Visualizing the escape paths of quaternion fractals

S. Halayka*

September 18, 2018

Abstract

The escape paths of the points in some quaternion fractal sets are visualized using OpenGL. C++ source code is provided.

1 Escape paths that do not reach infinity

As discussed in [1], a 3D scalar field of quaternion magnitudes (e.g. $|Z|$) results from calculating a quaternion fractal set when using a finite 3D lattice of regularly spaced points as input.

Here we will visualize the escape paths, using OpenGL, for those points that maintain a quaternion magnitude less than the infinity threshold value (e.g. 4.0) during the iteration process (e.g. 8 iterations).

The notion that the escape paths can make for nice visualizations was independently found by Paul Bourke [2]. Bourke's work was inspired by the Buddhabrot. Bourke's work focuses on the escape paths that reach infinity, whereas here we will focus on escape paths that do not reach infinity.

Here we will use Bezier curves and cylinders to draw the escape paths. The length of these 'shaggy' escape paths will be shortened to only 20% of the total length, giving the impression of an isotropic buzz cut.

The C++ code for this paper can be found at [3]. Ultra high definition versions of the figures given in this paper can be found at [4].

*sjhalayka@gmail.com

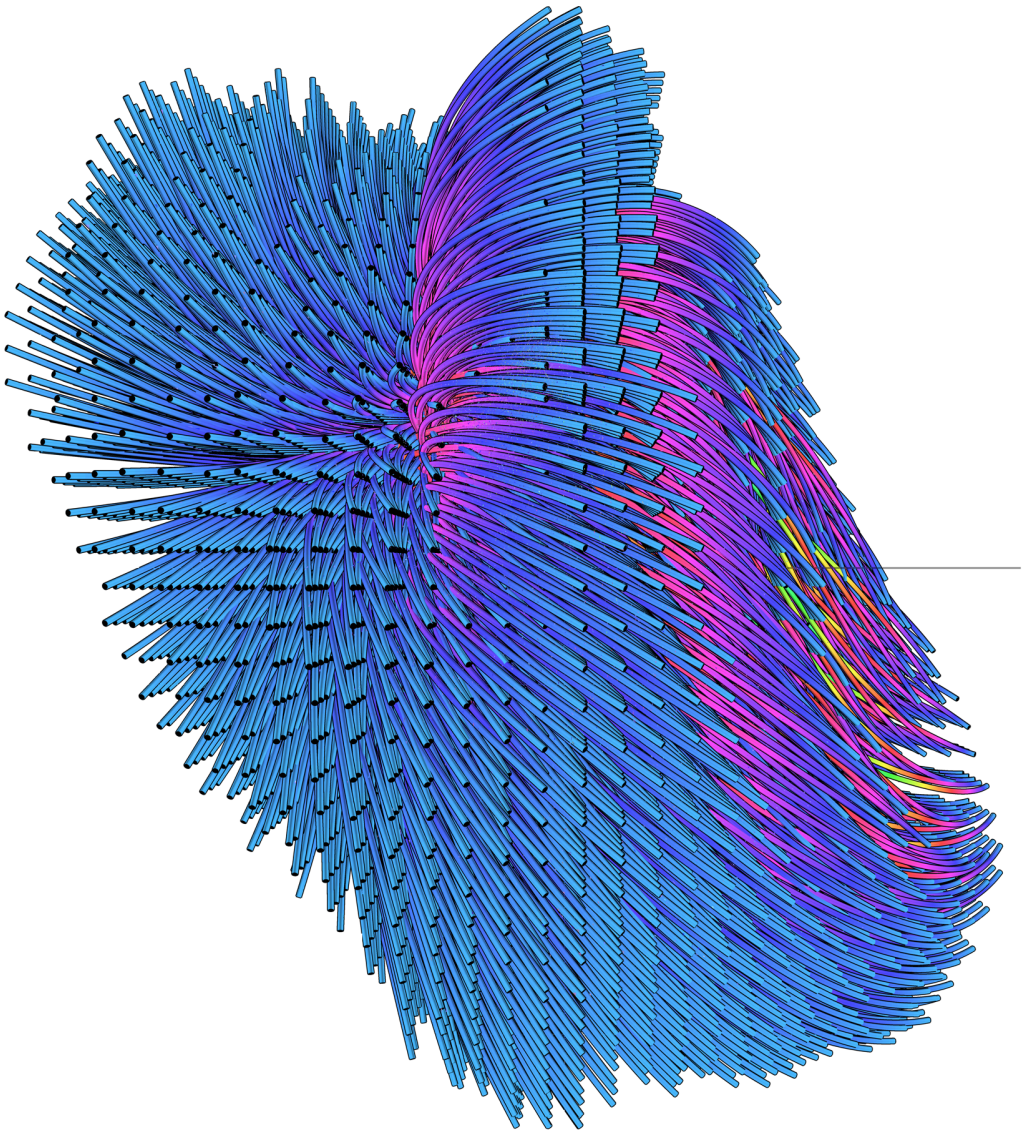

Figure 1: $Z' = Z^2 + C$, where $C_{xyzw} = 0.3, 0.5, 0.4, 0.2$.

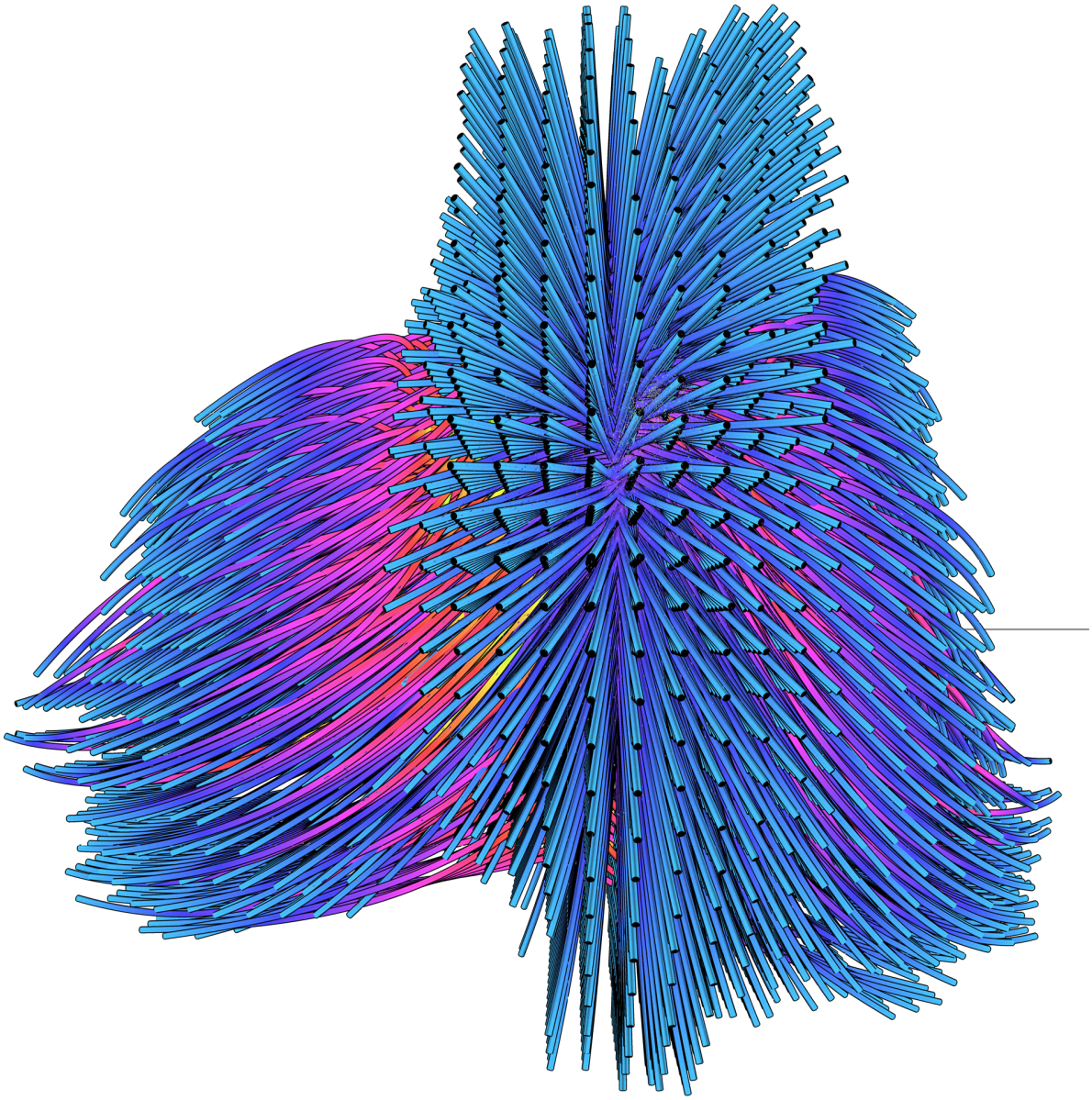

Figure 2: $Z' = Z^3 + C$, where $C_{xyzw} = 0.3, 0.5, 0.4, 0.2$.

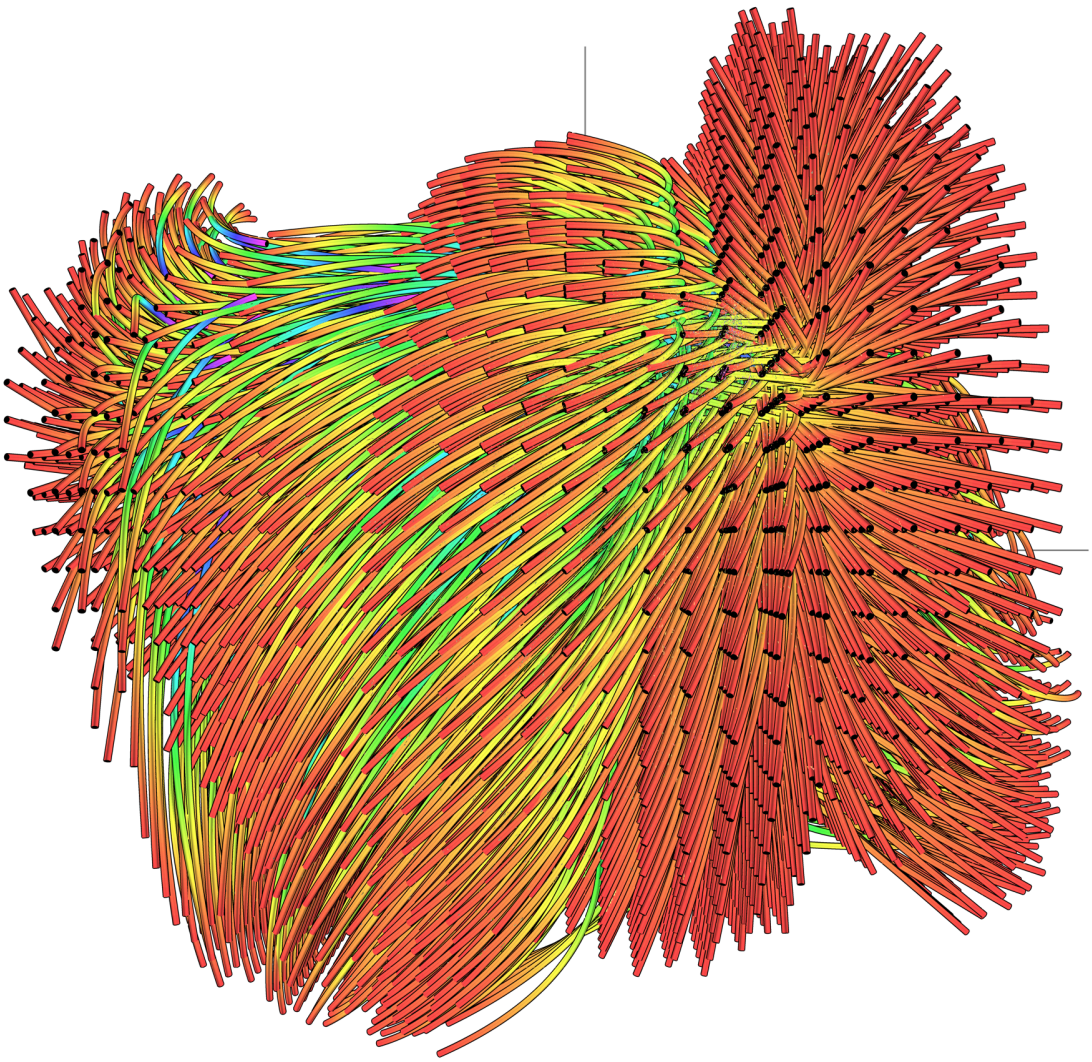

Figure 3: 'Pinhead': $Z' = Z^4 + C$, where $C_{xyzw} = 0.3, 0.5, 0.4, 0.2$.

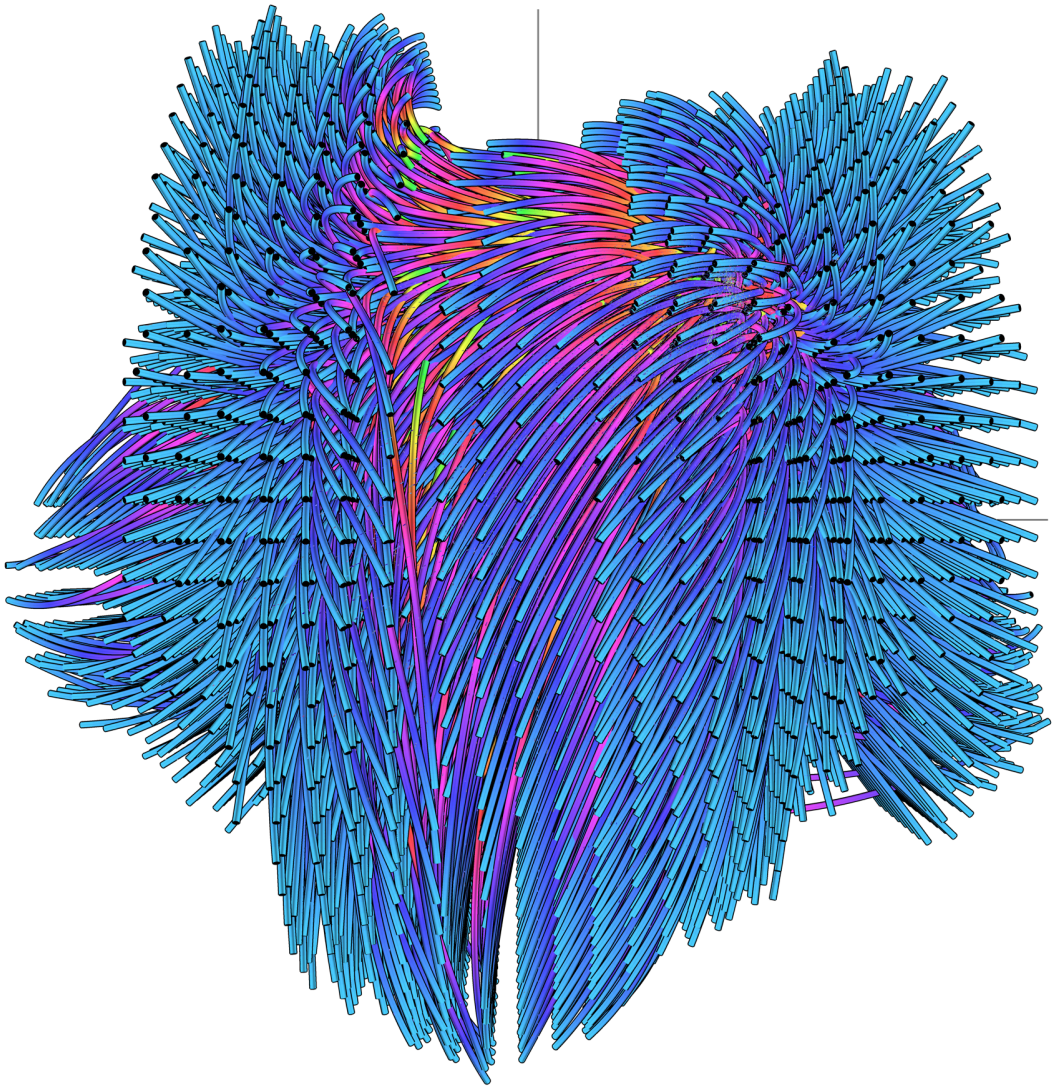

Figure 4: $Z' = Z^5 + C$, where $C_{xyzw} = 0.3, 0.5, 0.4, 0.2$.

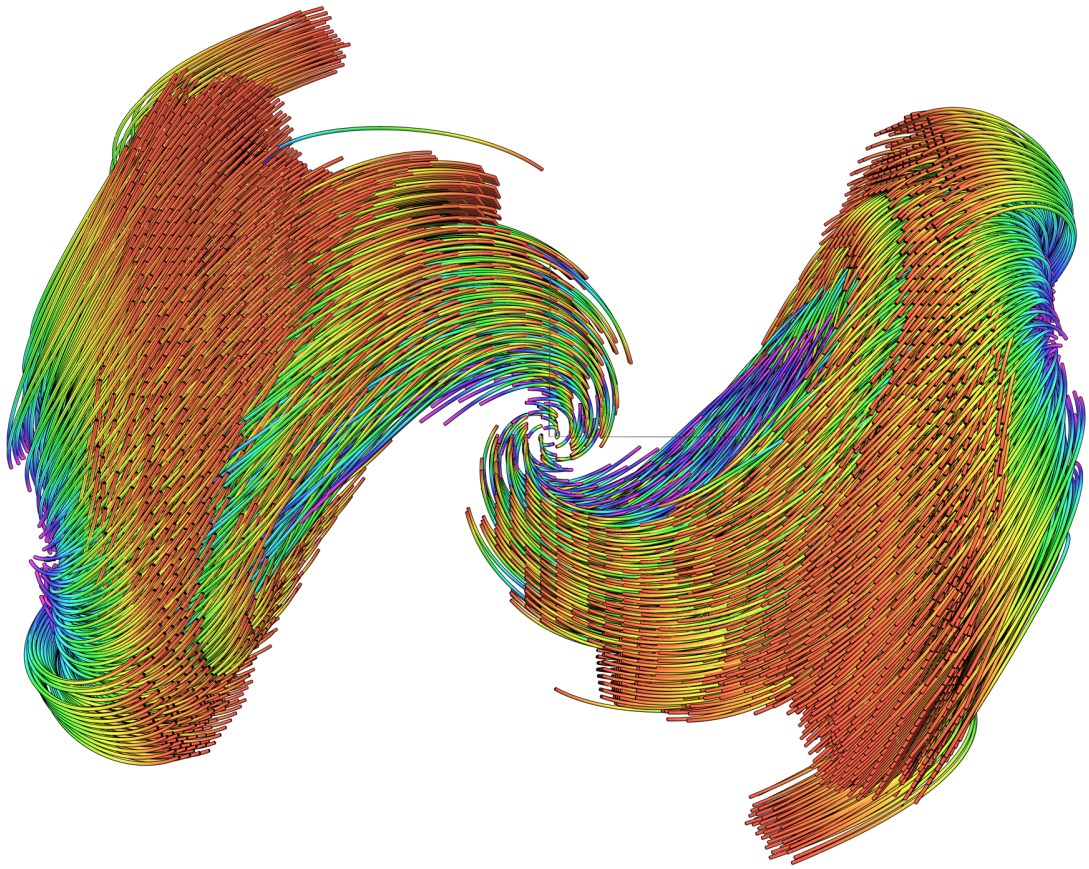

Figure 5: $Z' = \sin(Z) + C \cdot \sin(Z)$, where $C_{xyzw} = 0.3, 0.5, 0.4, 0.2$.

The C++ code for obtaining a point along a Bezier curve, given in [5], is:

```
vector_3 getBezierPoint(vector<vector_3> points, float t)
{
        int i = points.size() - 1;

        while (i > 0)
        {
                for (int k = 0; k < i; k++)
                {
                        points[k].x += t * (points[k + 1].x - points[k].x);
                        points[k].y += t * (points[k + 1].y - points[k].y);
                        points[k].z += t * (points[k + 1].z - points[k].z);
                }

                i--;
        }

        return points[0];
}
```

The C++ code for drawing a cylinder in OpenGL, given by JYK on gamedev.net, is:

```
static const float rad_to_deg = 180.0f/pi;

vector_3 line = pos[i][j + 1] - pos[i][j];

glPushMatrix();
glTranslatef(pos[i][j].x, pos[i][j].y, pos[i][j].z);

float line_len = line.length();
line.normalize();

float yaw = 0.0f;

if (fabsf(line.x) < 0.00001 && fabsf(line.z) < 0.00001)
        yaw = 0.0f;
else
        yaw = atan2f(line.x, line.z);

float pitch = -atan2f(line.y, sqrt(line.x*line.x + line.z*line.z));

glRotatef(yaw*rad_to_deg, 0.0f, 1.0f, 0.0f);
glRotatef(pitch*rad_to_deg, 1.0f, 0.0f, 0.0f);

gluCylinder(glu_obj, 0.005, 0.005, line_len, 20, 2);

glPopMatrix();
```

References

[1] Halayka S. *Some visually interesting non-standard quaternion fractal sets* Chaos, Solitons & Fractals Vol. 41, Issue 5

[2] http://paulbourke.net/fractals/trajectories/

[3] https://github.com/sjhalayka/bezier_fractal

[4] https://drive.google.com/uc?id=1G0xnkXQa0Rc3bv0Bxih4VYU0lj-RJI4G&export=download

[5] https://stackoverflow.com/questions/785097/how-do-i-implement-a-bezier-curve-in-c