Distance estimation method for drawing Mandelbrot and Julia sets

Dr. Lindsay Robert Wilson
http://imajeenyus.com

20/11/12

Credit & sources

The distance estimation method (DEM) is one of several techniques commonly used for visualising Mandelbrot and Julia sets. It is slightly more complicated than the simplistic "in or out" colouring scheme, but gives much better results, especially in regions which are "filamentous" - those that have a lot of structure, but contain relatively little of the set. However, I couldn't find a really good explanation of how the DEM works, so I've tried to provide a decent description and examples here. I won't go into detail of the different colour gradient mappings I've used - there's many different ways of achieving a "nice" result, and it's best just to play around with them.

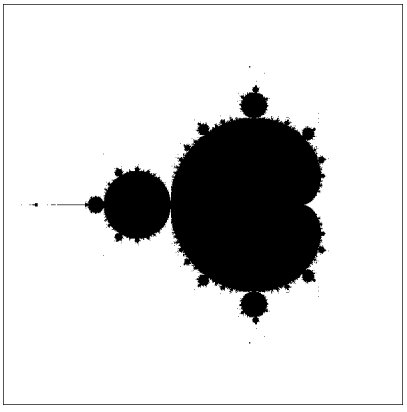
The main inspiration was Iñigo Quilez' page at http://www.iquilezles.org/www/articles/distancefractals/distancefractals.htm. Other good references are http://mrob.com/pub/muency/distanceestimator.html and http://en.wikibooks.org/wiki/Fractals/Iterations_in_the_complex_plane/Julia_set. For a good overview of the complex quadratic polynomial and its derivatives, see http://en.wikipedia.org/wiki/Complex_quadratic_polynomial.

Mandelbrot set recap

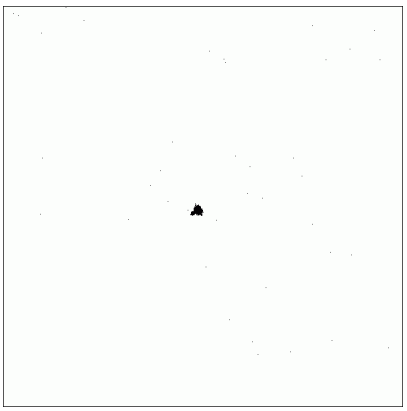
The Mandelbrot set is the set of all points c in the complex plane for which the iterative process $z_{n+1} = z_n^2 + c, z_0 = 0 + 0i$ is bounded. Pick a point c and calculate a series of z_{n+1}. If the modulus of z_{n+1} heads off to infinity, then c is outside the set, otherwise c is inside. In practice, this is determined by comparing $|z_{n+1}|$ against a limit radius - if the modulus is greater than the limit radius, the original c point is assumed to be outside. A cap on the maximum number of iterations is also used, to save on time.

"In or out" colouring scheme

The simplest method of displaying the set is to colour a pixel black if it is within the set and white if it is outside. This produces the following familiar result for the whole set.



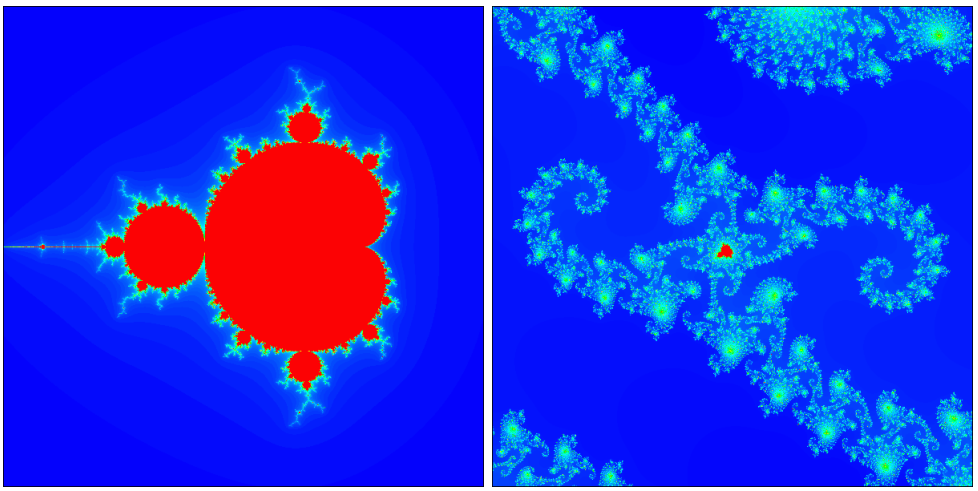
This looks pretty reasonable, but look at what happens if we try and look at a region in greater detail. The next image is centered at $(-0.74364085, 0.13182733)$ and has a width and height of 0.00012.



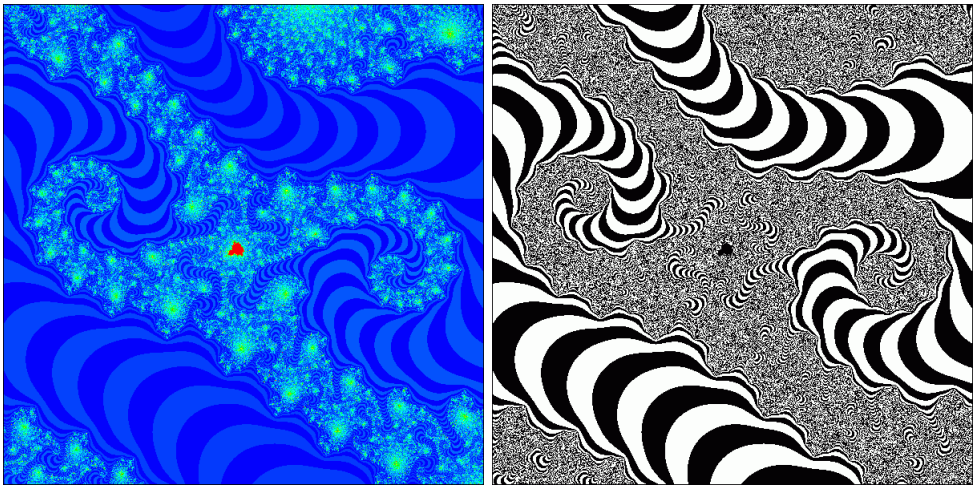
Pretty poor. There is, in fact, a lot of detail around the central miniature Mandelbrot set (see colour pictures later), but the details are smaller than a single pixel and appear invisible. Fortunately, there is a better way of visualising the set, which doesn't involve too much extra calculation.

"Escape time" colouring scheme

During the iteration process performed on each point, we calculate the number of iterations required for $|z_{n+1}|$ to exceed the limit radius. This is called the escape time, or escape count. Points which are farther from the set will have a smaller escape time, because they require fewer iterations to reach the limit radius. Points inside the set will have the maximum allowed escape time (remember that the maximum number of iterations is limited). If we colour each point according to its escape time, we can obtain a much better representation. The two images below are of the whole set and the same zoomed region as before. Note that we are actually showing points which are outside the set.



While this is clearly a much better representation, it still isn't as good as what can be achieved using the distance estimation method. It clearly shows the existence of connecting filaments which were not visible in the "in or out" colouring scheme. Because the escape time is a discrete integer (number of iterations), bands are visible in the colour image. These are shown enhanced in the images below.

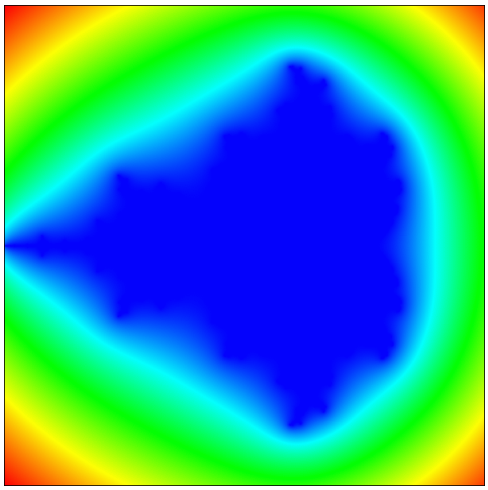


Distance estimation method

For each point outside the set, it is possible to calculate an estimate of the shortest distance between it and the set. There is a function called the Hubbard-Douady potential which gives the "potential" of every point outside the set. (Imagine a metal plate in the shape of the set with a voltage applied to it. The Hubbary-Douady potential would give the voltage (electric potential) at each point outside the set.) The potential function at a point c is given by

$$G(c) = \lim_{n\to\infty} \frac{1}{2^n} \ln |z_n| \tag{1}$$

This is plotted below for the entire set, with blue=small $G(c)$ and red=large $G(c)$.



The distance estimate is given by (similar to the expression for the distance to an isoline of a 2D function)

$$d = \frac{G(c)}{|G'(c)|} \tag{2}$$

The derivative of $G(c)$ with respect to c is

$$G'(c) = \lim_{n\to\infty} \frac{1}{2^n} \frac{|z'_n|}{|z_n|} \tag{3}$$

Therefore,

$$d = \lim_{n\to\infty} \frac{|z_n| \ln |z_n|}{|z'_n|} \tag{4}$$

How do we find z'_n? Note that this derivative is with respect to c. If we differentiate the complex quadratic equation

$$z_{n+1} = z_n^2 + c \tag{5}$$

with respect to c, we obtain

$$z'_{n+1} \quad = \quad 2z_n z'_n + 1 \tag{6}$$

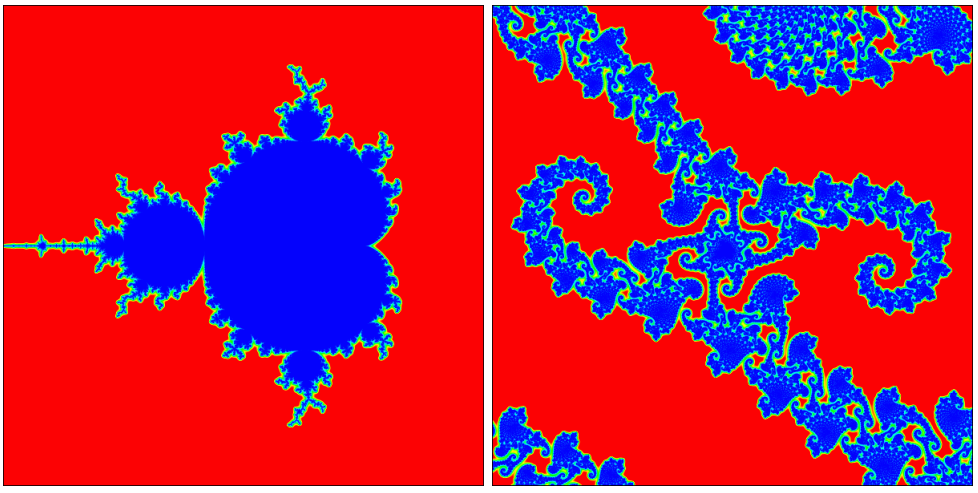
This can be calculated during the iteration process, with an initial value of $z'_0 = 1$. Here is a pseudocode example of a function which returns the distance from a point (c_x, c_y) to the set, calculated using (4).

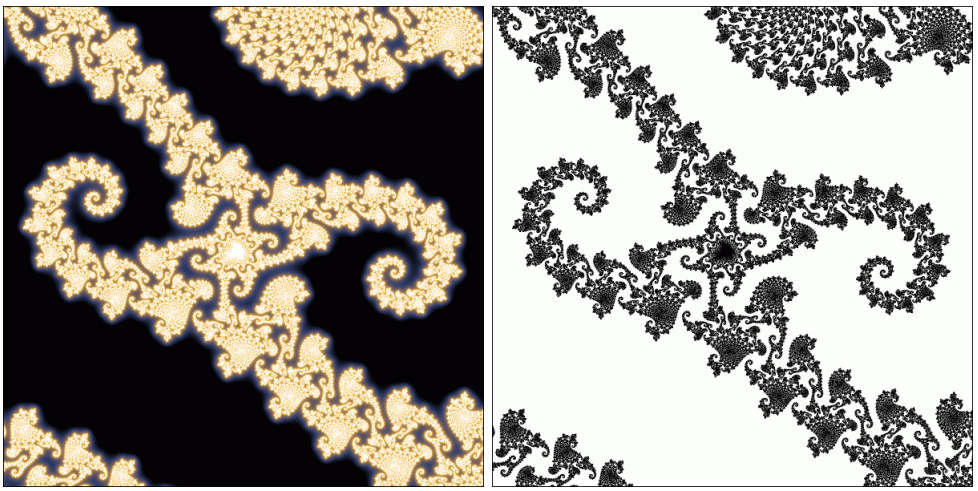
```
function set_distance(cx,cy) as float

    dim c,z,z_new,dz,dz_new as complex
    ' init variables
    c=cx+cy*i
    z=0+0*i
    dz=1+0*i
    cnt=1 ' initialise iteration counter
    do
        z_new=z*z+c ' iterate the quadratic equation
        dz_new=2*z*dz+1 ' iterate the derivative
        z=z_new ' roll values
        dz=dz_new ' roll values
        if modulus(z)>limit_radius then exit do ' if we exceed the limit then stop iterating
        cnt=cnt+1 ' increment counter
    loop until cnt>max_iterations
    return modulus(z)*log(modulus(z))/modulus(dz) ' return the distance

end function
```

Here are some examples of using the DEM to draw the set, with different colour gradient maps.





It gives a much clearer representation of the set. Note that most of the points in the zoomed region are actually outside the set, but are made visible by use of the distance estimation function.

Combinations of each of the above display techniques can also be used for greater visual effect, for example by using a different technique for each of the red, green, blue channels and then combining them. The potential function itself could also be used to display the set. There are many additional display techniques available, but these are probably the simplest to implement.

Julia sets

The distance estimation method can also be applied to Julia sets, with a slight modification. The Julia set at a point c is the set of all points z_0 for which the iterative process $z_{n+1} = z_n^2 + c$ remains bounded. (By contrast, the Mandelbrot set was the set of all points c, with $z_0 = 0 + 0i$ for all points.) According to `http://mrob.com/pub/muency/distanceestimator.html`, the Hubbard-Douady potential takes the same form - this is understandable, since the potential depends only on the value of z_n as $n \to \infty$. However, it is now a function of z_0, not c, and is given by

$$G(z_0) = \lim_{n\to\infty} \frac{1}{2^n} \ln |z_n| \tag{7}$$

The distance estimate is now

$$d = \frac{G(z_0)}{|G'(z_0)|} \tag{8}$$

The derivative of $G(z_0)$ with respect to z is

$$G'(z_0) = \lim_{n\to\infty} \frac{1}{2^n} \frac{|z_n'|}{|z_n|} \tag{9}$$

Therefore, we again have

$$d = \lim_{n\to\infty} \frac{|z_n| \ln |z_n|}{|z_n'|} \tag{10}$$

However, z_n' is now the derivative with respect to z_0 and is given by

$$z_{n+1}' = 2 z_n z_n' \tag{11}$$

with an initial value of $z_0' = 1$. This is very similar to (6), with the absence of the +1 term. Here is another pseudocode example for a function which returns the distance from a point (z_{0x}, z_{0y}) to the Julia set. (c_x, c_y) is also specified. Note how similar this is to the code for the Mandelbrot set - the only differences are the requirement to specify (z_{0x}, z_{0y}) and the slightly different derivative iteration equation.

```
function set_distance(cx,cy,z0x,z0y) as float
dim c,z,z_new,dz,dz_new as complex
    ' init variables
    c=cx+cy*i
    z=z0x+z0y*i
    dz=1+0*i
    cnt=1 ' initialise iteration counter
    do
        z_new=z*z+c ' iterate the quadratic equation
        dz_new=2*z*dz ' iterate the derivative
        z=z_new ' roll values
        dz=dz_new ' roll values
        if modulus(z)>limit_radius then exit do ' if we exceed the limit then stop iterating
        cnt=cnt+1 ' increment counter
    loop until cnt>max_iterations
    return modulus(z)*log(modulus(z))/modulus(dz) ' return the distance
end function
```

Here are a few plots of the Julia set using the distance estimation method. The first image is for $c = -0.8 + 0.16i$. The second image is a zoom of the upper part of the first. The third image is a zoom of the set for $c = -1.2 + 0.156i$.