Area-Construction Algorithms Using Tangent Circles

Ching-Shoei Chiang*, Hung-Chieh Li

Computer Science and Information Management, Soochow University, Taipei, Taiwan

Received 17 March 2019; received in revised form 29 August 2019; accepted 02 October 2019

DOI: https://doi.org/10.46604/aiti.2020.819

Abstract

Computer aided geometric design employs mathematical and computational methods for describing geometric objects, such as curves, areas in two dimensions (2D) and surfaces, and solids in 3D. An area can be represented using its boundary curves, and a solid can be represented using its boundary surfaces with intersection curves among these boundary surfaces. In addition, other methods, such as the medial-axis transform, can also be used to represent an area. Although most researchers have presented algorithms that find the medial-axis transform from an area, a algorithm using the contrasting approach is proposed; i.e., it finds an area using a medial-axis transform. The medial-axis transform is constructed using discrete points on a curve and referred to as the skeleton of the area. Subsequently, using the aforementioned discrete points, medial-axis circles are generated and referred to as the muscles of the area. Finally, these medial-axis circles are blended and referred to as the blended boundary curves skin of the area; consequently, the boundary of the area generated is smooth.

Keywords: medial-axis transform, hermite curve, bezier curve, computer-aided geometric design

1. Introduction

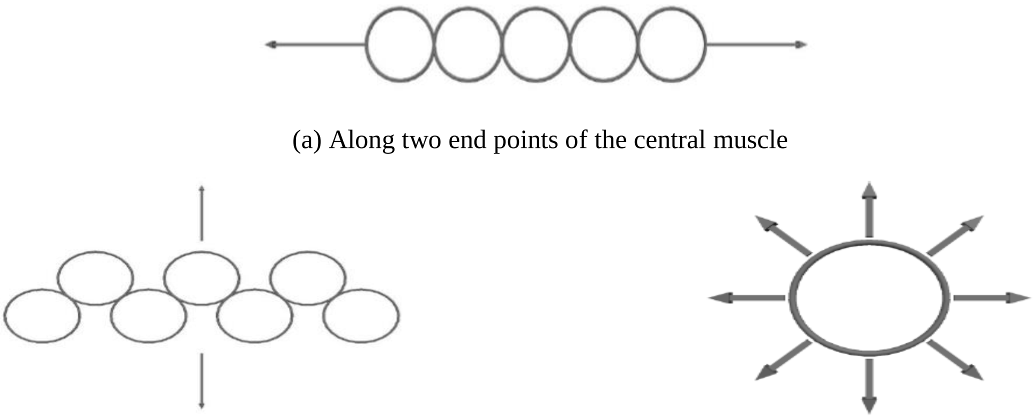
In the domain of computer-aided geometric design, an area can be represented using various methods, such as via its boundary curves, via its medial-axis transform [1], or via set operations, such as union, intersection, and difference, of numerous primitive geometric objects. To describe a curve or area in two dimensions (2D), researchers attempted to use different curve representations in different applications. For example, Cinque, Levialdi, and Malizia [2] used a cubic Bezier curve to describe the shape via its boundary curve with orientation. Yang, Lu, and Lee [3] used a Bezier curve to describe the shape of Chinese calligraphy characters [4]. Chang and Yan [5] derived an algorithm to simulate hand-drawn images using a cubic Bezier curve. Cao and Kot [6] derived an algorithm to embed data using electronic inks without the loss of data. Furthermore, algorithms were proposed to derive the interior/exterior offset curves and medial axis of an area using the boundary curves [7]. In 3D, surface representations are used to simulate natural objects, such as flowers [8]. In addition, the circle packing problems, i.e., the arrangement of circles with same or different radii inside a specified area, were also investigated by researchers [9-11]. In this study, methods are proposed to grow an area. An area is initialized using two approaches, one with the use of medial-axis circles, and the other via points on a curve. Subsequently, the algorithms are proposed to grow the areas using the initialized area.

Given a cubic Bezier curve with n discrete points on it, n circles are attempted to find centered at these n points with different radii, such that the neighboring circles are tangent to each other. This is the initial area that is constructed from a curve that has points on it. Next, the area along different directions is grown; the first direction is along the two end points of the curve, as depicted in Fig. 1(a); the second direction is along the two sides of the curve, as depicted in Fig. 1(b). A single circle is also considered to be an initial area, in this case; the area then grows along different directions, as depicted in Fig. 1(c).

* Corresponding author. E-mail address: chiang@gm.scu.edu.tw
Tel.: +886-2-23111531 ext. 3801; Fax: +886-2-23756878

The initial area generated by the aforementioned curve is referred to as the skeleton of initial area, this curve is referred to as the skeleton of the initial area and these circles as the muscle of the initial area. The union of all the muscles is referred to as the central muscle of the area. Upon extending the area along different directions, the circular arc connecting two tangent circles needs to be blended such that the boundary of the extended area is smooth. The entire process, from skeleton to muscles, and, subsequently, from muscles to skins, leads to the generation of the desired area. The generation of the skeleton and skin is simple. Therefore, the extension of the muscles is mainly emphasized.

In the first section, the problem is illustrated. In the second section, an algorithm is proposed to generate the central muscles, and in the third section, the extending of the muscles is introduced from one side of the skeleton, or from every direction of a circle muscle. The conclusions are presented in the last section.



(a) Along two end points of the central muscle

(b) Along one side of the central muscle

(c) Along multiple directions of a circle

Fig. 1 Muscle growth

2. Generating Central Muscles

Consider the following problem. Given n distinct ordered points p_i, where i = 0, …, n, a set of circles has to be found, C_i, whose circles are centered at these points p_i, and each circle is tangent to its neighboring circles.

In this problem, there are n+1 variables (r_i, where i = 0, …, n) with only n constraint equations ($l_i = \text{dist}(p_i, p_{i+1})$, i = 0, …, n–1). Therefore, one more constraint equation needs solving the aforementioned problem. The following algorithm can be obtained by inputting the value r_0:

***Algorithm 1*:**

```
1. Input the distinct ordered points, i.e., $p_0, p_1, \dots, p_n$.
2. Input $r_0$
3. For i in 0, 1, …, n–1:
4. Calculate $r_{i+1} = l_i - r_i$, where $0 < r_0 < l_0$
```

Although the neighboring circles are tangents to each other, two neighbors of the same circle cannot be ensured that they would not intersect each other. Therefore, to avoid the aforementioned intersection, constraints are placed on the radius of each circle using the following theorems:

Theorem 1: Given p_i, where i = 0, ..., n, if C_{i-1} did not intersect C_{i+1}, where i = 1,.., n–1, then $r_i > (l_{i-1}+l_i-\text{dist}(p_{i+1},p_{i-1}))/2$. (see Fig. 2)

Proof: The result is derived directly from the following equation: dist $(p_{i+1}, p_{i-1}) > r_{i-1}+r_{i+1}$, where $r_{i-1}=l_{i-1}-r_i$ and $r_{i+1}=l_i-r_i$.

Theorem 1 is used to calculate the lower bound of the radius of circle centered at p_i, and these radii are denoted by mr_i, where $i = 1,\ldots, n-1$. Notably, $l_{i-1}+l_i$-dist $(p_{i+1},p_{i-1})>0$ by triangle inequality. Consider the case wherein two circles $C_i = C(p_i, mr_i)$ and $C_{i+1} = C(p_{i+1}, mr_{i+1})$, where $i = 1, \ldots, n-2$ intersect each other; therefore, it is impossible to find a set of circles that are tangents to each other, i.e., the circles do not intersect each other. Therefore, the following example is provided:

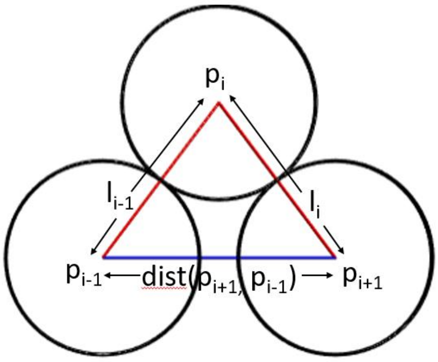
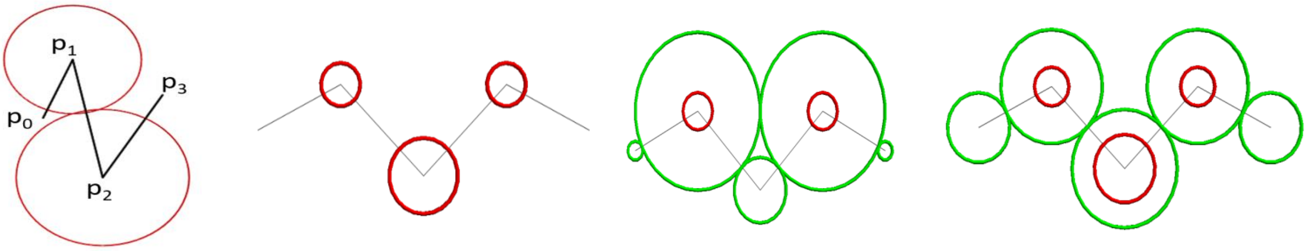


Fig. 2 Intersection of two neighboring circles



(a) Invalid input (b) Minimal radius (c) C_{j-1} is tangent to C_{j+1} (d) Better result

Fig. 3 Circles with different radii

Example 1: Given four points, namely, $p_0 = (0, 0)$, $p_1 = (2, 5)$, $p_2 = (4, -5)$, and $p_3 = (8, 2)$. The minimal radius can be calculated, called mr_i, for p_1, p_2, and p_3 as $mr_1 = 4.59$, $mr_2 = 5.78$, and $mr_1 + mr_2 > l_1 = 10.198$, respectively (see Fig. 3(a)). In this case, irrespective of the values are assigned to r_1 and r_2, there is always the case wherein C_0 intersects C_2 or C_1 intersects C_3.

Example 2: Using the input $p_0 = (0, 0)$, $p_1 = (2, 1)$, $p_2 = (4, -1)$, $p_3 = (6, 1)$, and $p_4 = (8, 0)$, mr_1, mr_2, and mr_3 are calculated as 0.47, 0.82, and 0.47, respectively, and the circles centred at p_i with radius mr_i are depicted in Fig. 3(b).

Assuming j to be the index of the maximum value of mr_i, $r_j = mr_j$ can be set. Consequently, other radii are calculated starting from r_j, and $r_{j+1}, r_{j+2}, \ldots, r_n$ and $r_{j-1}, r_{j-1}, \ldots, r_0$ are also calculated using an approach similar to that used in Algorithm 1. Accordingly, it yields the following algorithm:

***Algorithm 2*:**

1. Input the distinct ordered points, i.e., $p_0, p_1, \ldots, p_n$. Check whether the input is appropriate for the algorithm. If the input is inappropriate, a new set of points is requested for as the input data.
2. Calculate mr_i, where $i = 1, ..., n-1$. Compute index j for $\max\{mr_i$, where $i = 1, \ldots, n-1\}$. Set $r_j = mr_j$.
3. From r_j, repeat $r_{k+1} = l_k - r_k$ for $k = j, \ldots, n-1$.
4. From r_j, repeat $r_{k-1} = l_{k-1} - r_k$ for $k = j, \ldots, 1$.

In this case, the circles are found with p_j as their centers and mr_j their radii, where $mr_j + mr_{j+1} < l_j$. A set of radius r_k is found, where $k=1,2, \ldots, j-1, j+1, \ldots n$, such that the circles will not intersect locally (C_{k-1} and C_k are tangent to each other, and C_j and C_{k+1} are also tangent to each other; in addition, C_{k-1} and C_{k+1} may be tangent to each other but cannot intersect otherwise). For the case $r_j = mr_j$, circle C_{j-1} will be tangent to C_{j+1}, as depicted in Fig. 3(c).

Example 3: Using the same input as that used in example 2, $r_2 = mr_2 = 0.82$ was fixed and r_3 and r_4 was calculated to be 2 and 0.24, respectively. In addition, r_1 and r_0 are equal to 2 and 0.24, respectively, as depicted in Fig. 3(c).

The radius of C_j is increased to avoid C_{j-1} and C_{j+1} being tangents to one another, as depicted in Fig. 3(c). A "better" chain of circles must be produced, such that all circles C_{i-1} and C_{i+1} on the chain are separated locally. The "better" chain of circles may be the case wherein the radii of circles are within a small range. Theoretically, to minimize the value of max $\{r_i, i=0, n\}$ – min $\{r_i, i=0, n\}$. Using the algorithm, all mr_i are calculated, where i = 1, …, n–1, and MC_i, where MC_i are the circles centered at p_i with radius mr_i. If MC_{i-1} did not intersect MC_{i+1}, where i = 1, …, n–1, a set of r_i is found, such that C_i is tangent to C_{i+1} with the property that C_{i-1} did not intersect C_{i+1}. Moreover, r_j has both a lower bound, i.e., mr_j, and an upper bound, which could be min{dist(p_j, p_{j+1})-r_{j+1}, dist(p_j, p_{j-1})-r_{j-1}}; notably, upon increasing r_j, circles MC_{j-1} and MC_{j+1} will not intersect each other. The radius was discretized from its lower bound to upper bound and tried each case and, subsequently, produced the "better" chain of circles satisfying the case wherein max$\{r_i, i=0, n\}$ – min$\{r_i, i=0, n\}$ was minimum. Therefore, there is the following algorithm can be found:

***Algorithm 3*:**

```
1. Input the distinct ordered points, i.e., $p_0, p_1, …, p_n$. Check whether the input is appropriate for the algorithm. If the input is not appropriate, ask for a new set of points as the input.
2. Calculate $mr_i$, where i = 1,.., n–1. Compute index j for max{$mr_i$, where i=1,.., n–1}. Set upper = min{$l_i–mr_{i+1}$, $l_{i-1}–mr_{i-1}$}
3. Set m such that it divides r from its lower bound to upper bound into m+1 values.
4. For $r_j$ from $mr_j$ to its upper bound with step (upper – $mr_j$)/m
    4.1 From $r_j$, repeat $r_{k+1} = l_k – r_k$ for k = j, …, n–1.
    4.2 Repeat $r_{k-1} = l_{k-1} – r_k$ for k = j, …, 1.
    4.3 Calculate max {$r_j$} – min {$r_j$} for j = 1, …, n–1. Memorize k when max {$r_k$} – min {$r_k$} is minimum.
```

Example 4: Using the same input as that used in example 2 to have the lower and upper bounds for $r_2 \in (0.82, 2.36)$. This range linearly was discretized into 11 different radii and observed that when $r_2 = 1.44$, a "better" chain of circles was produced. The result is depicted in Fig. 3(d).

A sequence of points, P = [$p_0, p_1, …, p_n$] is used to produce the associated radii for point p_i using the above algorithm. P is called the skeleton/bone of the area. The produced circles are referred to as the muscles of the area. If the path that connects these ordered points p_i has a sharp angle, it is highly probable that the circles generated using Algorithm 3 intersect other circles locally. To avoid the aforementioned situation, skeleton must be constructed using points on Bezier or Hermite curve.

Consider the case wherein points lie on a Bezier curve B(t), where t is equal to 0, 0.25, 0.5, 0.75, and 1.0. The points are used on the curve as the input for Algorithm 3.

Example 5: Consider the Bezier curve whose control polygon is denoted by [P_0, P_1, P_2, P_3], where P_0 = [0, 0], P_1 = [2, 6], p_2 = [7, – 1], and p_3 = [9, 3]. The control polygon and Bezier curve are depicted using gray line and red curve, respectively, in Fig. 4. Using Algorithm 3, the muscles are calculated to be C(0, 0, 1.969), C(1.969, 2.438, 1.164), C(4.5, 2.25, 1.374), C(7.03, 1.688, 1.219), and C(9.0, 3.0, 1.147), as depicted using green circles in Fig. 4. The case is selected wherein the difference between the maximum radius and minimum radius is minimum. Notably, the polygon that connects points B(0), B(0.25), B(0.5), B(0.75), and B(1.0) did not have sharp angles, and the values of mr_i, where i =1, 2, and 3, are considerably small, i.e., 0.32, 0.007, and 0.199, respectively. MC_1 and MC_3 are depicted as left and right red circles in Fig. 4.

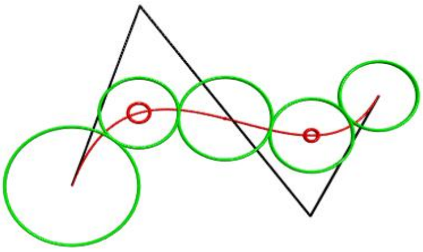
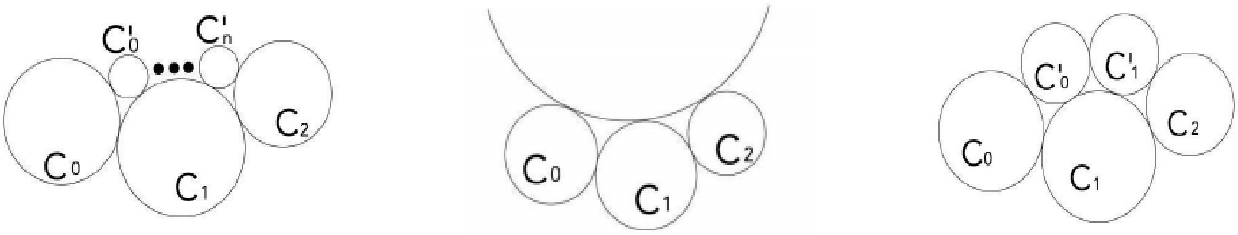


Fig. 4 Skeleton on Bezier curve

3. Generating External Muscles

After generating the central muscles the generation of external muscles is launched. For this, continuous circles that are tangents to one side of the central muscle must be produce, as depicted in Fig. 1(b). Consider 3 circles C_0, C_1, and C_2, as depicted in Fig. 5. A sequence of circles wants finding, such that the following conditions are satisfied: the first circle C_0' is tangent to C_0 and C_1; the last circle C_n' is tangent to C_1 and C_2; circle C_i' between C_0' and C_n' is tangent to C_1 and its neighbors C_{i-1}' and C_{i+1}', as depicted in Fig. 5(a). Fig. 5(b) and Fig. 5(c) depicts the cases for $n = 0$ and $n = 1$. Algorithms are developed for this case in this section. Two subcases are considered, the first case wherein circles C_i', where $i = 0, \ldots, n$, have the same radius, and the second case wherein the radii of these circles have the arithmetic progression property.

The first case is started from considering the case wherein a circle is tangent to 3 circles, as depicted in Fig. 5(a); this case comprises a well-known problem called Apollonius's problem, which has 8 solutions. On the basis of the orientation of circles (a circle oriented clockwise has positive radius, and that oriented counter-clockwise has negative radius), the problem can be simplified by solving a quadratic equation. After solving the equation, two circles are founded, one contains all the three circles, and the other contains none of the three circles. The wanted circle, in this case, is the one that contains none of the three circles.



(a) n+1 circles

(b) One circle

(c) Two circles

Fig. 5 Construction of circles along one side

For finding a circle that is tangent to the three circles, the problem can be solved algebraically. Assume that the three circles have (x_i, y_i) as their centers and r_i as their radii, where $i = 1, 2$, and 3. The circle centered at (x, y) with radius r is tangent to these three circles. Therefore, there are three equations as $(x - x_i)^2 + (y - y_i)^2 = (r - r_i)^2$, where $i = 0, 1$, and 2, with three variables x, y, and r. Because each of the aforementioned three equations are of degree two, they should have eight solutions in total. These three equations are referred to as eq_0, eq_1, and eq_2. Notably, by performing $eq_0 - eq_1$, the terms with degree two are eliminated, and, accordingly, the equation obtained is a linear equation. The system of equation can be simplified $\{eqn_0, eqn_1, eqn_2\}$ into $\{eqn_0, eqn_0 - eqn_1, eqn_0 - eqn_2\}$, where the new system of equation is of degrees two, one, and one, respectively; therefore, the number of solutions becomes two.

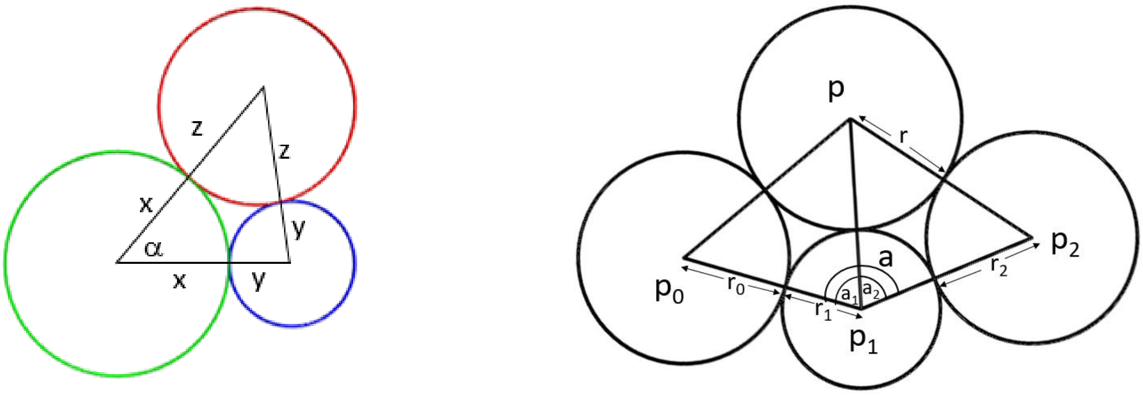
In the algorithm, the binary approach is used to find the radius of the tangent circle. This algorithm is introduced because it can be easily extended to the case with more circles, as depicted in Fig. 5(b) and 5(c). Before the algorithm, there are the following theorems.

Theorem 2: Three circles are tangent to each other, and their radii are x, y, and z. Furthermore, the angle α of the triangle that is formed by connecting their centers has the opposite side of length y + z, as depicted in Fig. 6(a). Then:

$$\cos(a) = \frac{x^2 + xy + xz - yz}{x^2 + xy + xz + yz} \tag{1}$$

Proof: The result is derived directly from the cosine theorem, that is, $(y + z)^2 = (x + y)^2 + (x + z)^2 - 2(x + y)(x + z)\cos(\alpha)$.

Using the equation in Theorem 2, there are four variables, namely, x, y, z, and α, with one equation. Given any three variables, the fourth one can be calculated. Assume that there are two circles with radii x and y, and angle α. Accordingly, radius z can be calculated. Besides, if x and y are known, and assumed radius z, α can be calculated as well.



(a) One circle tangent to two circles

(b) One circle tangent to three circles

Fig. 6 Relation between radii and angles

Now, consider 3 circles C_i, where i = 0, 1, and 2. Assume that C_0 is tangent to C_1 and that C_1 is tangent to C_2. In addition, assume that the circle that is tangent to C_0, C_1, and C_2 has radius r. From this assumption and considering Fig. 6(b), angles α_1 and α_2 can be calculated using the equation in Theorem 2, and, subsequently, their sum is compared with the angle between vectors $p_0 - p_1$ and $p_2 - p_1$. If the sum is less than the angle between the vectors, radius r enlarged, and if the sum is greater than the angle between the vectors, radius r reduced, until the solution is found within a tolerance. This aforementioned idea can be implemented using the following algorithm:

***Algorithm 4*:**

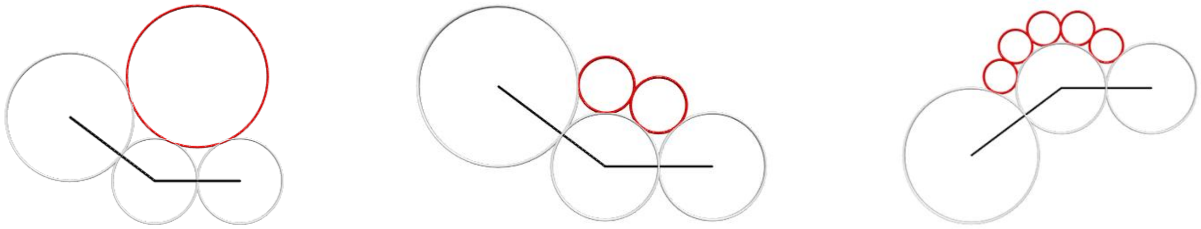
```
1. Input three circles $C_0$, $C_1$, and $C_2$. #assume that $C_0$ is tangent to $C_1$ and that $C_1$ is tangent to $C_2$.
2. Calculate the angle between vectors $p_0$-$p_1$ and $p_2$-$p_1$, and refer to it as $\alpha$.
3. Set r, rmin, and rmax as 1, 0, and 999, respectively.
4. Set eps = 0.001
5. Calculate angles $\alpha_1$ and $\alpha_2$ using the assumed radius r.
6. While $|\alpha_1 + \alpha_2 - \alpha| >$ eps:
   If $\alpha_1 + \alpha_2 > \alpha$:
   rmax = r
   r = (rmax + rmin)/2
   else:
   rmin = r
   r = (rmax + rmin)/2
7. Display the result.
```

Example 5: Consider three circles C_0 (–4, 3, 3), C_1 (0, 0, 2), and C_2 (4, 0, 2). Notably, C_0 is tangent to C_1 and C_1 is tangent to C_2. Circle C'_0 (2, 6.60, 4.90) is found, as depicted in Fig. 7(a).

Now consider the case wherein n+1 circles want adding. In this case, the first circle C_0' is tangent to C_0 and C_1; the last circle C_n' is tangent to C_1 and C_2; circle Ci′ between C_0' and C_n' is tangent to both C_1 and its neighbors C_{i-1}' and C_{i+1}', as depicted in Fig. 5(c). A similar idea as that for Algorithm 4 can be applied. In Algorithm 4, The sum of two angles, α_1 and α_2, are compared with angle α, and now the sum of n + 2 angles is compared with angle α. It is referred to as the extended Algorithm 4.

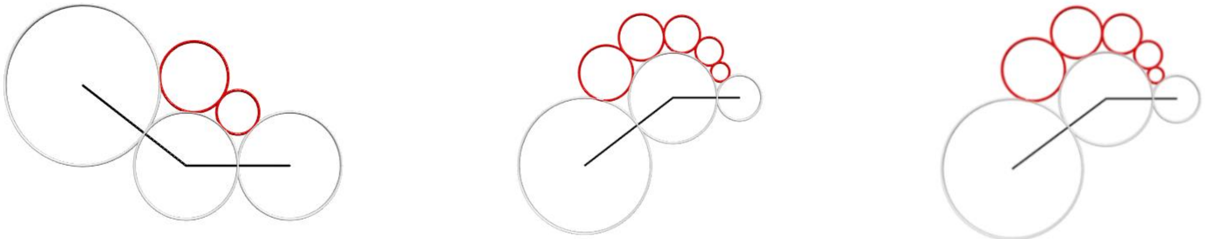
Example 6: Using the same input as that used in example 5, 2 circles that are tangent to each other want finding, where the first circle is tangent to C_0 and C_1 and the second circle is tangent to C_1 and C_2. Using the extended Algorithm 4, these two circles are found centered at (0.047, 3.045) and (2, 2, 97) each of radius r = 1.046, as depicted in Fig. 7(b).

Example 7: Given three circles C_0 (–4, –3, 3), C_1 (0, 0, 2), and C_2 (4, 0, 2). Using the extended Algorithm 4, 5 circles are found between C_0 and C_2, such that these 5 circles are tangent to C_1. The circles are found centered at (–2.697, 0.513), (–2.031, 1.846), (–0.767, 2.636), (0.723, 2.648), and (2.0, 1.88) each of radius 0.745, as depicted in Fig. 7(c).



(a) One circle (b) Two circles (c) Five circles

Fig. 7 Tangent circles with similar radii



(a) 2 circles with dr = 0.5 (b) 5 circles with dr = 0.2 (c) 5 circles with dr = 0.25

Fig. 8 Tangent circles with different radii

Let people consider the case wherein the radii of the circles along one side of initial area are in an arithmetic progression. Assume that dr is the common difference. Accordingly, the first circle has radius r; the second circle has radius r – dr; the third circle has radius r – 2*dr, …, and so on. Using the binary approach as that employed in Algorithm 4, to find 2 or 5 tangent circles, examples 8 and 9 are referred to as depicted in Fig. 8(a) and (b), respectively.

Example 8: Using the same input data as those used in example 5 and setting dr = 0.5, two circles are found: C'_0 (0.31, 3.31, 1.32) and C'_1 (2.0, 1.99, 0.82), as depicted in Fig. 8(a).

Example 9: Using the same input data as those used in example 7 and setting dr = 0.2, 5 circles are found: C'_0 (–3.03, 1.09, 1.22), C'_1 (–1.43, 2.66, 1.02), C'_2 (0.41, 2.79, 0.82), C'_3 (1.64, 2.04, 0.62), and C'_4 (2.14, 1.13, 0.42), as depicted in Fig. 8(b). Upon increasing dr, the difference between the radii of C'_0 and C'_4 is also increased. The difference can be seen in radii by comparing Fig. 8(c), where dr = 0.25, with Fig. 8(b), where dr = 0.2.

Now, consider the case wherein the external muscle/circle grows along every direction of a muscle/circle, as depicted in Figs. 1(c) and Fig. 9. In this case, a list of angles is needed; the list indicates the direction along which new circles want extending. The extension might have no solution. Consider a unit circle centered at the origin, with the list of angles, AList, where the first element in AList is 0. Other cases can be converted into this situation by translating, rotating, and scaling the circles in the problem. With this initial condition, it yields the following algorithm:

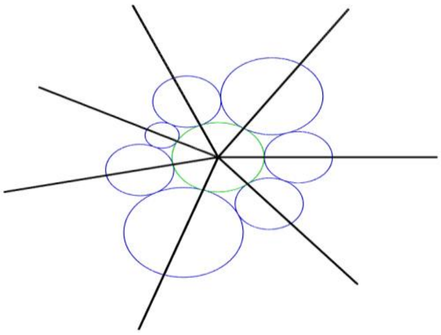


Fig. 9 Growth direction of circles

Algorithm 5**:** (eps is a considerably small number)

1. Input C(0,0,1) and a list of angles, Alist. The length of Alist denotes len(Alist)
2. Assume the initial radius of the first circle along the X+ axis, and refer to it as r_0.
3. Calculate r_1 using the equation in Theorem 2 with the angle data provided by Alist. Continue this process until all r_i, where i = 1, 2, …, len(Alist) – 1 have been calculated.
4. Calculate the sum of the angles between the angle generated by connecting the center of three circles as C_0CC_1, C_1CC_2, …, C_nCC_0, and refer to it as rsum.
5. While abs(rsum – 2π)>eps:

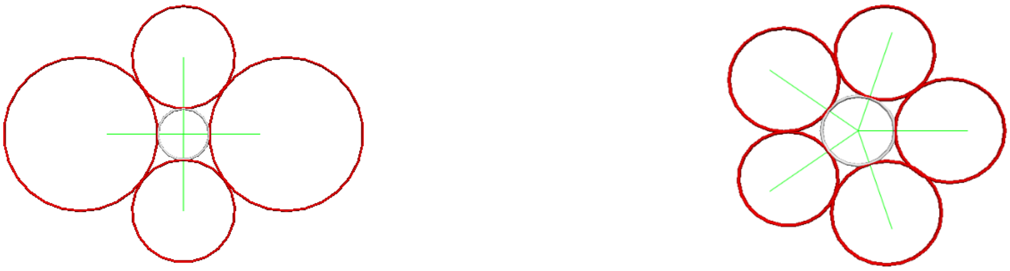
   If rsum > 2π:

   reduce the assumed radius r_0

   else:

   increase the assumed radius $r_{0.}$

Example 10: Assuming the input as Alist=[0, 90, 180, 270], the four circles obtained are C(4, 0, 3), C(0,3,2), C(–4, 0,3), and C(0, –3, 2), as depicted in Fig. 10(a). Notably, the radii of these circles are not the same, although their directions of extension are east, north, west, and south, respectively. There are more than one solution, and the algorithm finds an appropriate one.



(a) Four directions

(b) Five directions

Fig. 10 grow external muscle

Example 11: Assuming the input as Alist=[0, 72, 144, 216, 288], the five circles obtained are C(2.5, 0, 1.5), C(0.73, 2.42, 1.36), C(–2.02, 1.47, 1.5), C(–1.91, –1.39, 1.36), and C(0.77, –2.38, 1.5).

To generate skin to wrap an area, such as the area in Figs. (7)-(10), the interior and exterior circle muscles must be identified. For two exterior muscles tangents to one another, the skin to connect these two circles can be easily produced. A circular arc is an appropriate choice to blend these two circles, provided the radii of the circles contain the circular arc. Using the chosen radius, both the center of the circle and the start/end angle of the circular arc can be easily obtained.

4. Conclusion and Future Research

Tangency among circles has numerous important applications, such as in packing problems. Although most researchers concentrate on finding circles inside an area, a contrasting approach is employed by constructing the area inside out, from a 1D curve (skeleton) to 2D area (union of circles/muscles), in order to ensure a smooth boundary area (G^1-continuity boundary curve). The algorithm used to extend the area is based on the binary approach, and it can be easily used for growing areas.

Future research will emphasize on the design of area-growth directions. For example, the area generated from piecewise Cubic Bezier Curve, so that the whole area is extended to produce an elongated area, certain data structures, such as medial-axis transforms, can be easily generated. The other example is to automatically grow an area to approach a bounded one.

Acknowledgment

This work was supported in part by the National Science Council in Taiwan under Grant MOST 104-2221- E-031-005.

Conflicts of Interest

The authors declare no conflict of interest.

References

[1] C. S. Chiang, “The medial Axis transform of the region defined by circles and Hermit Curve,” International Conference on Computer Science and Engineering (ICCSE), July 22-24 2015.

[2] L. Cinque, S. Levialdi, and A. Malizia, “Shape description using cubic polynomial Bezier curves,” Pattern Recognition Letters, vol. 19, pp. 821-828, 1998.

[3] H. M. Yang, J. J. Lu, and H. J. Lee, “A Bezier curve-based approach to shape description for Chinese calligraphy characters,” Proceedings of the Sixth International Conference on Document Analysis and Recognition, pp. 276-280, 2001.

[4] C. S. Chiang and L. Y. Hsu, “Describing the edge contour of Chinese calligraphy with circle and cubic Hermite Curve,” 2013 Computer Graphics Workshop, July 2013.

[5] H. H. Chang and H. Yan, “Vectorization of hand-drawn image using piecewise cubic Bezier curves fitting,” Pattern Recognition, vol. 31, no. 11, pp. 1747-1755, 1998.

[6] H. Cao and A. C. Kot, “Lossless data embedding in electronic inks,” IEEE Transactions on Information Forensics and Security, vol. 5, no. 2, pp. 314-323, 2010.

[7] L. Cao, Z. Jia, and J. Liu, “Computation of medial axis and offset curves of curved boundaries in planar domains based on the Cesaro’s approach,” Computer Aided Geometric Design, vol. 26, no. 4, pp. 444-454, 2009.

[8] P. Qin and C. Chen, “Simulation model of flower using the integration of L-systems with Bezier surfaces,” International Journal of Computer Science and Network Security, vol. 6, no. 2, pp. 65-68, 2006.

[9] G. L. Orick, K. Stephenson, and C. Collins, “A linearized circle packing algorithm,” *Computational Geometry* 64, pp. 13-29, 2017.

[10] O. Angel, T. Hutchcroft, A. Nachmias, and G. Ray, “Unimodular hyperbolic triangulations: circle packing and random walk,” *Inventiones mathematicae* 206.1, pp. 229-268, 2016.

[11] K. E. Stange, “The sensual Apollonian circle packing,” Expositiones Mathematicae 34.4, pp. 364-395, 2016.