A Low Distortion Map Between Disk and Square

Peter Shirley
Computer Science Department
3190 MEB
University of Utah
Salt Lake City, UT 84112
shirley@cs.utah.edu

Kenneth Chiu
Computer Science Department
Lindley Hall
Indiana University
Bloomington, IN 47405
chiuk@cs.indiana.edu

Abstract

We present a map between squares and disks that associates concentric squares with concentric circles. This map preserves adjacency and fractional area, and has proven useful in many sampling applications where correspondences must be maintained between the two shapes. We also provide code to compute the map that minimizes branching and is robust for all inputs. Finally, we extend the map to the hemisphere. Though this map has been used before in publications, details of its computation have never previously been published.

Background

Many graphics applications map points on the unit square $\mathcal{S} = [0,1]^2$ to points on the unit disk $\mathcal{D} = \{(x,y) \mid x^2 + y^2 \leq 1\}$. Sampling disk-shaped camera lenses is one such application. Though each application can have its own unique requirements, experience has shown that a good, general-purpose map should have the following properties.

Preserve fractional area. Let $R \in M$ be a region in set M. The fractional area of R is defined as $a(R)/a(M)$, where the function a denotes area. Then a map $m : \mathcal{S} \rightarrow \mathcal{D}$ preserves fractional area if the fractional area of R is the same as the fractional area $a(m(R))/a(\mathcal{D})$ for all $R \in \mathcal{S}$ (Figure 1). This property ensures that a "fair" set of points on the square will map to a fair set on the disk. For distribution ray tracers this means a jittered set of points on the square will transform to a jittered set of points on the disk.

Bicontinuous. A map is *bicontinuous* if the map and its inverse are both continuous. Such a map will preserve adjacency; that is, points that are close on the disk came from points that are close on the square, and vice versa. This property is useful primarily when working on the hemisphere. It is necessary, for example, if we wish to use linear distance on the square as an estimate of angular distance on the hemisphere.

Low distortion. By low distortion, we mean that shapes are reasonably well preserved. Defining distortion more formally is possible [1], but probably of limited benefit because no single definition is clearly suitable for a wide range of applications.

Figure 2 shows the *polar map*, probably the most obvious way to map the square to the disk: r is a

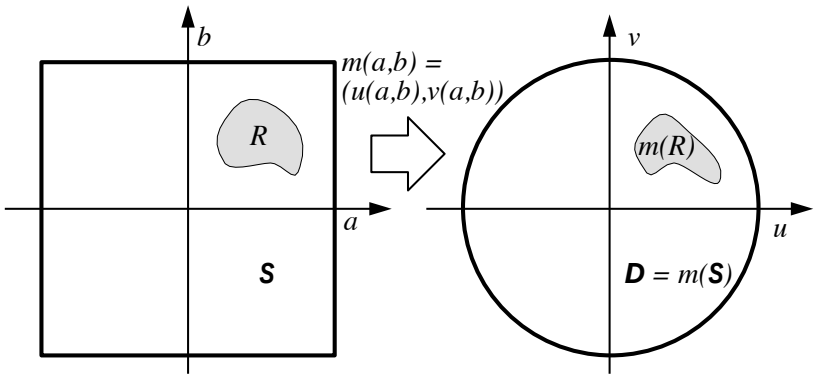

Figure 1: *A map that preserves fractional area will map regions with the area constraint that* $a(R)/a(\mathcal{S}) = a(m(R))/a(\mathcal{D})$.

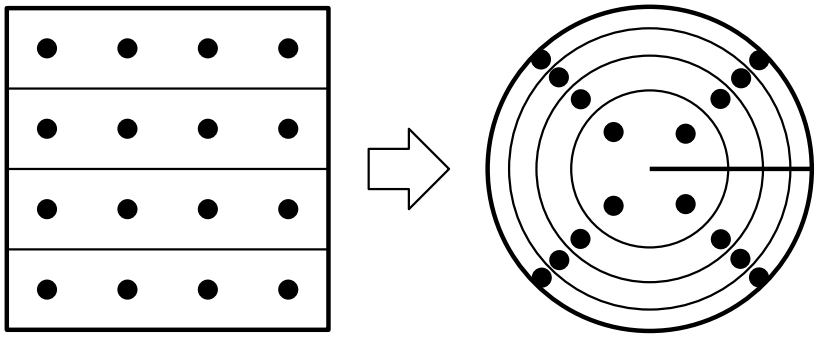

Figure 2: *The polar map takes horizontal strips to concentric rings*.

function of x and ϕ is a function of y. This will map vertical strips on $\mathcal{S}$ to rings on $\mathcal{D}$. Because an outer ring will have greater area than an inner ring of equivalent "width," r cannot be a linear function of x:

$$\begin{aligned} r &= \sqrt{x} \\ \phi &= 2\pi y \end{aligned}$$

This map preserves fractional area, but does not satisfy our other properties. As can be seen in the image, shapes are grossly distorted. Another problem is that although $(r, \phi) = (\sqrt{x}, 2\pi y)$ is continuous, the inverse map is not. For some applications we would prefer a bicontinuous map.

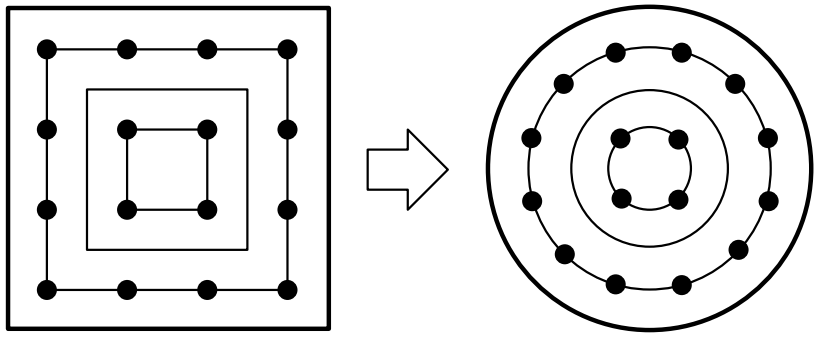

Figure 3: *The concentric map takes concentric square strips to concentric rings*.

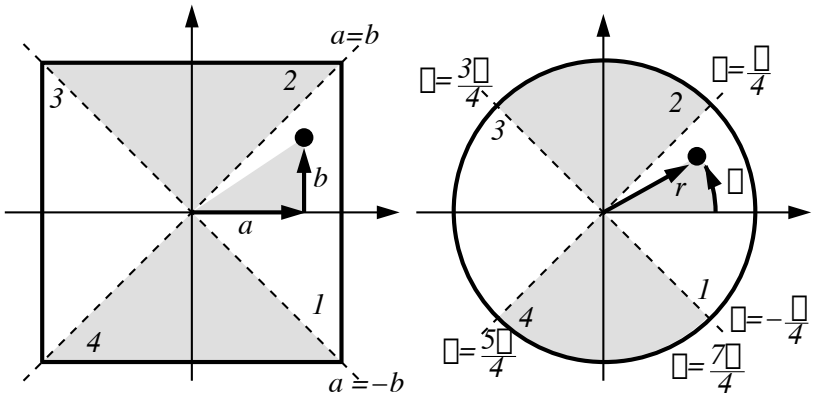

Figure 4: *Quantities for mapping region 1.*

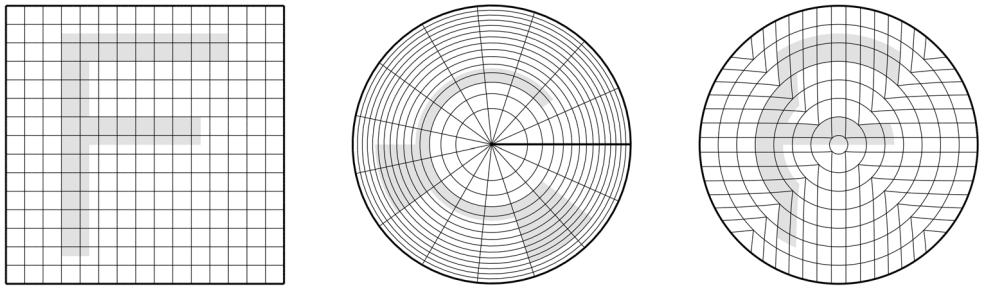

Figure 5: *Left: unit square with gridlines and letter "F". Middle: polar mapping of gridlines and "F". Right: concentric mapping of gridlines and "F".*

The Concentric Map

We now present the *concentric map*, which maps concentric squares to concentric circles, as shown in Figure 3. This map has the properties listed above, and is easy to compute. It has been advocated for ray tracing applications [3] and has been shown empirically to provide lower error for stochastic camera lens sampling [2], but the details of its computation have not been previously published.

The algebra behind the map is illustrated in Figure 4. The square is mapped to $(a, b) \in [-1, 1]^2$, and is divided into four regions by the lines $a = b$ and $a = -b$. For the first region, the map is:

$$\begin{aligned} r &= a \\ \phi &= \frac{\pi}{4}\frac{b}{a} \end{aligned}$$

This produces an angle $\phi \in [-\pi/4, \pi/4]$. The other four regions have analogous transforms. We show in the Appendix that this map preserves fractional area. How the polar map and concentric map transform the connected points in a uniform grid is shown in Figure 5. Note that the "F" is still recognizable in the concentric map, but has been grossly distorted by the polar map.

This concentric square to concentric circle map can be implemented in a small piece of robust code which has been written to minimize branching:

```
Vector2 ToUnitDisk( Vector2 onSquare )
   real phi, r, u, v
   real a = 2*onSquare.X-1      // (a,b) is now on [-1,1]^2
   real b = 2*onSquare.Y-1

   if (a > -b)                  // region 1 or 2
       if (a > b)               // region 1, also |a| > |b|
           r = a
           phi = (PI/4 ) * (b/a)
       else                     // region 2, also |b| > |a|
           r = b;
           phi = (PI/4) * (2 - (a/b))
   else                         // region 3 or 4
       if (a < b)               // region 3, also |a| >= |b|, a != 0
            r = -a
            phi = (PI/4) * (4 + (b/a))
       else // region 4, |b| >= |a|, but a==0 and b==0 could occur.
            r = -b
            if (b != 0)
                phi = (PI/4) * (6 - (a/b))
            else
                phi = 0

   u = r * cos( phi )
   v = r * sin( phi )
   return Vector2( u, v )
end
```

The inverse map is straightforward provided the function `atan2()` is available. The code below assumes `atan2()` returns a number in the range $[-\pi, \pi]$, as it does in ANSI *C*, Draft ANSI *C++*, ANSI *FORTRAN*, and *Java*.

```
Vector2 FromUnitDisk(Vector2 onDisk)
    real r = sqrt(onDisk.X * onDisk.X + onDisk.Y * onDisk.Y)
    Real phi = atan2(onDisk.Y, onDisk.X );
    if (phi < -PI/4) phi += 2*PI    // in range [-pi/4,7pi/4]
    Real a, b, x, y
    if (phi < PI/4)                // region 1
        a = r
        b = phi * a / (PI/4)
    else if (phi < 3*PI/4 )        // region 2
        b = r
        a = -(phi - PI/2) * b / (PI/4)
    else if (phi < 5*PI/4)        // region 3
        a = -r
        b = (phi - PI) * a / (PI/4)
    else                           // region 4
        b = -r
        a = -(phi - 3*PI/2) * b / (PI/4)
    x = (a + 1) / 2
    y = (b + 1) / 2
    return Vector2(x, y)
end
```

The concentric map can be extended to the hemisphere by projecting the points up from the disk to the hemisphere. By controlling exactly how the points are projected, we can generate different distributions, as shown in the next section.

Applications

The concentric map can be used in a number of different ways.

- **Random point on a disk.** A random point on the disk of radius R can be generated by passing a random point on $[0, 1]^2$ through the map and multiplying both coordinates by R.

- **Jittered point on a disk.** A set of n^2 jittered points can be generated similarly using uniform random numbers between zero and one such as generated by a call to `drand48()`:

```
int s = 0;
for (int i = 0; i < n; i++)
   for (int j = 0; j < n; j++)
       Vector2  onSquare((i + drand48()) / n, (j + drand48()) / n )
       onDisk[s++] = ToUnitDisk(onSquare )
```

The jittered sampling produced by the concentric map tends to produce lower variance than the polar map, because the jitter cells stay relatively square instead of becoming long and thin. This means that the integrand will tend to have less variation within each jitter cell.

- **Cosine distribution on a hemisphere.** Radiosity and path tracing applications often need points on the unit hemisphere with density proportional to the z-coordinate of the point. This is commonly referred to as a *cosine distribution* because the z-coordinate of a point on a unit sphere is the cosine between the z-axis and the direction from the sphere's center to the point. Such a distribution is commonly generated from uniform points on the disk by projecting the points along the z-axis. Assuming (u, v) are the coordinates on the disk and letting $r = \sqrt{u^2 + v^2}$, the cosine-distributed point (x, y, z) on the hemisphere oriented along the positive z-axis is

$$\begin{aligned} x &= u \\ y &= v \\ z &= \sqrt{1 - r^2} \end{aligned}$$

- **Uniform distribution on a hemisphere.** Generating a uniform distribution of the z-coordinate will create a uniform distribution on the hemisphere. This can be accomplished by noting that if we have a uniform distribution on a disk, then the distribution of r^2 is also uniform. So given a uniformly distributed random point (u, v) on a disk, we can generate a uniformly distributed point on the hemisphere by first generating a z coordinate from r^2, and then assigning x and y such that the point is on the hemisphere [4].

$$\begin{aligned} x &= u\sqrt{2 - r^2} \\ y &= v\sqrt{2 - r^2} \\ z &= 1 - r^2 \end{aligned}$$

- **Phong-like distribution on a hemisphere.** The uniform and cosine distributions can be generalized to a Phong-like distribution where density is proportional to a power of the cosine, or z^N:

$$\begin{aligned} x &= u\frac{\sqrt{1 - z^2}}{r} \\ y &= v\frac{\sqrt{1 - z^2}}{r} \\ z &= \left(1 - r^2\right)^{\frac{1}{N+1}} \end{aligned}$$

  Note that the two previous formulas are special cases of this formula.

- **Subdivision data.** In any application where a subdivision data structure must be kept on the hemisphere or disk, the bicontinuous and low-distortion properties make the map ideal. For example, a *k-d tree* organizing points on the disk can be kept on the transformed points on the square so that conventional axis-parallel two-dimensional divisions can be used.

Appendix

When the concentric map is expanded out so that we map from $(a,b) \in [-1,1]^2$ to (u,v) on the unit disk, the map is:

$$\begin{aligned} u &= a \cos\left(\frac{\pi b}{4a}\right) \\ v &= a \sin\left(\frac{\pi b}{4a}\right) \end{aligned}$$

The ratio of differential areas in the range and domain for the map is given by the determinant of the Jacobian matrix. If this determinant is the constant ratio of the area of the entire domain to the entire range, then the map preserves fractional area as is desired. This is in fact the case for the concentric map:

$$\left| \begin{array}{cc} \frac{\partial u}{\partial a} & \frac{\partial u}{\partial b} \\ \frac{\partial v}{\partial a} & \frac{\partial v}{\partial b} \end{array} \right| = \left| \begin{array}{cc} \cos\left(\frac{\pi b}{4a}\right) + \frac{\pi b}{4a}\sin\left(\frac{\pi b}{4a}\right) & -\frac{\pi}{4}\sin\left(\frac{\pi b}{4a}\right) \\ \sin\left(\frac{\pi b}{4a}\right) - \frac{\pi b}{4a}\cos\left(\frac{\pi b}{4a}\right) & \frac{\pi}{4}\cos\left(\frac{\pi b}{4a}\right) \end{array} \right| = \frac{\pi}{4}$$

This is the constant $\pi/4$ because that is the ratio of the area of the unit disk to the area of $[-1,1]^2$. By symmetry, the determinant of the Jacobian matrix is the same for the other three regions.

Acknowledgements

Thanks to Ronen Barzel, Eric Haines, Eric Lafortune, Bill Martin, Simon Premoze, Peter-Pike Sloan, and Brian Smits for comments on the paper. Figure 5 was generated using software by Michael Callahan and Nate Robins. This work was done with support from NSF grant ASC- 9720192.

References

[1] Frank Canters and Hugo Decleir. *The World in Perspective*. Bath Press, Avon, Great Britain, 1989.

[2] Craig Kolb, Pat Hanrahan, and Don Mitchell. A realistic camera model for computer graphics. In Rob Cook, editor, *Proceedings of SIGGRAPH '95 (Anaheim, California, August 6–11, 1995)*, Computer Graphics Proceedings, Annual Conference Series, pages 317–324, August 1995.

[3] Peter Shirley. A ray tracing method for illumination calculation in diffuse-specular scenes. In *Proceedings of Graphics Interface '90*, pages 205–212, May 1990.

[4] Peter Shirley. Nonuniform random point sets via warping. In David Kirk, editor, *Graphics Gems III*, pages 80–83. Academic Press, San Diego, 1992. summary of various useful sample generation transformations.