35th International Conference on Computers and Industrial Engineering, 2005, 453-458; Istanbul, Turkey.
[4] Chiang, C.-S.; Hoffmann, C.: Hardware Assistance for Constrained Circle Constructions I: Sequential Problems, CAD and Applications, 7(1), 2010, 17-32.
[5] Fudos, I.; Hoffmann, C.: A graph-constructive approach to solving systems of geometric constraints, ACM Transactions on Graphics 16(2), 1992, 179–216.
[6] Fudos, I.: Constraint-Based Parametric Conics for CAD, CAD 28, 1996, 91-100.
[7] Hoffmann, C.; Chiang, C.-S.: Variable-Radius Circles in Cluster Merging Part I: Translational Cluster, CAD 34, 2002, 787-798.
[8] Hoffmann, C.; Chiang, C.-S.: Variable-Radius Circles in Cluster Merging Part II: Rotational Cluster, CAD 34, Sep., 2002, 799-806.
[9] Hoffmann, C.; Lomonosov, A.; Sitharam, M.; Decomposition plans for geometric constraint systems, Part I: Performance measures for CAD, Journal of Symbolic Computation, 31(4), 2001, 367–408.
[10] Hoffmann, C.; Lomonosov, A.; Sitharam, M.: Decomposition plans for geometric constraint systems, Part II: New algorithms, Journal of Symbolic Computation, 31(4), 2001, 409–427.
[11] Hoffmann, C.; Peters, J.: Geometric Constraints for CAGD, in Mathematical Methods for Curves and Surfaces, M. Daehlen, T. Lyche, L. Schumaker, eds., Vanderbilt University Press, 1995, 237-254.
[12] Hoff III, K.; Culver, T.; Keyser, J.; Lin M.; Manocha D.: Fast Computation of Generalized Voronoi Diagrams Using Graphics Hardware, Siggraph '99, 277-286, Los Angeles, CA.
[13] Jermann, C.; Trombettoni, G.; Neveu, B.; Mathis, P.: Decomposition of geometric constraints systems: A survey, International Journal of Computational Geometry and Applications, 23(7), 2006, 1–35.
[14] Lauterbach, C.; Garland, M.; Sengupta, S.; Luebke, D.; Manocha, D.: Fast BVH Construction on GPUs, Computer Graphics Forum 28, 2009, 375–384.
[15] Müller, E.; Krames, J.: Die Zyklographie, Franz Deuticke, Leipzig und Wien, 1929.
[16] Owen, J.: Algebraic solution for geometry from dimensional constraints, in SMA'91: Proceedings of the first ACM symposium on Solid modeling Foundations and CAD/CAM applications, 1991, 397–407, ACM Press, New York.
[17] Owens, J.; Luebke, D.; Govindaraju, N.; Harris, M.; Krüger, J.; Lefohn, A.; Purcell, T: A Survey of General-Purpose Computation on Graphics Hardware. Eurographics 2005, State of the Art Reports, 2005, 21-51.
[18] Sud, A.; Govindaraju, N.; Gayle, R.; Manocha, D: Interactive 3D Distance Field Computation using Linear Factorization, ACM Symp. on Interactive 3D Graphics and Games (I3D), 2006, 117-124.
[19] van der Meiden, H. A.: Semantics of Families of Objects, PhD Thesis, Delft University of Technology, The Netherlands, 2008.
[20] Realpaver: http://www.sciences.univ-nantes.fr/info/perso/permanents/granvil/realpaver/