Shape Modeling with Fractals

Tim McGraw and Donald Herring

Purdue University

Abstract. Fractal phenomena are a source of geometric detail that can be difficult to harness for general purpose shape modeling. We present a method for modeling surfaces by warping a fractal onto a given mesh. The warp is specified by the user as a coarse displacement field and interpolated by triharmonic radial basis functions. Efficient methods for rendering the warped fractal by ray tracing and isosurface extraction are presented. Examples are shown using several escape-time fractals and low-poly meshes.

1 Introduction

A key feature of fractal patterns is that they display self-similarity over all scales. The patterns can be generated by several techniques, e.g. escape-time formulas, strange attractors and iterated function systems. The fractal nature of some iterated function systems, such as the Sierpinski gasket and Koch snowflake, is intuitive from the description of their construction. The Koch snowflake is constructed by starting with an equilateral triangle. The inner third of each edge is removed and replaced with another equilateral triangle. This process is repeated indefinitely. At all scales the snowflake has a characteristic spiky star-like appearance.

However, in fractal images generated by escape-time fractals, it is less obvious how the recursive construction leads to the resulting image. The Mandelbrot set [1] and Julia sets are generated from simple recursive calculations in the complex plane but give rise to a rich array of patterns. Upon exploration some unexpected details may emerge. In general, escape-time fractals are defined as the set of points that diverge (or "escape") when iterated through a given formula. For the quaternion Julia set the recurrence formula is $z \mapsto z^2 + c$, where z and c are quaternions. In practice, a fixed distance threshold is used to approximate which points diverge. For many fractals it is possible to estimate the distance to the set from a given point. This permits accelerated ray tracing algorithms and extraction of isosurfaces of the distance estimate.

Recursive escape-time fractal systems, such as the Mandelbox and Mandelbulb (named for Benoit Mandelbrot [1]), can generate a dazzling array of structures and patterns. Open source software such as Fragmentarium [2] facilitates exploration of these systems. Aside from abstract computational art, it is unclear how to exploit these mathematical constructs for general computer graphics modeling. An ideal application of fractals to surface modeling requires the solution of a difficult inverse problem: finding the equations which give rise to a

desired shape. The Collage Theorem [3] states general conditions for the existence of a solution to this inverse problem and describes a general approach to its construction. In 2D this has led to fractal image compression algorithms, but general 3D solutions have been elusive.

Our approach to utilizing fractals in 3D surface modeling is to warp a selected volume of an escape-time fractal into another volume bounded by a mesh. In this work the warp is specified in terms of radial basis functions. The resulting shape can be visualized using ray tracing, or by rasterizing a triangle mesh generated by isosurface extraction. Our approach permits the user to specify correspondence between a set of mesh vertices and points in the domain of the fractal, much like a mesh is unwrapped for texture mapping. In our case the unwrapping happens in 3D rather than in the image plane.

2 Previous Work

In this section we review the formulations of some fractals which serve as a source of detail for our later experiments. We also describe the use of radial basis functions for surface fitting and warping.

2.1 3D Fractals

The search for a true 3D analogue of the Mandelbrot set has resulted in escape-time fractal systems with interesting detail at many scales. Strangely, the quaternion Julia sets which use an extension of complex numbers to 4D, do not have the rich variety of detail one might expect given the detail seen in the Mandelbrot set which is computed in the 2D complex plane. Some recent formulations from the generative art community rely instead on manipulation of spatial coordinates in R^3.

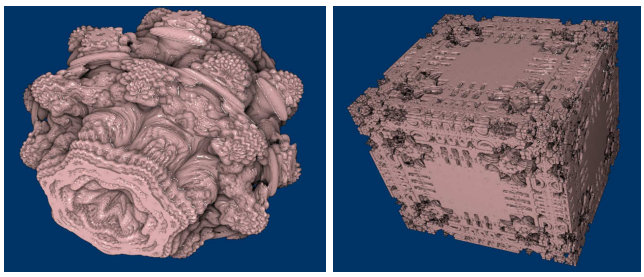

Fig. 1: Mandelbulb (*left*) and Mandelbox (*right*)

The Mandelbulb, Figure (1), is computed by the recurrence $z \mapsto z^n + c$ where $z \in R^3$, with

$$z^n = [x, y, z]^n = r^n[\sin(n\theta)\cos(n\phi), \sin(n\theta)\sin(n\phi), \cos(n\theta)]\,, \qquad (1)$$

where r, ϕ, θ are the spherical coordinates of x, y, z. It emerged from the generative art community at fractalforums.com. Much more on its history has been written by Daniel White [4].

Hart et al. [5] introduced the idea of determining bounds on the distance to a fractal surface to accelerate ray tracing. A distance estimate to the surface of the Mandelbulb can be computed by updating one additional variable during iteration: initialize $dr = 1$, and during iteration $dr \mapsto nr^{n-1}dr + 1$. After the last iteration the distance estimate is given by $DE = (1/2)\log(r)r/dr$. The variable dr is a derivative that gets updated during iteration, and the final distance estimation can be seen as an application of Newton's method.

The Mandelbox, Figure (1), discovered by Tom Lowe [6], is usually defined using pseudocode, and written in terms of "folds" as

$$v \mapsto s\,\text{ballFold}(r_0, f\,\text{boxFold}(v)) + c\,, \tag{2}$$

where $r_0 = 0.5, s = 2, f = 1$. The folds are defined as "conditional reflections." The boxFold(v) and ballFold(r_0, v) procedures are defined below in Algorithm (1).

```
1: function boxFold(v)
2:     for i = 1 to dim do
3:         if v_i > 1 then
4:             v_i = 2 − v_i
5:         else if v_i < −1 then
6:             v_i = −2 − v_i
7:         end if
8:     end for
9: end function
```

```
1: function ballFold(r_0, v)
2:     if ||v|| < r_0 then
3:         v = v/r_0^2
4:     else if ||v|| < 1 then
5:         v = v/||v||^2
6:     end if
7: end function
```

Alg. 1: Pseudocode for boxFold and ballFold as used in Mandelbox

Similar to the distance estimate described for the Mandelbulb, the distance to the Mandelbox (and many other fractals) can also be estimated by keeping track of a running derivative during iteration.

2.2 Radial Basis Functions

Radial basis functions (RBFs) have been popular for image warping and registration, but have also been applied to surface reconstruction from points [7], surface editing [8] and posing [9].

For shape modeling applications the surface is represented as the zero-valued isosurface of $s(\mathbf{x})$ given by

$$s(\mathbf{x}) = p(\mathbf{x}) + \sum_{i=1}^{n} \lambda_i \phi(|\mathbf{x} - \mathbf{x}_i|) \tag{3}$$

where $\mathbf{x}_i$ are points on the surface (often referred to as RBF centers), ϕ is the radial basis function, λ is a set of weights and $p(\mathbf{x})$ is a low degree polynomial. In our results we have used $\phi(r) = r$ and $\phi(r) = r^3$.

Given a set of RBF centers, $\mathbf{x}_i$, the weights and polynomial coefficients are found by solving the linear system

$$\begin{bmatrix} A & P \\ P^T & 0 \end{bmatrix} \begin{bmatrix} \lambda \\ c \end{bmatrix} = B \begin{bmatrix} \lambda \\ c \end{bmatrix} = \begin{bmatrix} f \\ 0 \end{bmatrix} \tag{4}$$

where

$$A_{ij} = \phi(|\mathbf{x}_i - \mathbf{x}_j|) \tag{5}$$

$$P_{ij} = p_j(\mathbf{x}_i)\,. \tag{6}$$

The low-order polynomial, p was chosen to be $p(\mathbf{x}) = c_1 + c_2x + c_3y + c_4z$, making row i of P equal to $[1, x_1, y_i, z_i]$, and $c = [c_1, c_2, c_3, c_4]^T$. To specify that the zero valued isosurface $s(\mathbf{x}) = 0$ is the surface of interest we set $f = 0$, and solve Equation (4) for λ and c. There is a trivial solution to this problem where $f = 0$ everywhere. To avoid this, additional RBF centers are added by finding additional points inside and outside the surface and setting the corresponding f values to be negative and positive values respectively. These points are found by searching along the normal direction from each vertex, as illustrated in Figure (2). During this search it must be kept in mind that simply projecting outward by a fixed distance, d, doesn't necessarily give a point outside the surface, since the surface may be concave. By setting $f(x_i, y_i, z_i) = \pm d_i$ we can solve for an approximate signed distance function.

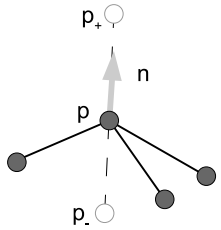

Fig. 2: Off-surface points p_- and p_+ are placed along the line through p in the direction of the vertex normal, n

The problem of finding an RBF warp that maps mesh M_0 to M_1 is nearly identical, except that s, λ, and p are vector-valued. Note that in our work M_1 is simply a copy of M_0 with edited vertex locations, so we do not need to solve a mesh correspondence problem, as shown in Figure (3). Finding the warp requires solution of

$$B \begin{bmatrix} \lambda_x & \lambda_y & \lambda_z \\ c_x & c_y & c_z \end{bmatrix} = \begin{bmatrix} f_x & f_y & f_z \\ 0 & 0 & 0 \end{bmatrix} \tag{7}$$

where $\mathbf{x}_i$ are vertices of M_0, and $\mathbf{f} = (f_x, f_y, f_z)$ are vertices of M_1 and B is the same as in Equation (4). The warp from M_0 to M_1 is given by

$$W(\mathbf{x}) = \mathbf{p}(\mathbf{x}) + \sum_{i=1}^{n} \lambda_\mathbf{i} \phi(|\mathbf{x} - \mathbf{x_i}|) \tag{8}$$

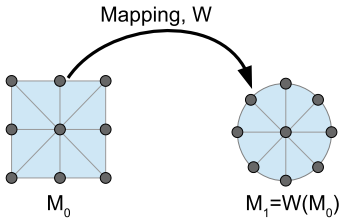

Fig. 3: Mapping, W, from M_0 to M_1

RBFs do give rise to some computational challenges. Finding the n coefficients requires solution of a dense $(n+4) \times (n+4)$ linear system. Matters would be simplified if we could use RBFs with compact support since the matrix B would then be sparse, but compactly-supported RBFs would not allow us to extrapolate the warp field away from the surface of M_0, which is critical to our modeling application. Once the coefficients are found, evaluating the RBF can be expensive. The use of fast multipole methods (FMMs) can overcome this problem [10]. FMMs work by grouping centers in a hierarchical structure, like a kd-tree, and treating each cluster as a single point if the error in doing so falls below a given threshold.

3 Methods

The high-level pipeline of our application is described below.

1. Select fractal equation and parameters. The user may interactively explore the fractal to find a region of useful pattern.

2. Select a mesh, M_0 to apply the fractal to.

3. Define the warp by deforming the vertices of M_0 to obtain a mesh M_1 in the domain of the fractal and defining off-surface points. This step is analogous to unwrapping a mesh for texture mapping. Compute the RBF coefficients for the warp.

4. Obtain the result by ray tracing or isosurface extraction. For the points generated by marching along a ray, or sampling a grid (isosurface extraction), transform each point into the warped domain, and evaluate the distance estimation formula for the escape-time fractal. That distance is stored as the embedding function value for the isosurface, or is used to select an adaptive step size along the ray.

3.1 Ray Tracing Warped Fractals

Distance estimation for fractals can accelerate ray tracing since it gives a lower bound on the distance to the surface, and we can safely step by this distance along the ray without intersecting the surface.

Ray tracing the fractal warped onto M_0 can be seen as ray tracing with curved rays in the space of M_1, as shown in Figure (4). For each point along the linear ray in M_0, that point is mapped into M_1, and the distance to the fractal is estimated. However a closed-form expression for the warped ray is expensive to evaluate. To overcome this problem we use a Taylor series expansion of the equation of the warped ray in M_1 to estimate how far we can march.

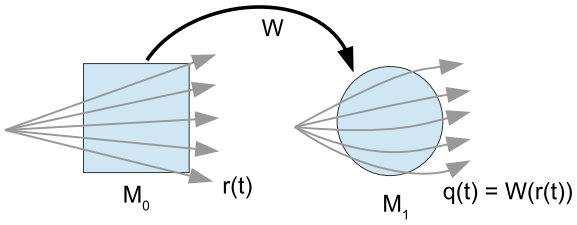

Fig. 4: Rays, $r(t)$, cast from the eye into the scene containing M_0 are warped by W into curves which intersect M_1

Let the ray through the eye point $\mathbf{p}$ be given by $\mathbf{r}(t) = \mathbf{p} + t\mathbf{d}$, and let $\mathbf{q}(t) = W(\mathbf{r}(t))$. Then the 3rd-order Taylor series expansion of $\mathbf{q}$ about $t = t_0$ is given by

$$\mathbf{q}(t) \approx \mathbf{q}(t_0) + \mathbf{q}'(t_0)(t - t_0) + \frac{\mathbf{q}''(t_0)}{2!}(t - t_0)^2 + \frac{\mathbf{q}^{(3)}(t_0)}{3!}(t - t_0)^3 \tag{9}$$

Having computed the warped position $\mathbf{q}(t_0)$ along the ray we can compute the derivatives $\mathbf{q}', \mathbf{q}'', \mathbf{q}^{(3)}, \mathbf{q}^{(4)}$ efficiently by reusing the intermediate results $\delta_i = x_i - x$ and $r_i = ||\delta_i||$,

$$\begin{aligned}
\mathbf{q}' &= (\mathbf{I} + \mathbf{P})\mathbf{d} + 3\sum_{i=1}^{n}(\mathbf{d} \cdot \delta_i) r_i \lambda_i \\
\mathbf{q}'' &= 3\sum_{i=1}^{n}\left(\frac{(\mathbf{d} \cdot \delta_i)^2}{r_i} + (\mathbf{d} \cdot \mathbf{d}) r_i\right)\lambda_i \\
\mathbf{q}^{(3)} &= \sum_{i=1}^{n}\left(-3\frac{(\mathbf{d} \cdot \delta_i)^3}{r_i^3} + 9\frac{(\mathbf{d} \cdot \mathbf{d})(\mathbf{d} \cdot \delta_i)}{r_i}\right)\lambda_i \\
\mathbf{q}^{(4)} &= \sum_{i=1}^{n}\left(9\frac{(\mathbf{d} \cdot \delta_i)^4}{r_i^5} - 18\frac{(\mathbf{d} \cdot \mathbf{d})(\mathbf{d} \cdot \delta_i)^2}{r_i^3} + 9\frac{(\mathbf{d} \cdot \mathbf{d})^2}{r_i}\right)\lambda_i.
\end{aligned} \tag{10}$$

There is no closed-form expression for the distance along the parametric cubic curve given by Equation (9), so we estimate it using the Vincent-Forsey rule described by Floater and Rasmussen [11]

$$L(\mathbf{q}_{[t0,t1]}) = \frac{4}{3}(||\mathbf{q}(t_m) - \mathbf{q}(t_0)|| + ||\mathbf{q}(t_1) - \mathbf{q}(t_m)||) - \frac{1}{3}||\mathbf{q}(t_1) - \mathbf{q}(t_0)|| \quad (11)$$

where $t_m = (t_0 + t_1)/2$.

The magnitude of the error of the 3rd-order Taylor series approximation can be bounded by

$$||E_3(t)|| \leq M\frac{(t - t_0)^4}{4!} \quad (12)$$

where $||\mathbf{q}^{(4)}(t)|| \leq M$ for $t_0 \leq t \leq t_1$.

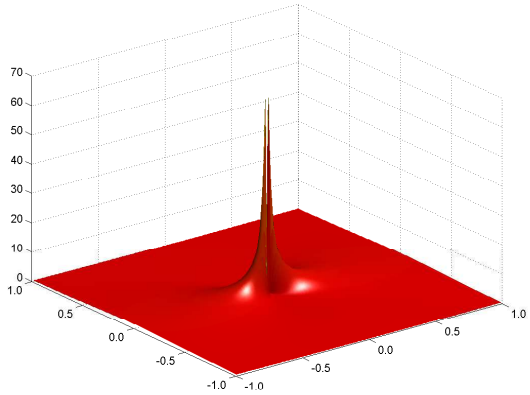

Fig. 5: Plot of $||\mathbf{q}^{(4)}(t)||$ for rays in a plane passing through an RBF center. Note that the value is nearly zero, except in a small area near the center

A plot of $||\mathbf{q}^{(4)}(t)||$ in the neighborhood of an RBF center is shown in Figure (5). From this plot it can be seen that M, and therefore $||E_3(t)||$, are small except for rays that pass very close to an RBF center. By supersampling 4 or more rays per pixel we reduce the impact of the high-error rays. As we search along the warped ray we simply let M equal the maximum value of $||\mathbf{q}^{(4)}||$ we have encountered along the ray.

Pseudocode of the warped fractal ray tracing process is given below. The routine returns the parameter value t of the intersection with the warped fractal.

1. For each ray $\mathbf{r}(t) = \mathbf{p} + t\mathbf{d}$ compute the approximate warped ray $\mathbf{q}(t)$ using Equations (9) and (10)
2. Let $t_0 = 0$, $t_1 = 1$, $n = 0$
3. Compute distance L_{target} to fractal from $\mathbf{q}(t_0)$ using fractal distance estimation.
4. If $L_{target} < \epsilon$ then return t_0

5. Perform binary search along $\mathbf{q}(t)$ between t_0 and t_1 for the point $\mathbf{q}(t_{new})$ such that $L(\mathbf{q}_{[t0,t_{new}]}) = L_{target}$, using Equation (11).
6. $n = n + 1$
7. Let $t_0 = t_{new}$
8. If $n < n_{max}$ then goto 3

During the binary search in step 5, the approximation error of the Taylor series expansion of $\mathbf{q}(t)$ is taken into account by incorporating the error estimate, E in the midpoint evaluation step:

1. $t_{mid} = (t_0 + t_1)/2$
2. $M = \max(M, q^{(4)}(t_{mid}))$
3. $E = E_3(t_{mid})$
4. $L_{mid} = L(\mathbf{q}_{[t0,t_{mid}]})$
5. if $L_{mid} + E < L_{target}$ then $t_0 = t_{mid}$
6. else $t_1 = t_{mid}$
7. goto 1

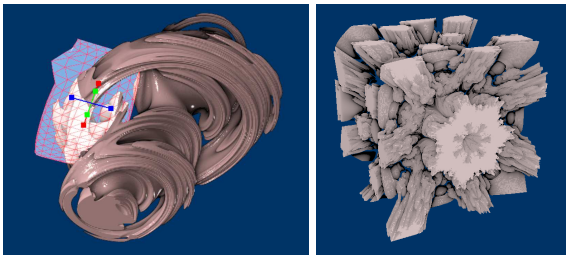

Fig. 6: Example of mesh M_1 being positioned relative to Julia set to define a warp (*left*) and ray-traced warped Mandelbulb (*right*)

An example ray-traced warped fractal is shown in Figure (6). In our final application we use the ray-traced image as a preview of the final results for the user, and the extracted isosurface is saved to a file for final rendering.

3.2 Isosurface Extraction of Warped Fractals

For isosurface extraction we use a variant of marching cubes proposed by Lewiner et al. [12] that guarantees topological correctness. However, the mesh generation process poses two problems that we need to address. First, extracting an isosurface with detail at small scales can lead to noisy looking mesh. Second, extracting a high resolution mesh of a warped fractal can be time consuming due to the large number of points at which the warp and fractal distance estimate will be computed.

Due to the occurrence of fractal detail at many scales, we can end up with much geometric detail within a single cube of the marching cubes array. In this

case the underlying surface cannot be accurately reconstructed using only the 8 distance values at the corners of the cube. Our approach to this problem is simple: we offset the surface in the outward normal direction by a small amount, ϵ. This has the effect of smoothing small scale fractal detail that cannot be represented by the triangulated mesh. In theory, we can guarantee that $\epsilon = s/2$, where s is the marching cubes step size, will be sufficient. In practice, we make ϵ a user-supplied parameter.

The second problem we only solve in the case where the extracted surface consists of a single connected component. Rather than exhaustively evaluating the warped fractal distance function for each vertex in the marching cubes array, we instead cast rays through the volume to find a single surface intersection point. From this seed point we then march along adjacent cubes that intersect the surface. This permits much empty space to be skipped, greatly reducing the number of warp and distance function evaluations.

4 Results

Our fractal modeling system was implemented in C++ and OpenGL on a Dell Optiplex workstation with 3.4 GHz Intel Core i7-3770 CPU and 8GB RAM, Nvidia GeForce GTX 750 Ti with 640 shader cores and 2 GB GDDR5 dedicated video RAM. The results are shown with the original coarse mesh M_0 to the left.

Fig. 7: Gallery of shape modeling using fractals

The user of our system positioned mesh vertex locations relative to the selected fractal and previewed results using our accelerated ray tracing procedure. When satisfied with the results a final mesh was generated using our modified marching cubes algorithm. Results are shown for Mandelbulb-mushroom with

Julia set-stem, Mandelbox-hand, Mandelbulb-head, Mandelbulb-bunny, and Mandelbox - teapot. Our accelerated ray tracing algorithm was implemented in the OpenGL shading language (glsl), and permitted previewing results at 5-10 fps while the unaccelerated ray tracing procedure caused the video driver to timeout and reset.

5 Conclusions

In this paper we have described a system which can be used to harness the diverse patterns found in escape-time fractals for general purpose modeling. We described a warping technique using radial basis functions for mapping a user-specified region of the fractal onto an arbitrary mesh. We presented approaches for efficiently visualizing the results using ray tracing and isosurface extraction. Results show that this technique has promise for shape modeling purposes. Future work will involve exploring automatic techniques for finding a mapping between the fractal and mesh based on symmetry and other features.

References

1. Mandelbrot, B.B.: The fractal geometry of nature. Volume 173. Macmillan (1983)
2. Christensen, M.: Fragmentarium. http://syntopia.github.io/Fragmentarium (2013)
3. Barnsley, M.F., Ervin, V., Hardin, D., Lancaster, J.: Solution of an inverse problem for fractals and other sets. Proceedings of the National Academy of Sciences of the United States of America **83** (1986) 1975–1977
4. White, D.: The unravelling of the real 3d mandelbulb. http://www.skytopia.com/project/fractal/mandelbulb.html (2009)
5. Hart, J.C., Sandin, D.J., Kauffman, L.H.: Ray tracing deterministic 3-d fractals. ACM SIGGRAPH Computer Graphics **23** (1989) 289–296
6. Lowe, T.: What is a mandelbox? https://sites.google.com/site/mandelbox/what-is-a-mandelbox (2010)
7. Carr, J.C., Beatson, R.K., Cherrie, J.B., Mitchell, T.J., Fright, W.R., McCallum, B.C., Evans, T.R.: Reconstruction and representation of 3d objects with radial basis functions. In: Proceedings of the 28th annual conference on Computer Graphics and Interactive Techniques, ACM (2001) 67–76
8. Sieger, D., Menzel, S., Botsch, M.: High quality mesh morphing using triharmonic radial basis functions. In: Proceedings of the 21st International Meshing Roundtable. Springer (2013) 1–15
9. Botsch, M., Kobbelt, L.: Real-time shape editing using radial basis functions. Computer Graphics Forum **24** (2005) 611–621
10. Beatson, R., Powell, M., Tan, A.: Fast evaluation of polyharmonic splines in three dimensions. IMA Journal of Numerical Analysis **27** (2007) 427–450
11. Floater, M.S., Rasmussen, A.F.: Point-based methods for estimating the length of a parametric curve. Journal of computational and applied mathematics **196** (2006) 512–522
12. Lewiner, T., Lopes, H., Vieira, A.W., Tavares, G.: Efficient implementation of marching cubes' cases with topological guarantees. Journal of Graphics Tools **8** (2003) 1–15