CS6491-2017—Project 2: PCC Cage for FFD

Anna Greene **Dibyendu Mondal**

ABSTRACT

The first goal of this project is to define and implement a planar cage, B, that is controlled by 6 control points that has, for boundary, a smooth Piecewise-Circular Jordan Curve and that has a branch-free Medial Axis. The second goal is to parameterize B, so as to define a homeomorphism (continuous bijective mapping) between a rectangle and B. The third goal is to create an animation B(t) of B by prescribing a cyclic motion for each one of its control points and then to use the resulting time parameterized map to animate a portion of an image, I, that is defined as the intersection (cut-out) of I with a user-controlled initial version of B.

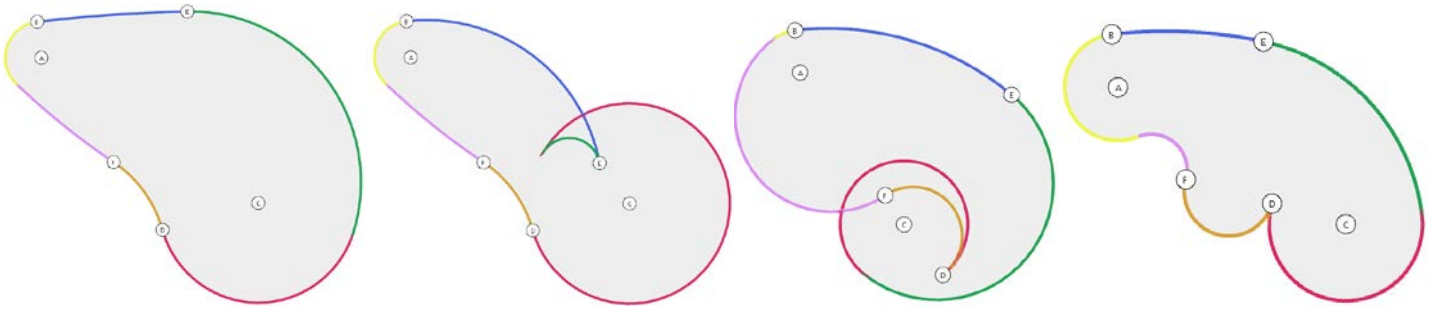
1 Smooth Piecewise-Circular Boundary (SPCB)

We define a ***Smooth Piecewise-Circular Boundary*** (***SPCB***), B, to be a planar curve that has the following properties:

- B is planar
- B is a Jordan curve (closed loop curve without self-crossing or self-overlap)
- B is composed of smoothly joined circular arcs.

Note that we consider a straight-line segment to be a degenerate (special case) circular arc.

We show below (left) an example of an SPCB with arcs drawn using different colors and (right) 3 examples of Piecewise-Circular Curves that are not SPCBs because they are not smooth, self-cross, or self-overlap.



2 Stroke and its control points

We define a ***stroke***, S, to be a planar region bounded by an SPCB made of 6 arcs.

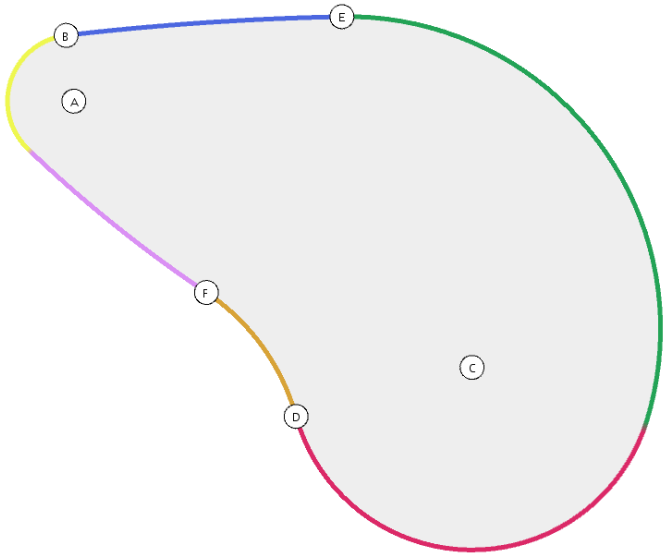
We propose to use a set of 6 ***control points*** (A, B, C, D, E, F) to control the shape of the SPCB that bounds stroke S.

This number of control points is optimal because it allows us to model all possible strokes. To justify this, consider the act of creating a stroke of paint on a canvas from left to right. For each stroke, there is a starting point where the brush first touches and an ending point when you eventually take the brush away. This start and end are represented by the two control circles as there is a variable radius of the brush itself. Then, while you are creating the rest of the stroke, one of three things can happen to the border of the shape at the top and the bottom. That border can either be a concave curve, a convex curve, or a curve that starts out as one and switches to the other somewhere in the middle. By controlling the stroke by six points, we are able to represent each of these cases just by moving the appropriate points.

The relation between a stroke and its control points is best described by the following construction referencing arc by their color and the accompanying diagram showing the arcs of the boundary bS of B in different colors.

Point A is the center of the circle that supports the yellow arc of bS. B is a point on that circle's circumference that marks the starting point for the blue arc. Point E is a point outside the two defined circles and is where the blue arc ends and the green arc starts.

Point C is the center of the circle that supports the red arc of bS. D is a point on that circle's circumference that marks the starting point for the orange arc. Point F is a point outside the two defined circles and is where the orange arc ends and the purple arc starts.

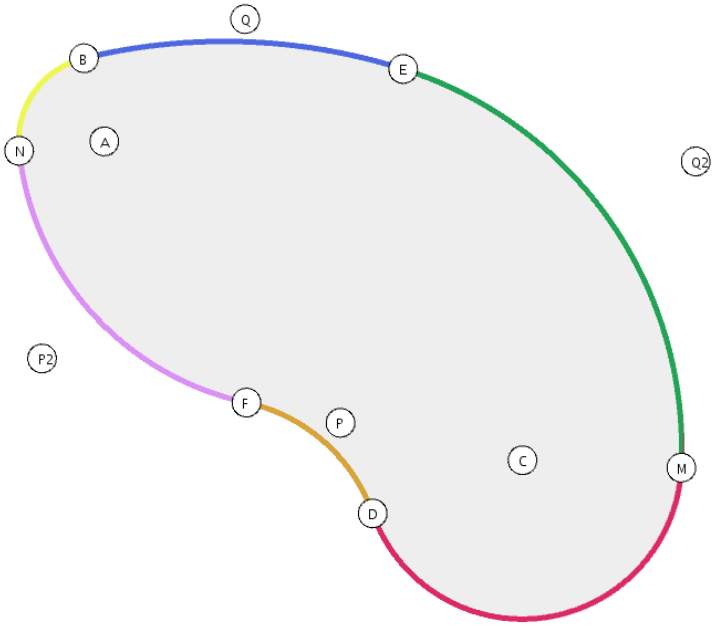


3 Representation and computation of the SPCB of a stroke

To draw and process a stroke S, we first compute its ***explicit representation*** from its control points. This representation stores points:

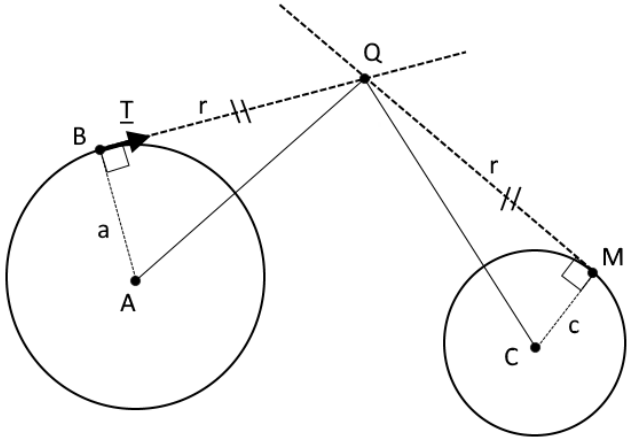
- Q, Q2, P, P2, that represent the points that define the hats of the blue, green, orange, and purple arcs, respectively.
- M, N that represent the points at which the green and purple arcs join the circles defined by points C and A, respectively.

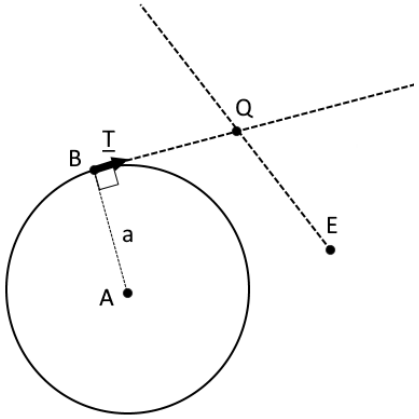
The figure below illustrates the role of these entities for a typical stroke shown with its multi-colored boundary.



Given control points (A, B, C, D, E, F) of a stroke S, we compute the explicit representation of S by the following process.

First, we calculate an expression for a point Q in between two circles that will be the top of the hat that allows us to draw an arc tangent to both circles starting at a certain point. In the diagram on the left below, triangles ABQ and CMQ are both right triangles, so we can use the Pythagorean theorem to our advantage. In this case, it is most convenient to consider the equation $|CQ|^2 = c^2 + r^2$. Since Q can also be represented in vector notation as $B + r\underline{T}$, we replace Q in our equation to get $(CB+r\underline{T})^2 = c^2 + r^2$, and we can solve for r.





However, since we want to find the hat starting at point B that ends at point E - in order to draw the blue arc - we simply consider the second circle in this diagram to be a circle of radius zero (c=0). This simplifies the equation further to $(EB+r\underline{T})^2 = r^2$, and we can again solve for r.

The mathematical formula for finding the length, r, of the two vectors BQ and QE that comprise the hat is as follows, assuming points A, B, C, D, and E as shown in the diagram above, radii |AB|=a and |CD|=c, and tangent vector $\underline{T} = \underline{AB}°$:

$r= (- |EB|^2) / (2 (EB \cdot \underline{T}))$

Then, since point Q is defined as $Q = B + r\underline{T}$, we can calculate Q and draw the hat between points B, Q, and E.

Next, we calculate point Q_2 to draw the green arc starting at point E, such that it is a continuous curve, and ending at point M on the circle defined by points C and D. To accomplish this, we find the starting direction, T_2 by examining the ending direction for the previous blue arc. Then we find the point, Q_2, along that direction such that the distance between Q_2 and E is the same as distance between Q_2 and point M. Another restriction that we took into account was that the vector MQ_2 must be tangent to the circle defined by points C and D. These new points, Q_2 and M, define the top and end of the hat (E, Q_2, M) that allows us to draw the appropriate green arc.

The formula for finding the length, r_2, of the two vectors EQ_2 and Q_2M that comprise the hat is as follows, assuming a new starting direction of $\underline{T_2} = \underline{QE}$, and using our original equation above, switching point B to E, of $(CE+ r_2\underline{T_2})^2 = c^2 + r_2^2$:

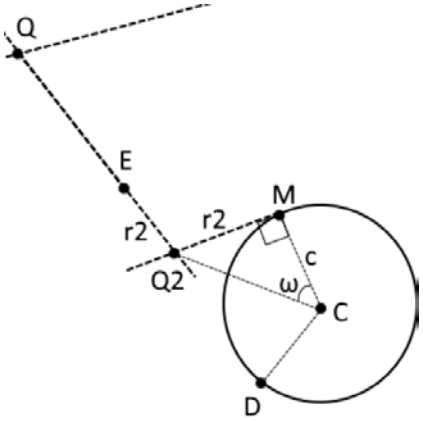
$r_2= (c^2 - |CE|^2) / (2 (CE \cdot \underline{T_2}))$

Point Q_2 can then be represented as:

$Q_2 = E + r_2\underline{T_2}$

Since triangle MCE is a right triangle, the angle ω is equal to the tangent of c/r. Knowing this angle, point M on circle C is defined as the vector in the direction CQ_2, with a magnitude of c, rotated ω radians to the right:

$M = C + c\underline{CQ_2}°(\tan^{-1}(r_2/c))$



We then calculate the blue and green arcs by drawing the arc in the hats defined by the points B, Q, E and E, Q_2, M respectively.

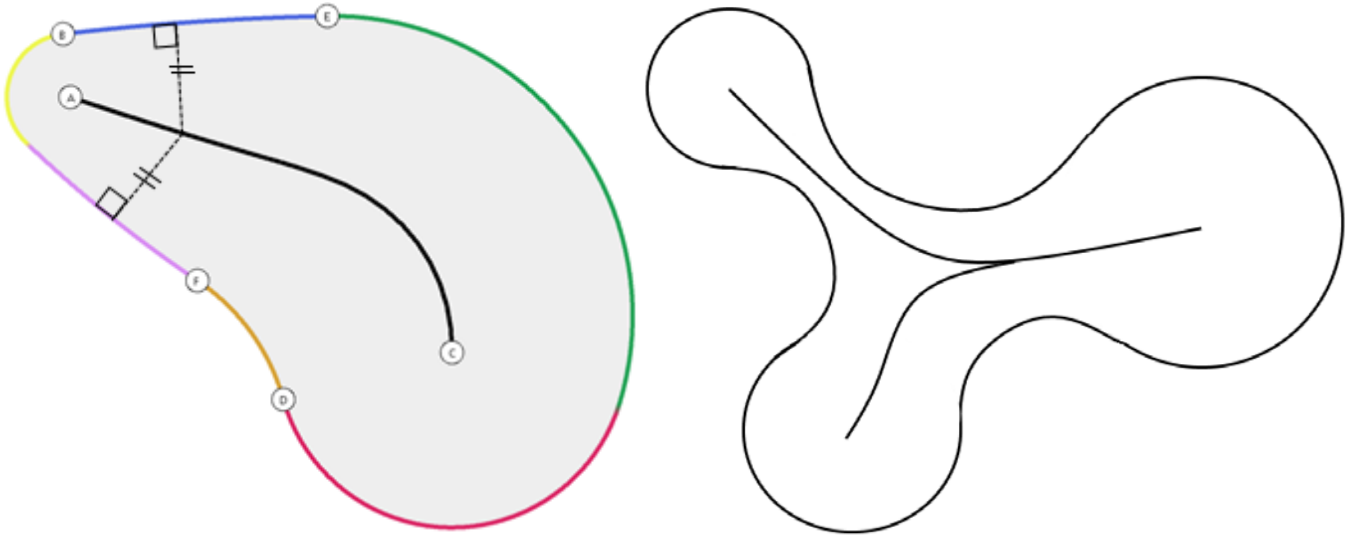
Then, we repeat the same vector calculations and formulas to draw the arcs from point D, through point F, and on to be tangential to circle A simply by replacing points B, E, and C with D, F, and A, respectively.

4 Medial Axis

The medial axis M of a stroke S can be considered as a sort of geometrical center of the shape. It is defined as the line segment such that the closest distance from a point on that segment to the boundary curve is the same whether you travel to the boundary above or below. For example, in the valid stroke below (left), the distance between that point and the boundary (in this case, the blue and purple arcs) is equivalent. These segments connecting the point on the medial axis to the boundary curves are normal to bS.

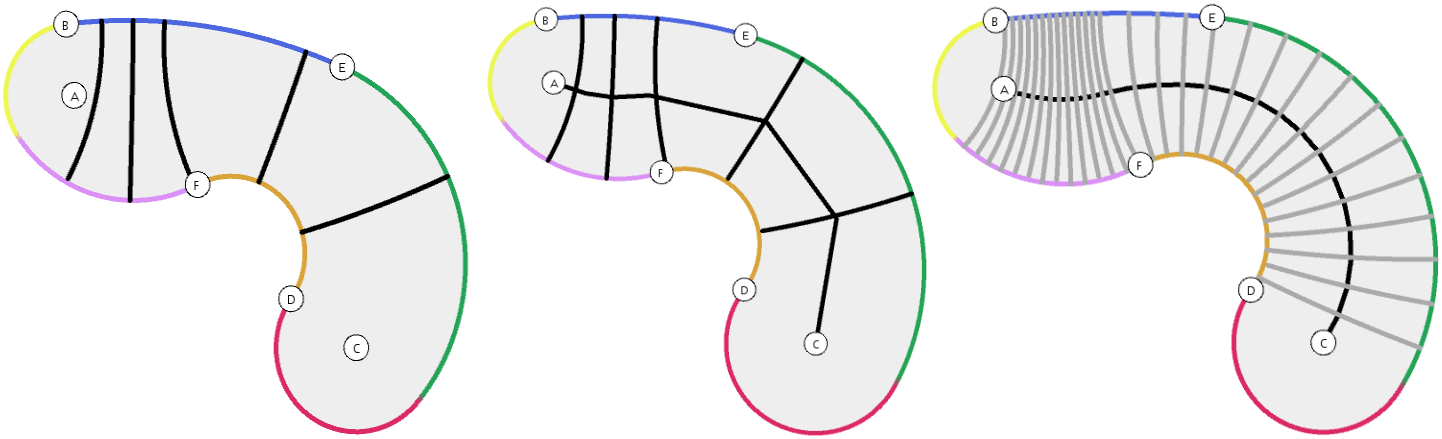
We restrict our solution to strokes for which the medial axis is a smooth curve segment without bifurcation.

We show below (left) a valid stroke and (right) an invalid stroke.



5 Computation of the Medial Axis

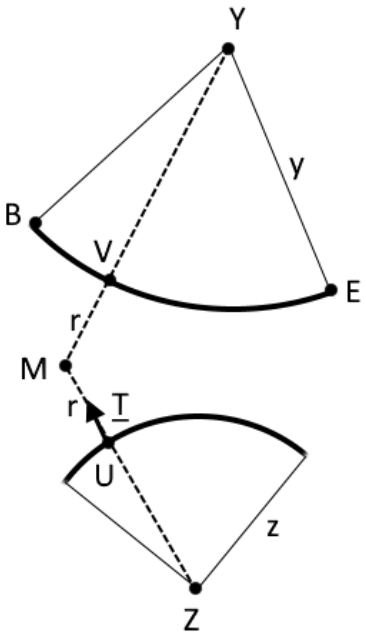
The medial axis M of a valid stroke S is a smooth curve composed of smoothly joined conic sections. To justify this claim, consider a decomposition of S into ***sectors*** such that each sector is bounded by 4 circular arcs, as shown in the diagram below (left). Since each point along the medial axis is equidistant from each of the two boundaries, which are smooth curves, the medial axis itself must also be a smooth curve. We approximated this curve instead as roughly uniform length line segments. The more points on the medial axis that we calculate will converge this segmented line towards a smooth curve.



To compute M, we start by defining four points, X, Y, Z, and W, that represent the centers of the circles that define each circular arc – blue, green, orange, and purple – respectively. Lengths x, y, z, and w, are the radii of these circles. Point X is defined as:

$X = x\underline{QE}°$ where $x = |QE| * \tan(QB\hat{}QE)$

Points Y, Z, and W are defined similarly.



We compute M by first computing points, U_i, along the orange and purple arcs that are a uniform arc-length apart, u, and then computing the point along the medial axis that is reached when you travel in a normal direction from the lower boundary of the stroke. Making u a constant arc length, means that the medial axis segments will not be uniform, but we chose this approximation as it takes into account the curvature of the purple arc versus the curvature of the orange arc, while still having defined, easily calculatable points, U_i. The details for finding the medial axis points M_i are as follows:

Assuming point U_i, point M_i can be represented in vector notation as:

$M_i = U_i + r\underline{T}$

The length of vector YM_i is equivalent to y + r, so squaring both sides to prevent rounding errors we get:

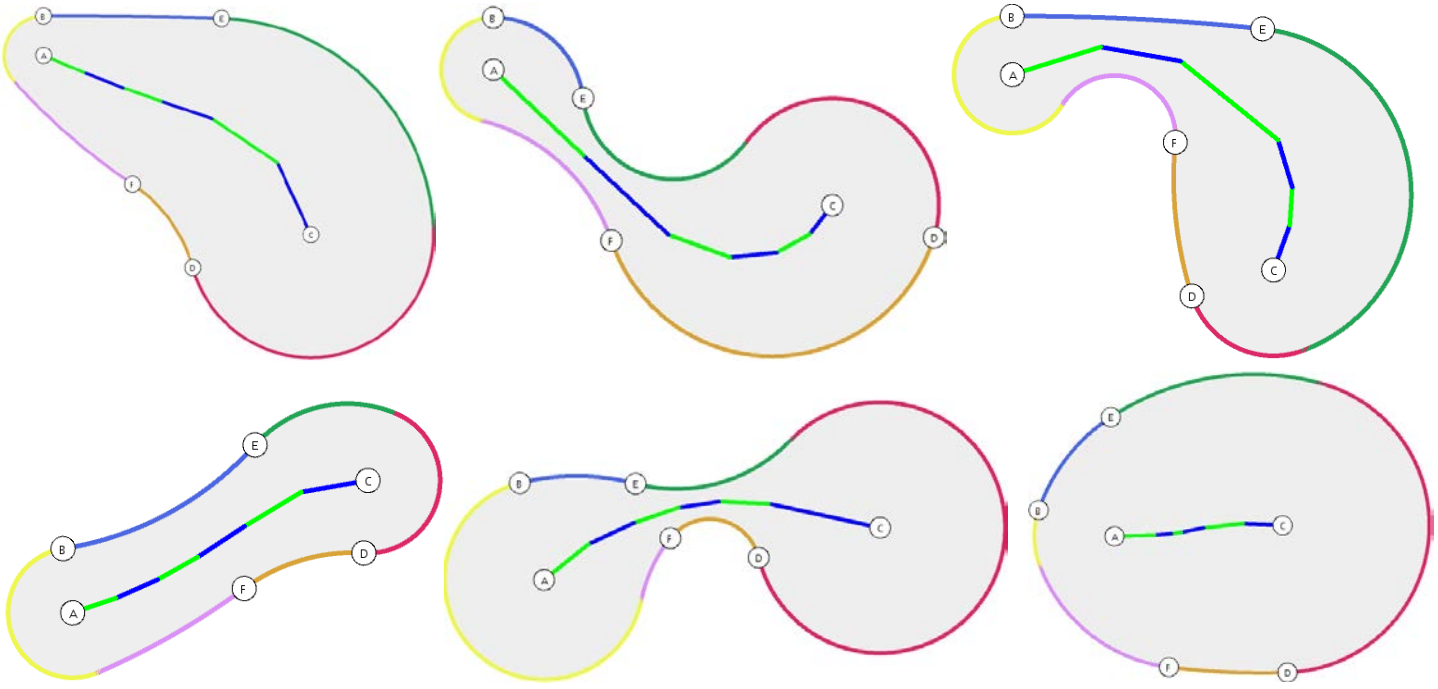
$|YM_i|^2 = (y + r)^2$, and substituting in for $M_i = U_i + r\underline{T}$, we can solve for r and get the following equation:

$r = y^2 - |YU_i|^2 / 2*(M_iU_i \cdot \underline{T} - y)$

Once r is calculated, we check to make sure that the closest point on the other arc, V_i, is in fact being drawn as part of our stroke, on the blue arc between B and E. To do so, we travel from U_i to V_i, and make sure that the two closest vertices that we are drawing (B and E) are in different directions using the det product and seeing that their signs are opposite. In the example provided, $U_iV_i:V_iB$ is negative and $U_iV_i:V_iE$ is positive, so the point V_i is on the curve in between points B and E. If it is not on the blue arc, then we know it must be on the green arc instead, since we defined U_i to be on the orange or purple arc, and because the medial axis is bounded by points A and C on either end.

While the math remains the same, the vector of $\underline{T}$ will change direction based on whether the curve U started on is convex or concave in relation to the body the stroke. We take this into account while calculating $\underline{T}$ so that it always faces towards the medial axis in the center of the stroke.

Results are shown below (with the different segments of M color coded differently) for a set of 6 very different strokes.



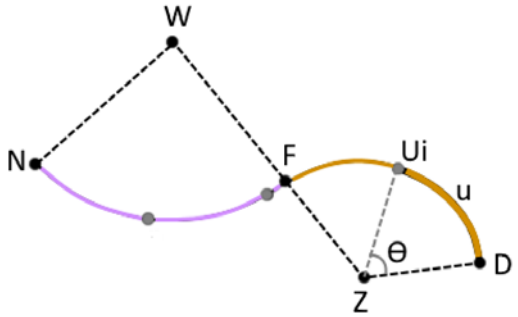
6 Transversals of a stroke

Given a sample M_i on M, the corresponding transversal T_i is a circular arc that starts and ends at points of bS and is tangent at these points to the normal vector to bS at these points.

We represent T_i by drawing an arc in the hat (U_i,M_i,V_i), all of which we inherently calculated while solving for point Mi.

We compute this representation as follows:

Point U_i is defined as one of n-1 equally spaced points along the purple and orange arcs combined. To find the arc length between them, u, we combine the arc lengths of the purple and orange arcs, and divide by n, the number of sectors. Then, knowing arc length, u, we can find the next point U_i, by rotating an angle of Θ radians counter-clockwise along the orange arc from point D. Θ for the orange arc is defined as u/z, with z being the radius of the circle that defines the orange arc in the diagram to the right. Θ for the purple arc is u/w, with w being the radius of the circle that defines the purple arc. Finally, we can represent U as:



$U_i = Z + z\underline{ZD}°(-Θ*i)$

for the points on the orange arc, and similarly:

$U_i = W + w\underline{WN}°(-Θ*i)$

for the points on the purple arc.

Once the point on the medial axis, M_i, is calculated as described in Section 5, we can then define point V_i (first assuming it is on the green arc) to be:

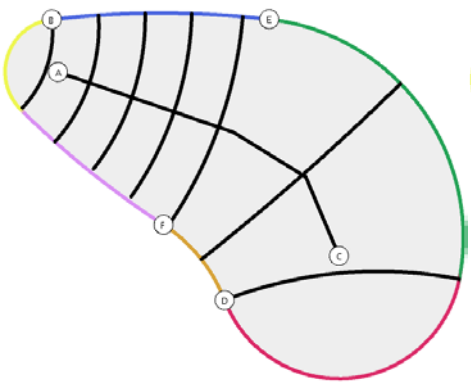
$V_i = Y + y\underline{YM_i}$

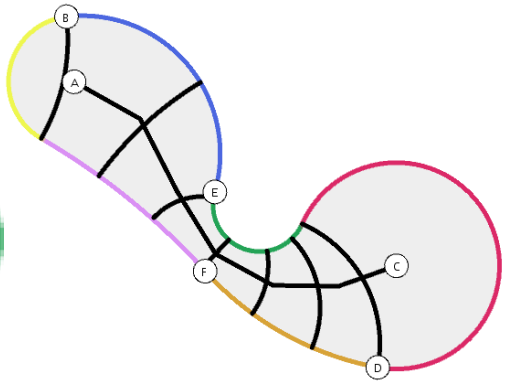
If our det calculation described in the previous section does not hold true, then the point V_i is not on the green arc and we recalculate M_i and V_i such that V_i is on the blue arc:

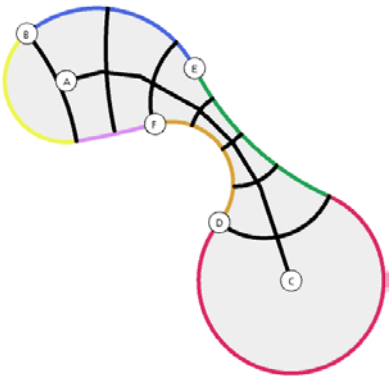
$V_i = X + x\underline{XM_i}$

We use this representation to draw T_i by using the procedure to draw a circular arc in a hat using poinst U_i, M_i, and V_i, which does the following. First it calculates the center of the circle for which U_i and V_i are on the circumference and U_iM_i and V_iM_i are both tangent to the circle. Then it draws the circumference for all angles between 0 and U_i^V_i radians.

We show below the set of transversals for several very different stokes.





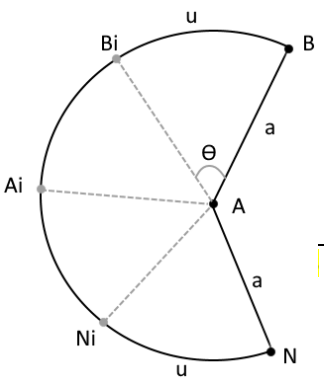


7 Extended transversals

We extend the stroke transversal construction past its medial axis as follows:

First, we calculate the angle Θ that represents how much we want to rotate along each circle's circumference to reach a point that is an arclength u away. For the circle defined by points A and B, Θ = u/a and for the circle defined by points C and D, Θ = u/c.

Then, we calculate the angles AN^AB and CM^CD. Since we always want the clockwise angle, in radians, between the two, we add 2*Pi to the result if the angle is calculated to be negative.



We compute these transversals as follows:

For the yellow arc defined by points A, B, and N, as shown to the right, we calculate new points B_i and N_i that are a distance u away from their respective points B and N. B_i and N_i are defined as:

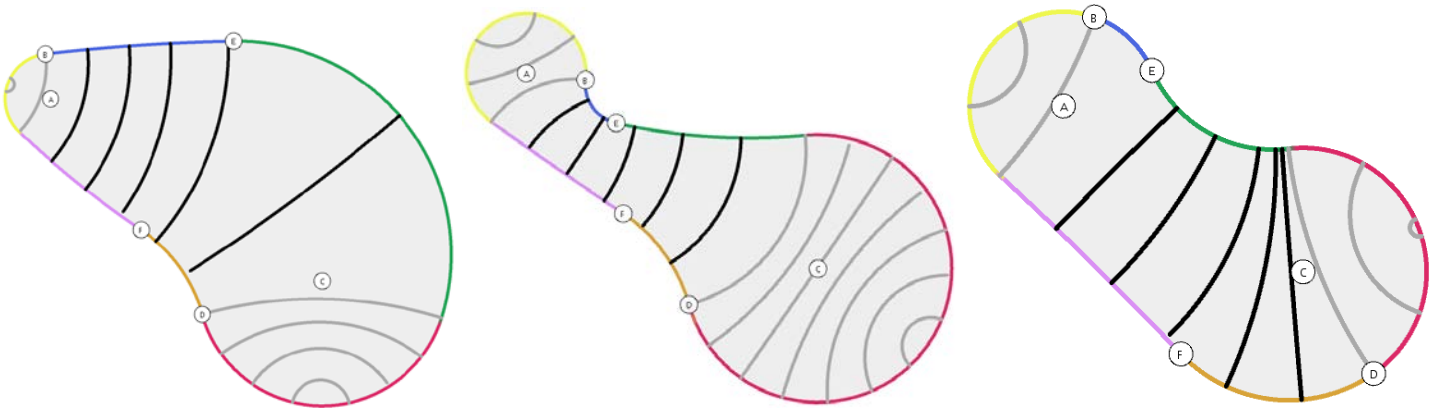
$B_i = A + AB°(- Θ*i)$ and $N_i = A + AN°(Θ*i)$.

If $2*Θ*i$ is greater than our angle AN^AB, then we stop drawing the extended transversals, as that means we have passed point A_i, which is defined as $A + AN°(AN^AB / 2)$ and marks the middle of the circular arc defined by B, A, and N.

For each B_i and N_i pair, we draw the circular arc in the hat, with point A as the top of the hat. Since B_i and N_i are both moving towards A_i, and the arcs are defined to be circular, these transversals will never cross over one another.

We repeat the same steps for the circle defined by points C, D, and M, to compute the additional transversals past point C.

We show below the set of natural transversals (discussed in the previous section) in black and the additional transversals in grey for several very different stokes.



8 Parameterization and quad mesh of a stroke

We parametrize the stroke G0 as follows:

G0.G[j*nC+i] = P((nC-i-1)*256/(nC-1),(nR-j-1)*256/(nR-1))

where, nC is the total number of traversals and extended traversals

nR is the total number of horizontal sections

0<=i<nC and 0<=j<nR

We assume that the image I is 256x256. We divide the 256x256 image into a grid of nC x nR and since we already drew the image I on the canvas, we directly take the points from the canvas and put them into our grid.

We parametrize the stroke G1 as follows:

For each transversal (natural and extended), we calculate and store nR points, including the two points U_i and V_i on boundary bS, such that each point is the same arclength apart on the curve. To do this, we define the center of the circle C_i to be:

$C_i = r\underline{M_iV_i}°$ where $r = |M_iV_i| * \tan(M_iU_i^{\wedge} M_iV_i)$

And then each point j on the transversal, stored as G1[j*nC+i], is:

$G1[j*nC+i] = C_i + C_iV_i°((C_iV_i ^{\wedge} C_iU_i / nR)*j)$

This populates the grid G1.

Next, we call the function *paintImage()* as: G1.paintImage(nC,nR,G0)

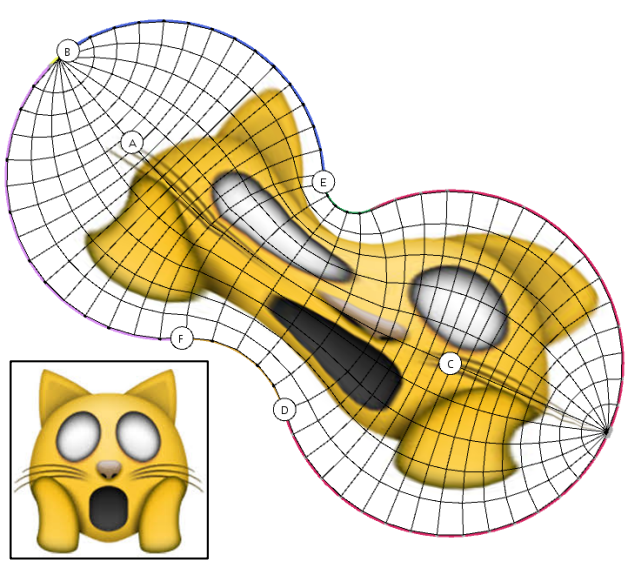
In this function, we iterate over nC and nR and create shapes using Quad Strips. The function *v()* maps the x and y coordinates of points in G1 to x and y coordinates of points in G0 respectively and marks an edge for the *beginShape()* method.

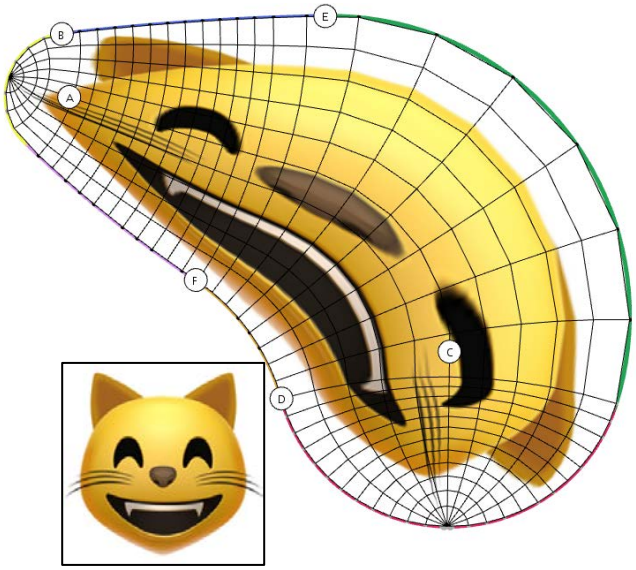
9 Texture transfer

To illustrate the power of strokes and of the mapping above, we show their application to Free-Form Deformation (FFD). We are given an image I that contains a cat emoticon, and initial stroke G_0 aligned with the 256x256 image to roughly lasso the face, and a final stroke G_1 over an empty canvas.

We use the local-to-global conversion of the vertices (crossings) of the natural quad mesh of G_0 to define their texture values (relative locations in I). We then use the local-to-global conversion of the vertices of the natural quad mesh of G_1 to display the quad moved and deformed natural quad mesh of G_1 using texture mapping of I.

Below, we show several examples of original images I, with overlaid lasso G_0 and moved and deformed cut-out defined by G_1.





10 Animation

To show the smoothness of our stroke construction when the control points are moved, we prescribe a different cyclic motion (with a different duration) to each control point of G1 and show (in the accompanying video) the animated FFD of the image cut-out.

To do this, we translate each control point by a value r_i and rotate it by w_i*t where r_i and w_i are randomly chosen vales and t is time.

To start animating, we press 'a' and '~' to save a particular frame. Once we have saved all the frames, we use a tool called 'VirtualDub' to load all the frames and save them in avi format.