Tenth Hungarian Conference on Computer Graphics and Geometry, Budapest, 2022

A circular parameterization for multi-sided patches

Péter Salvi

Budapest University of Technology and Economics

Abstract

Most genuine multi-sided surface representations depend on a 2D domain that enables a mapping between local parameters and global coordinates. The shape of this domain ranges from regular polygons to curved configurations, but the simple circular domain—to the best of our knowledge—has not been investigated yet.

Here we fill this gap, and introduce a parameter mapping ideal for use with periodic boundaries. It is based on circular arcs and satisfies constraints often needed in actual surface formulations. The proposed method is demonstrated through a corner-based variant of Generalized Bézier patches.

1. Introduction

The natural design of many models, from simple household objects to car bodies, often requires the use of multi-sided (non-quadrilateral) surface patches. This is generally solved either by creating a larger four-sided surface and cutting off the irrelevant parts (*trimming*), or by *splitting* the multi-sided region into smaller, four-sided subpatches.

Both of these approaches have their drawbacks; the alternative is to use genuine multi-sided patches. These non-standard surfaces can be defined by smoothly interpolating boundary constraints (*transfinite interpolation*), or by a network of control points.

Either way, a domain is needed. Although in some cases[3, 11] the domain does not take an explicit geometric form, it is usually a 2D shape. In the beginning it was assumed to be a regular n-sided polygon,[1] but later this gave place to arbitrary convex polygons,[8] concave polygons,[5] and even general curved domains.[10]

The rationale behind this progression is that a domain similar to the 3D configuration of the boundary curves reduces distortion. It is especially important to mimic the angle at the vertices where boundary curves meet. When this is equal to π, i.e., when the boundaries have parallel tangents, most existing methods cannot be used (but cf. splits[10] and periodic ribbons[6]).

In this paper we investigate a neglected option that seems tailored to these cases, the *circular* domain, and define a parameter mapping suitable for use in many multi-sided patch formulations.

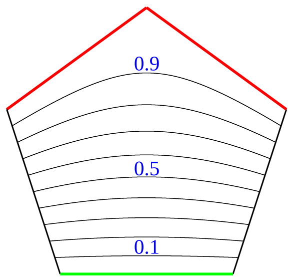


Figure 1: Constant parameter lines of a typical height parameter over a 5-sided polygonal domain. The base side is shown in green; distant sides in red.

2. Preliminaries

Parameterizations of 2D domains come in many varieties; in the context of multi-sided surfacing, these are used to map quadrilateral interpolants onto the domain. Depending on the patch representation, there may be several constraints on the parameters, as will be explained below.

The parameterization we are going to discuss in this paper is a distance or *height* parameter, usually denoted by h. It measures the distance of an interior point from a base side (see Figure 1), and needs to satisfy the following properties:

1. $h = 0$ on the base side.
2. h is continuous and varies monotonically.
3. h changes uniformly from 0 to 1 on the sides adjacent to the base side.

Such a mapping can be created for all possible base sides, resulting in n mappings, denoted by h_i ($i = 1 \dots n$). Depending on the application, there are often other constraints:

4. $h \leq 1$ everywhere inside the domain.
5. $h = 1$ on all distant sides, i.e., on all sides not equal or adjacent to the base side. [*full mapping*]
6. $h'_{i-1} = -h'_{i+1}$ on the ith side. [*constrained mapping*]

It is easy to see that (5) subsumes (4).

There have been parameterizations over other domains satisfying some of the above constraints; Salvi et al.,[4] in particular, showed how a mapping can be constructed that has all of these properties: there the parameterization itself is a (singular, 1D) multi-sided patch based on the work of Katō.[2]

Here we propose a height parameter that also satisfies all the constraints mentioned above, but over a circular domain.

3. Circular parameterization

A circular domain is, well, a circle—for ease of computation, let it be the unit circle around the origin. We allocate equal arcs to each side. Without loss of generality, let us assume that the base side is the $[-\pi/n, \pi/n]$ arc.

First we tackle the inverse problem, i.e., defining the constant parameter line for a given h value. We will use circular arcs for that purpose, with endpoints $\mathbf{p}_1$ and $\mathbf{p}_2$ uniformly ranging over the adjacent sides (property 3):

$$\mathbf{p}_1 = (\cos\varphi, \sin\varphi), \quad \mathbf{p}_2 = (\cos\varphi, -\sin\varphi) \tag{1}$$

where $\varphi = (2h+1)\pi/n$, see Figure 2.

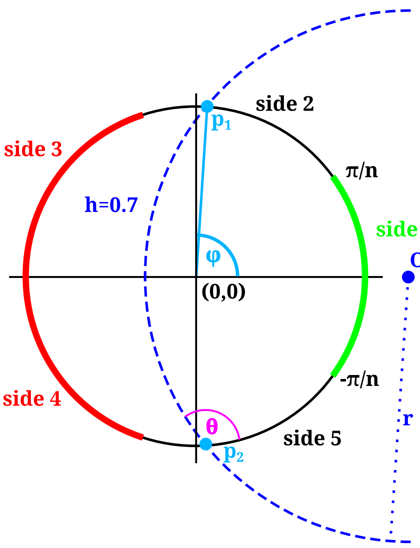


Figure 2: Notations in the canonical position, shown on a 5-sided configuration. Side 1 (green) is the base side, while sides 3 and 4 are the distant sides.

The idea is that we determine the circular arc by its angle to the domain circle (θ), which we require to change from 0 to π as h progresses from 0 to 1, i.e., $\theta = h\pi$. This means that the arc will be part of the domain circle when $h = 0$ or $h = 1$. A constant parameter line defined like this obviously satisfies all properties, including (5) and (6).

Straightforward algebra leads to the equations for the center point and radius of the constant parameter line:

$$\mathbf{O} = \left(\frac{\sin\theta}{\sin\psi}, 0\right), \quad r = \left|\frac{\sin\varphi}{\sin\psi}\right|, \tag{2}$$

where $\psi = \theta - \varphi = h\pi - \varphi$. (For details, see Appendix A.)

3.1. Mapping algorithm

Mapping a $\mathbf{p} = (u, v)$ point in the circle to a h parameter unfortunately leads to an equation of degree $2n$. Instead, we propose a procedural algorithm using bisection. It uses the fact that points with the coordinate $\hat{u} = \cos(\pi/(n-2))$ are on a *straight* constant parameter line (a circle with infinite radius), associated with $\hat{h} = 1/(n-2)$.

if $u > \hat{u}$ **then**
| **return** `bisection` $(\Delta, 0, \hat{h} - \varepsilon)$
if $u < \hat{u}$ **then**
| **return** `bisection` $(\Delta, \hat{h} + \varepsilon, 1)$
return $\hat{h}$

Here `bisection` searches for a value between its second and third arguments where the deviation vanishes. The deviation is given as the function $\Delta(h)$ shown below:

$$\Delta(h) = \|\mathbf{p} - \mathbf{O}(h)\| - r(h). \tag{3}$$

This is a fast and robust algorithm, which is very easy to implement; see Appendix B.

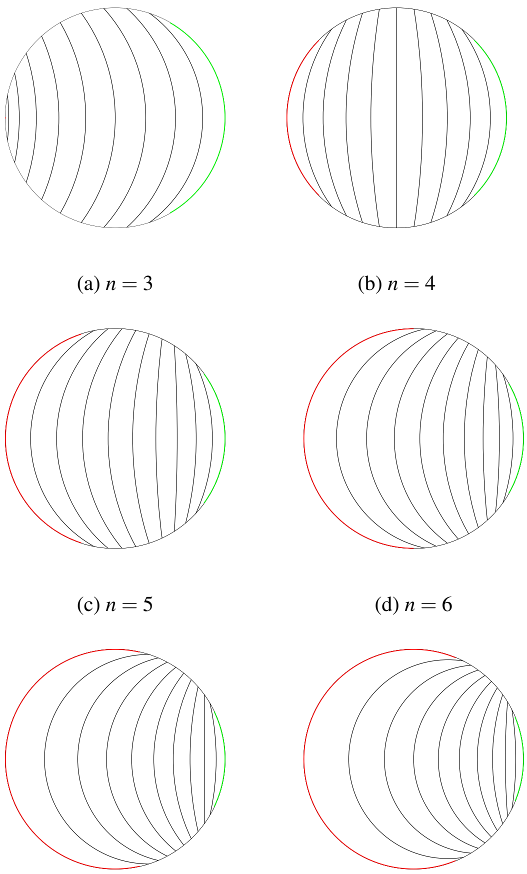
3.2. Examples

Parameterizations of 3- to 8-sided domains are shown in Figure 3. Taking parameterizations associated with adjacent base sides (i.e., h_{i-1} and h_i), we get a bivariate mapping (Figure 4), suitable for the parameterization of corner interpolants, as will be demonstrated in the next section.

The derivative constraint is exemplified in Figure 5, where the constant parameter lines of h_{i-1} and h_{i+1} have the same tangent at the ith side.

4. Application to Overlap-GB (OGB) patches

The Overlap patch[7] is a corner-based formulation, where the surface is defined as the sum of corner interpolants. Each interpolant is given by its corner vertex, two derivatives and a twist vector. The parameterization used in the original paper was a G^1-continuous composite function, but it can be replaced by our circular mapping.



(a) $n = 3$ (b) $n = 4$

(c) $n = 5$ (d) $n = 6$

(e) $n = 7$ (f) $n = 8$

Figure 3: Constant parameter lines on domains with 3 to 8 “sides”. The base side is shown in green; distant sides in red.

Generalizing the Overlap patch to arbitrary degrees leads to a formulation very similar to the Generalized Bézier (GB) patch,[9] but without the rational weight functions. Since patch formulations are not the topic of this paper, we only show the required equations here:

$$S = \sum_{i=1}^{n} \sum_{j=0}^{\lfloor d/2 \rfloor} \sum_{k=0}^{\lfloor d/2 \rfloor} \mathbf{P}_{ijk} B_j^d(h_{i+1}) B_k^d(h_i) + \mathbf{P}_0 B_0, \qquad (4)$$

where $\mathbf{P}_{ijk}$ are points constituting a control network, $B_\ell^d(h)$ are degree-d Bernstein polynomials, $\mathbf{P}_0$ is a central control point, and B_0 is the weight deficiency:

$$B_0 = 1 - \sum_{i=1}^{n} \sum_{j=0}^{\lfloor d/2 \rfloor} \sum_{k=0}^{\lfloor d/2 \rfloor} B_j^d(h_{i+1}) B_k^d(h_i). \qquad (5)$$

Here the degree d is assumed to be odd; it is easy to modify the above equations to allow even values, as well.

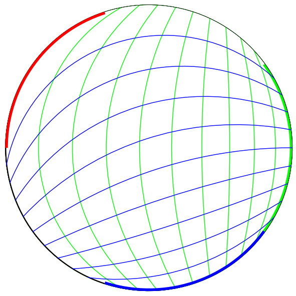


Figure 4: Corner parameterization for a 5-sided patch. The base sides and their respective constant parameter lines are shown in green and blue; the distant side is shown in red.

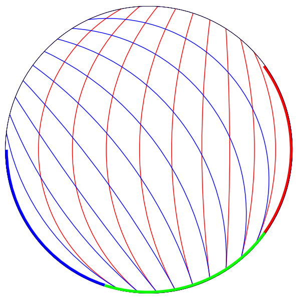


Figure 5: Demonstration of the derivative constraint on a 5-sided patch. The base sides and their respective constant parameter lines are shown in red and blue; the lines start in the same direction near the green arc.

Note that interpolation of the boundary constraints requires a full, constrained h-mapping. Figure 6 shows a 5-sided cubic patch based on the above equations, using circular parameterization. From the tessellation it can be seen that no distortion has been introduced, and the isophote lines also flow naturally.

Conclusion

We have investigated the simple circle as a candidate for a multi-sided domain, along with a distance parameter mapping satisfying all properties commonly needed for use in n-sided patches.

The proposed parameterization is perfect for handling periodic boundaries, and while it cannot be evaluated explicitly, it is both conceptually and algorithmically simpler than previous solutions.[4, 6]

Future work includes the investigation of derivative behavior in the corners.

Acknowledgements

The author would like to thank Tamás Várady and Márton Vaitkus for the many fruitful discussions on this topic.

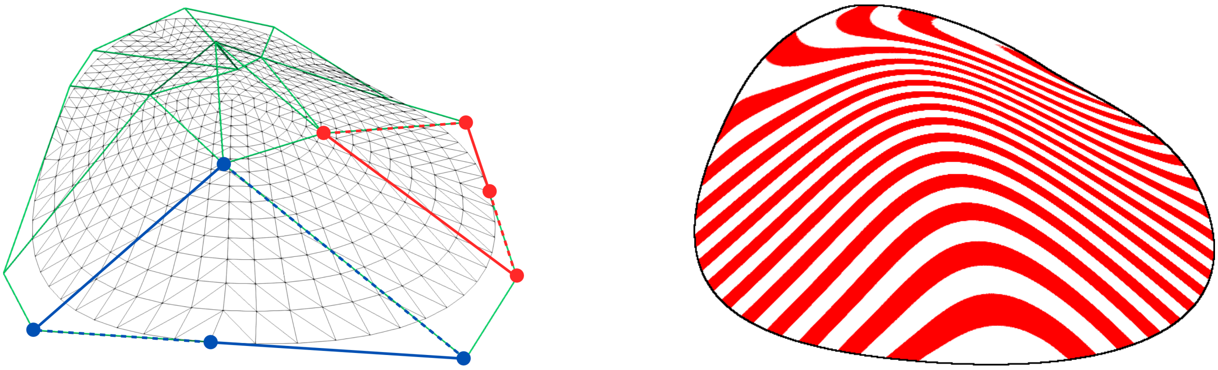


Figure 6: A 5-sided cubic OGB patch (*left*: control network highlighting two corner interpolants; *right*: isophote lines).

References

1. Peter Charrot and John A Gregory. A pentagonal surface patch for computer aided geometric design. *Computer Aided Geometric Design*, 1(1):87–94, 1984.
2. Kiyokata Kato. Generation of *n*-sided surface patches with holes. *Computer-Aided Design*, 23(10):676–683, 1991.
3. Malcolm Sabin. Non-rectangular surface patches suitable for inclusion in a B-spline surface. In *Proceedings of the Eurographics '83 Conference*, pages 57–70. Eurographics, 1983.
4. Péter Salvi, István Kovács, and Tamás Várady. Computationally efficient transfinite patches with fullness control. In *Proceedings of the Workshop on the Advances of Information Technology*, pages 96–100. BME, 2017.
5. Péter Salvi and Tamás Várady. Multi-sided Bézier surfaces over concave polygonal domains. *Computers & Graphics*, 74:56–65, 2018.
6. Márton Vaitkus, Tamás Várady, Péter Salvi, and Ágoston Sipos. Multi-sided B-spline surfaces over curved, multi-connected domains. *Computer Aided Geometric Design*, 89:102019, 2021.
7. Tamás Várady. Overlap patches: A new scheme for interpolating curve networks with *n*-sided regions. *Computer Aided Geometric Design*, 8(1):7–27, 1991.
8. Tamás Várady, Alyn Rockwood, and Péter Salvi. Transfinite surface interpolation over irregular *n*-sided domains. *Computer-Aided Design*, 43(11):1330–1340, 2011.
9. Tamás Várady, Péter Salvi, and György Karikó. A multi-sided Bézier patch with a simple control structure. *Computer Graphics Forum*, 35(2):307–317, 2016.
10. Tamás Várady, Péter Salvi, Márton Vaitkus, and Ágoston Sipos. Multi-sided Bézier surfaces over curved, multi-connected domains. *Computer Aided Geometric Design*, 78:101828, 2020.
11. Jinjin Zheng and Alan A Ball. Control point surfaces over non-four-sided areas. *Computer Aided Geometric Design*, 14(9):807–821, 1997.

Appendix A: Proof of Eq. (2).

The tangent to the unit circle at $\mathbf{p}_2 = (\cos\varphi, -\sin\varphi)$ is

$$\mathbf{t} = (\sin\varphi, \cos\varphi), \tag{6}$$

which we rotate by the angle θ to get the tangent to the constant parameter arc:

$$\begin{aligned}\mathbf{d} &= (\sin\varphi, \cos\varphi)\begin{pmatrix}\cos\theta & \sin\theta\\ -\sin\theta & \cos\theta\end{pmatrix}\\ &= (\sin(\varphi-\theta), \cos(\varphi-\theta)).\end{aligned} \tag{7}$$

This is perpendicular to the radius from $\mathbf{O} = (x, 0)$, i.e.,

$$\langle \mathbf{p}_2 - \mathbf{O}, \mathbf{d}\rangle = 0, \tag{8}$$

which leads to

$$(\cos\varphi - x)\sin(\varphi-\theta) = \sin\varphi\cos(\varphi-\theta). \tag{9}$$

Using the fact that

$$\sin\varphi\cos(\varphi-\theta) - \cos\varphi\sin(\varphi-\theta) = \sin\theta, \tag{10}$$

we arrive at

$$x = -\frac{\sin\theta}{\sin(\varphi-\theta)} = \frac{\sin\theta}{\sin(\theta-\varphi)} = \frac{\sin\theta}{\sin\psi}. \tag{11}$$

The radius can be computed by

$$\begin{aligned} r = \|\mathbf{p}_1 - \mathbf{O}\| &= \left\|\left(\cos\varphi - \frac{\sin\theta}{\sin\psi}, \sin\varphi\right)\right\| \\ &= \sqrt{\sin^2\varphi + \cos^2\varphi - 2\cos\varphi\frac{\sin\theta}{\sin\psi} + \frac{\sin^2\theta}{\sin^2\psi}} \\ (\text{using } \theta = \varphi + \psi) \quad &= \sqrt{\frac{\sin^2\varphi}{\sin^2\psi}} = \left|\frac{\sin\varphi}{\sin\psi}\right|. \;\square \end{aligned} \tag{12}$$

Appendix B: Computation of the height map.

The following JULIA program computes the h mapping of a given point, which is first rotated to the canonical position, i.e., where the base side is the $[-\pi/n, \pi/n]$ arc.

```
"""
    height(n, i, p)

Height parameter mapping of `p` on
an `n`-sided circular domain, based on
the `i`-th side (`1 ≤ i ≤ n`).
"""
function height(n, i, p)
    p = rotation((i - 1) * 2π / n) * p
    θ(h) = h * π
    φ(h) = (2h + 1) * π / n
    ψ(h) = θ(h) - φ(h)
    O(h) = [sin(θ(h)) / sin(ψ(h)), 0]
    r(h) = abs(sin(φ(h)) / sin(ψ(h)))
    Δ(h) = norm(p - O(h)) - r(h)
    h0 = 1 / (n - 2)
    u0 = cos(h0 * π)
    if p[1] > u0
        bisect(Δ, 0, h0)
    elseif p[1] < u0
        bisect(Δ, 1, h0)
    else
        h0
    end
end
```

Note that the above code needs the `LinearAlgebra` module of the standard library. The bisection algorithm is implemented as below.

```
"""
    bisect(f, a, b, ϵ)

Searches a root of `f` in the `[a,b)` interval
using bisection (`a > b` is a valid inverted
interval). The iteration exits when the error
becomes less than ϵ.

Note that `f(b)` is never evaluated.
"""
function bisect(f, a, b, ϵ = 1e-7)
    fa = f(a)
    abs(fa) < ϵ && return a
    while true
        x = (a + b) / 2
        fx = f(x)
        abs(fx) < ϵ && return x
        if fa * fx < 0
            b = x
        else
            a, fa = x, fx
        end
    end
end
```

Finally, the rotation matrix is computed simply by:

```
"""
    rotation(α)

Matrix of a rotation around the origin by `α`.
"""
rotation(α) = [cos(α) -sin(α); sin(α) cos(α)]
```

A bitmap generated by this code is shown in Figure 7.

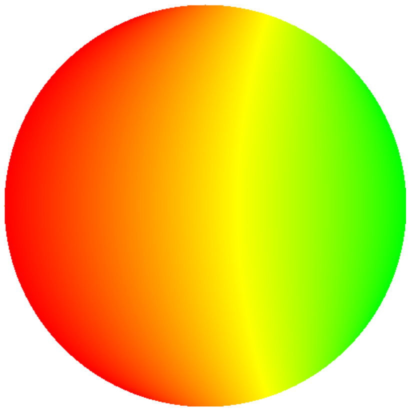


Figure 7: A bitmap showing h-values on a 5-sided domain (green: $h = 0$, yellow: $h = 0.5$, red: $h = 1$).