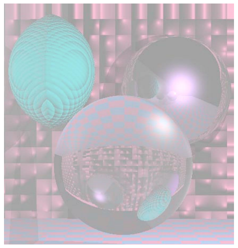


CS5310
Graduate Computer Graphics

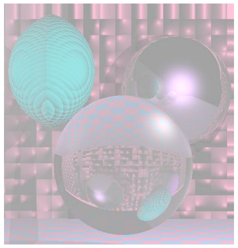
Prof. Harriet Fell

Spring 2011

Lecture 9 – March 23, 2011

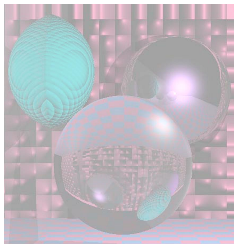
Today’s Topics

- Morphing
- Fractals



Morphing History

- *Morphing* is turning one image into another through a seamless transition.
- Early films used cross-fading picture of one actor or object to another.
- In 1985, "Cry" by Godley and Crème, parts of an image fade gradually to make a smother transition.
- Early-1990s computer techniques distorted one image as it faded into another.
  - Mark corresponding points and vectors on the "before" and "after" images used in the morph.
  - E.g. key points on the faces, such as the contour of the nose or location of an eye
  - Michael Jackson's "Black or White" (1991)
    - » http://en.wikipedia.org/wiki/Morphing



Morphing History

- 1992 Gryphon Software's “Morph” became available for Apple Macintosh.
- For high-end use, “Elastic Reality” (based on Morph Plus) became the de facto system of choice for films and earned two Academy Awards in 1996 for Scientific and Technical Achievement.
- Today many programs can automatically morph images that correspond closely enough with relatively little instruction from the user.
- Now morphing is used to do cross-fading.

Harriet George Harriet…

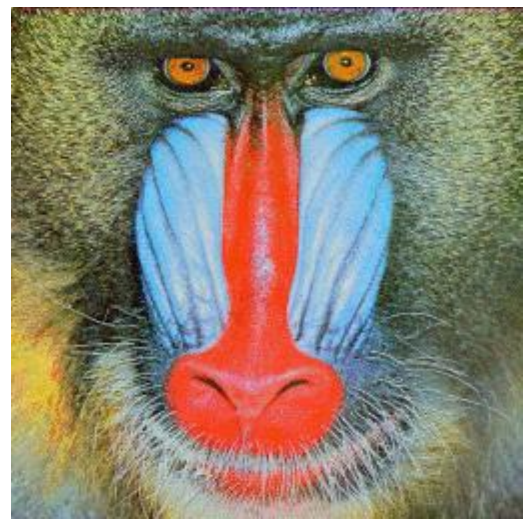
Feature Based Image Metamorphosis

Thaddeus Beier and Shawn Neely 1992

- The morph process consists
  - warping two images so that they have the same "shape"
  - cross dissolving the resulting images
- cross-dissolving is simple
- warping an image is hard

Harriet & Mandrill

Harriet 276x293



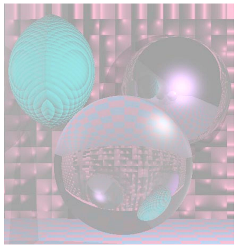
Mandrill 256x256

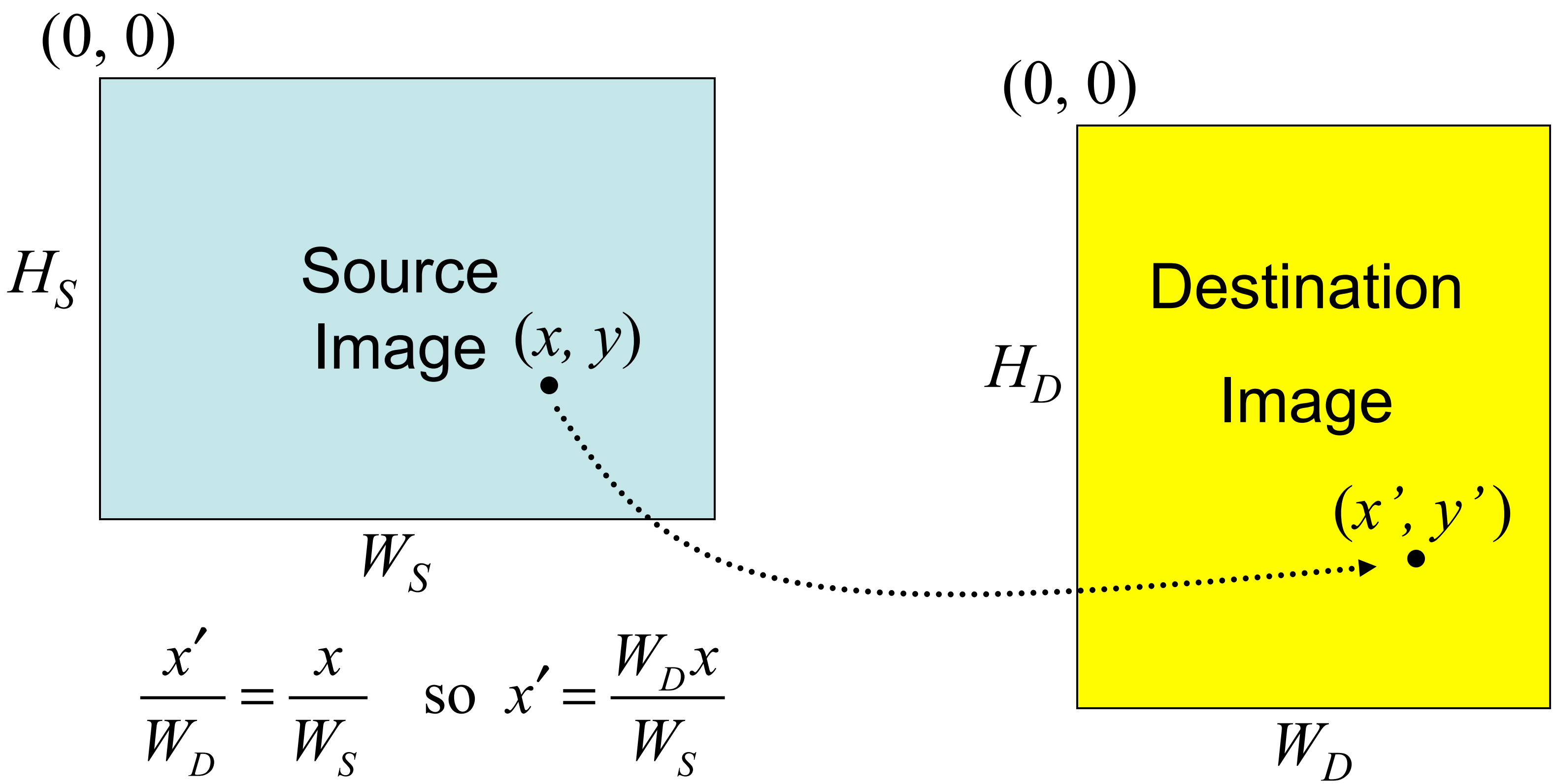
Warping an Image

There are two ways to warp an image:

- *forward mapping* - scan through source image pixel by pixel, and copy them to the appropriate place in the destination image.
  - some pixels in the destination might not get painted, and would have to be interpolated.
- *reverse mapping -* go through the destination image pixel by pixel, and sample the correct pixel(s) from the source image.
  - every pixel in the destination image gets set to something appropriate.

Forward Mapping

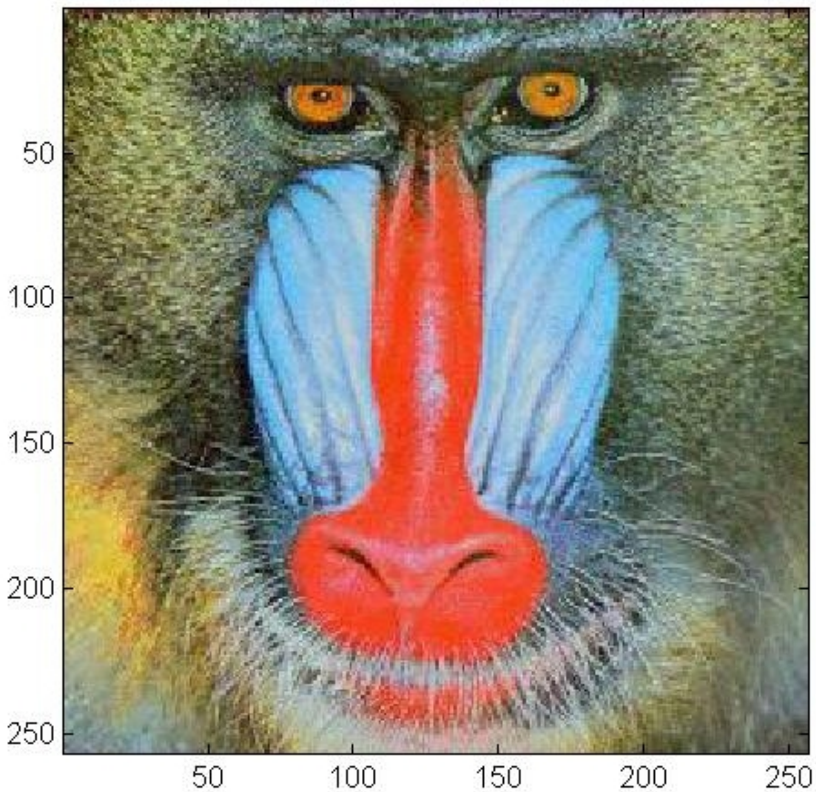


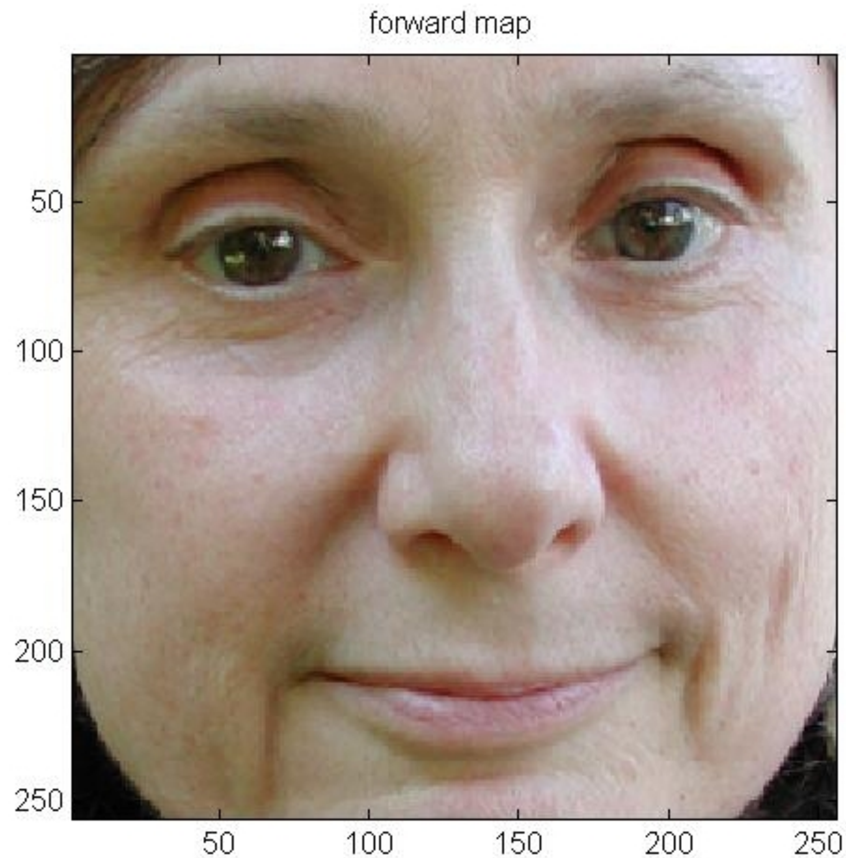


$$\frac{x'}{W_D} = \frac{x}{W_S} \quad \text{so} \quad x' = \frac{W_D x}{W_S}$$

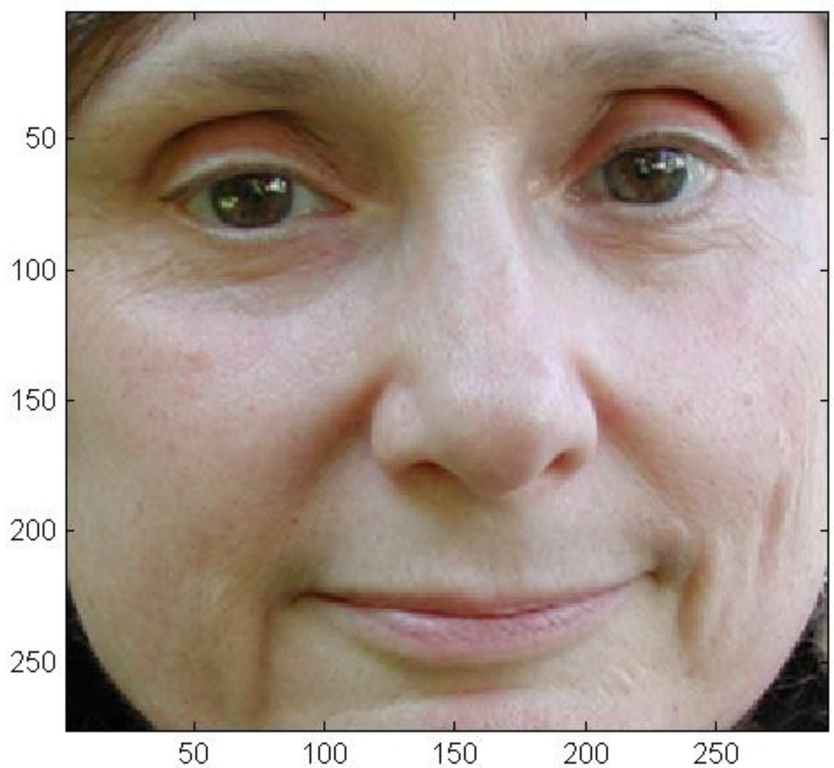
$$\frac{y'}{H_D} = \frac{y}{H_S} \quad \text{so} \quad y' = \frac{H_D y}{H_S}$$

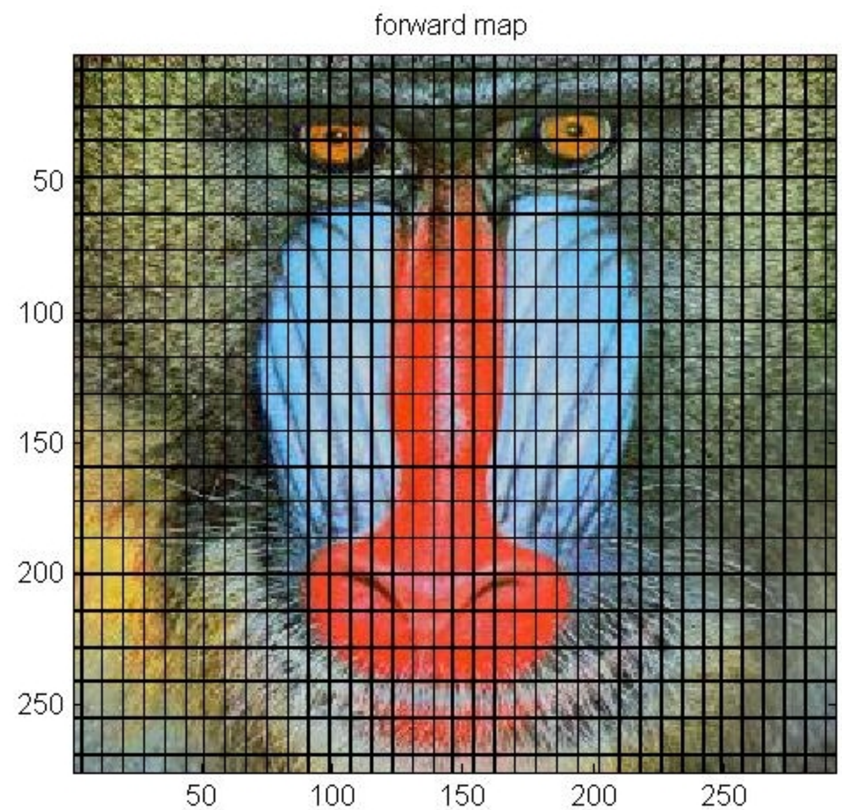
Forward Mapping
Harriet → Mandrill



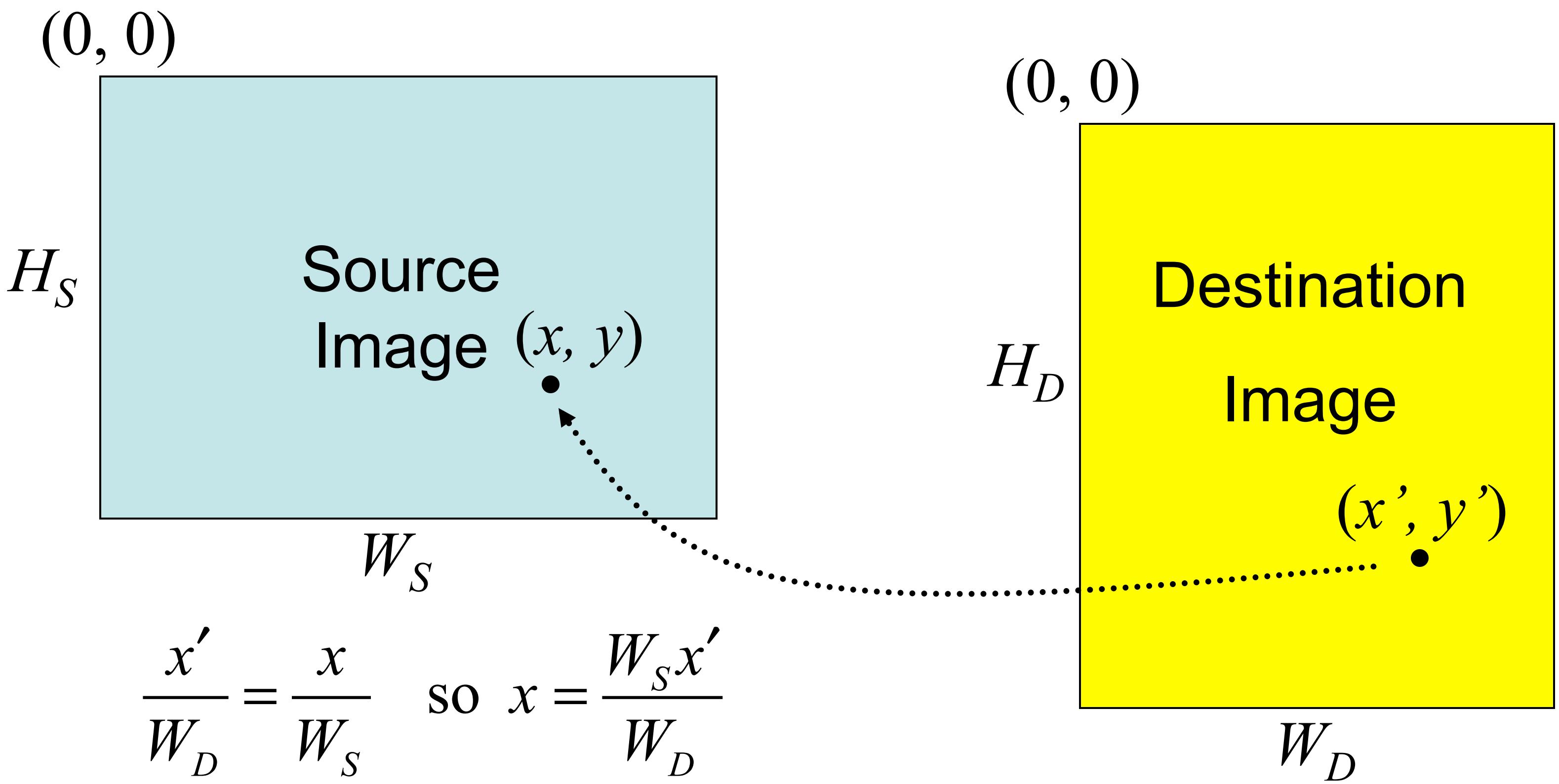


Forward Mapping
Mandrill → Harriet





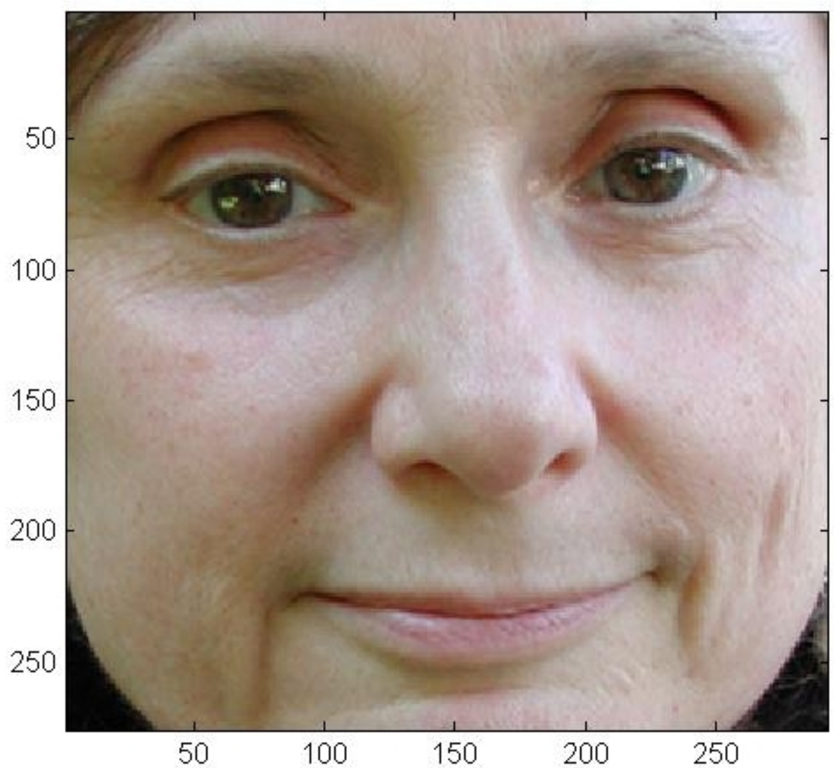
Inverse Mapping

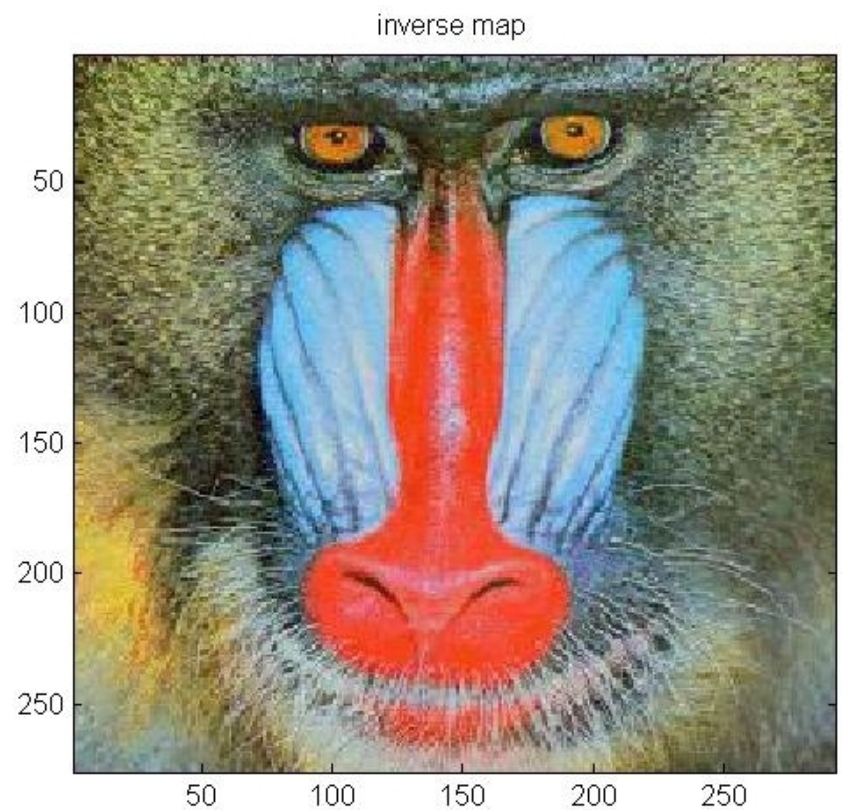


$$\frac{x'}{W_D} = \frac{x}{W_S} \quad \text{so} \quad x = \frac{W_S x'}{W_D}$$

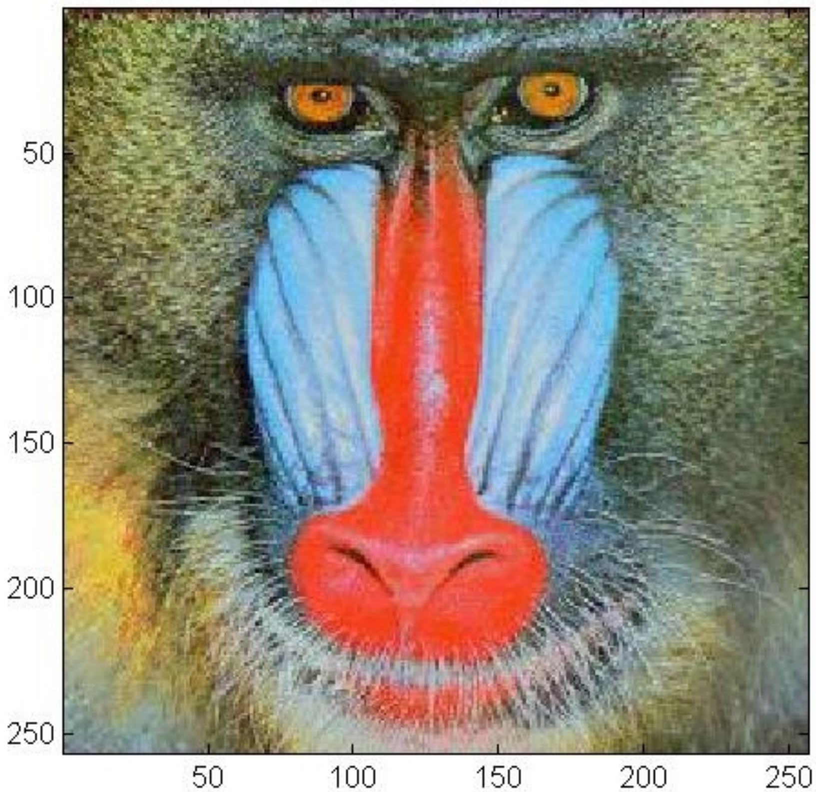
$$\frac{y'}{H_D} = \frac{y}{H_S} \quad \text{so} \quad y = \frac{H_S y'}{H_D}$$

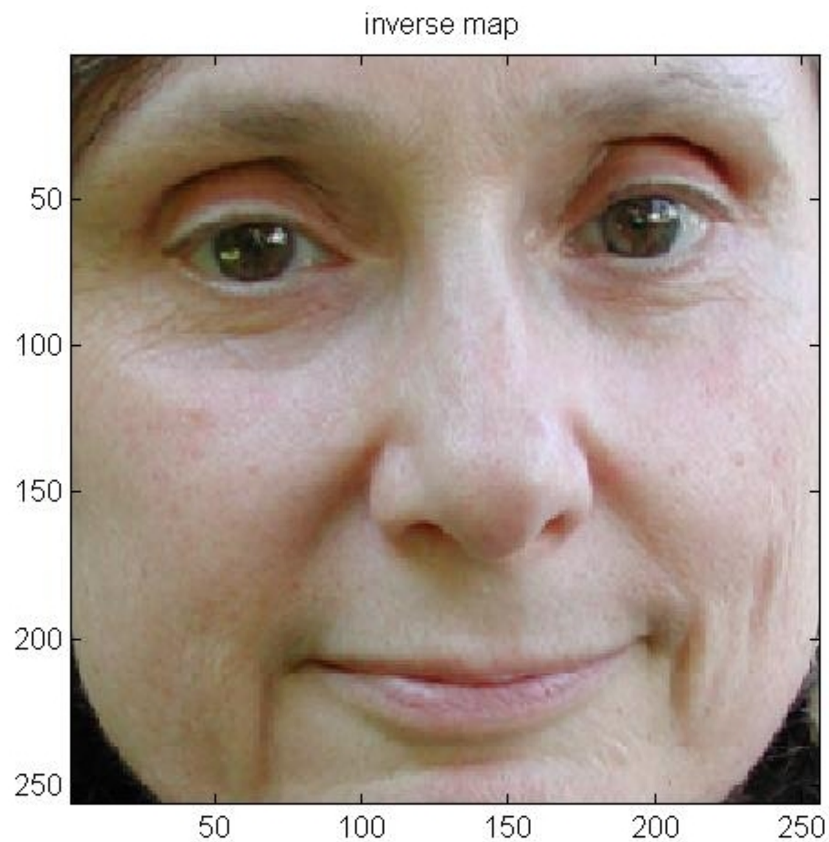
Inverse Mapping
Mandrill → Harriet



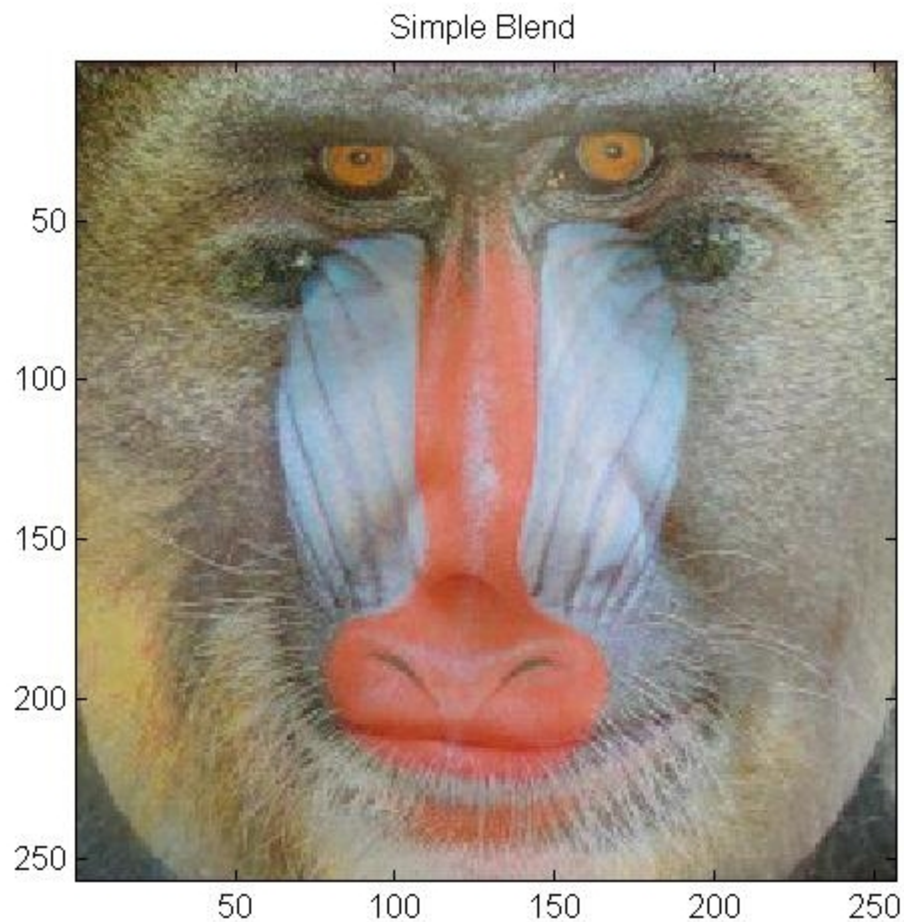


Inverse Mapping
Harriet → Mandrill



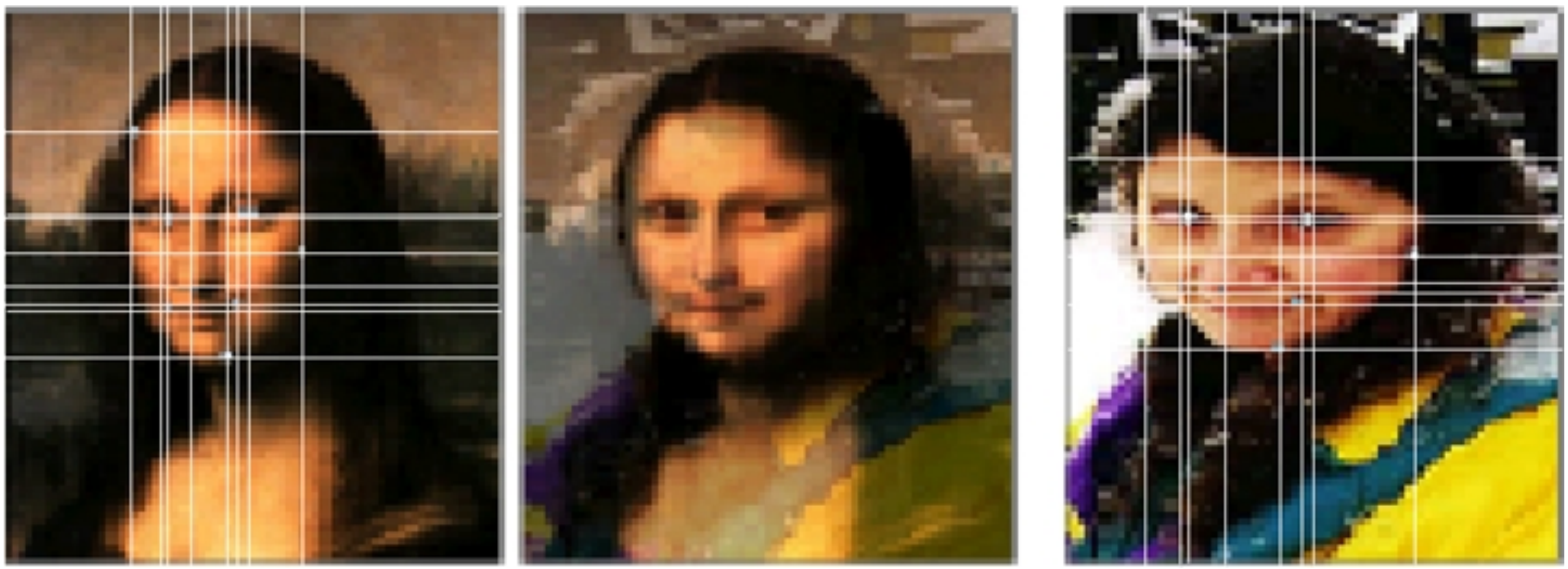


(harrietINV + mandrill)/2



Matching Points

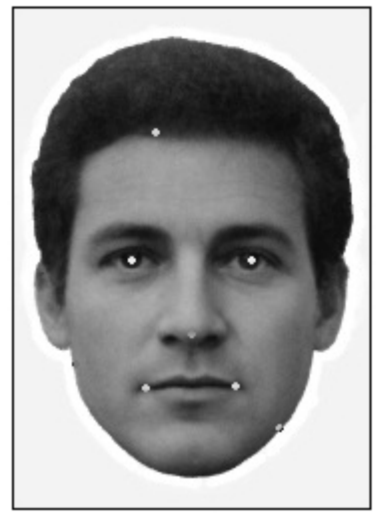
Matching Ponts
Rectangular Transforms



Halfway Blend

Image1

Image2



(1-t)Image1 + (t)Image2

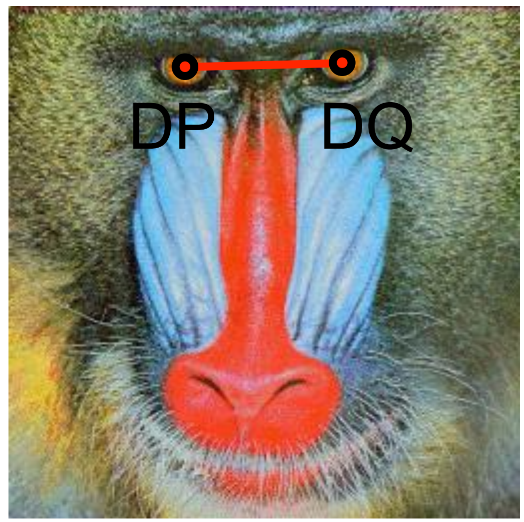
T = .5

Caricatures
Extreme Blends

t = 1.5

Harriet & Mandrill Matching Eyes

Match the endpoints of a line in the source with the endpoints of a line in the destination.

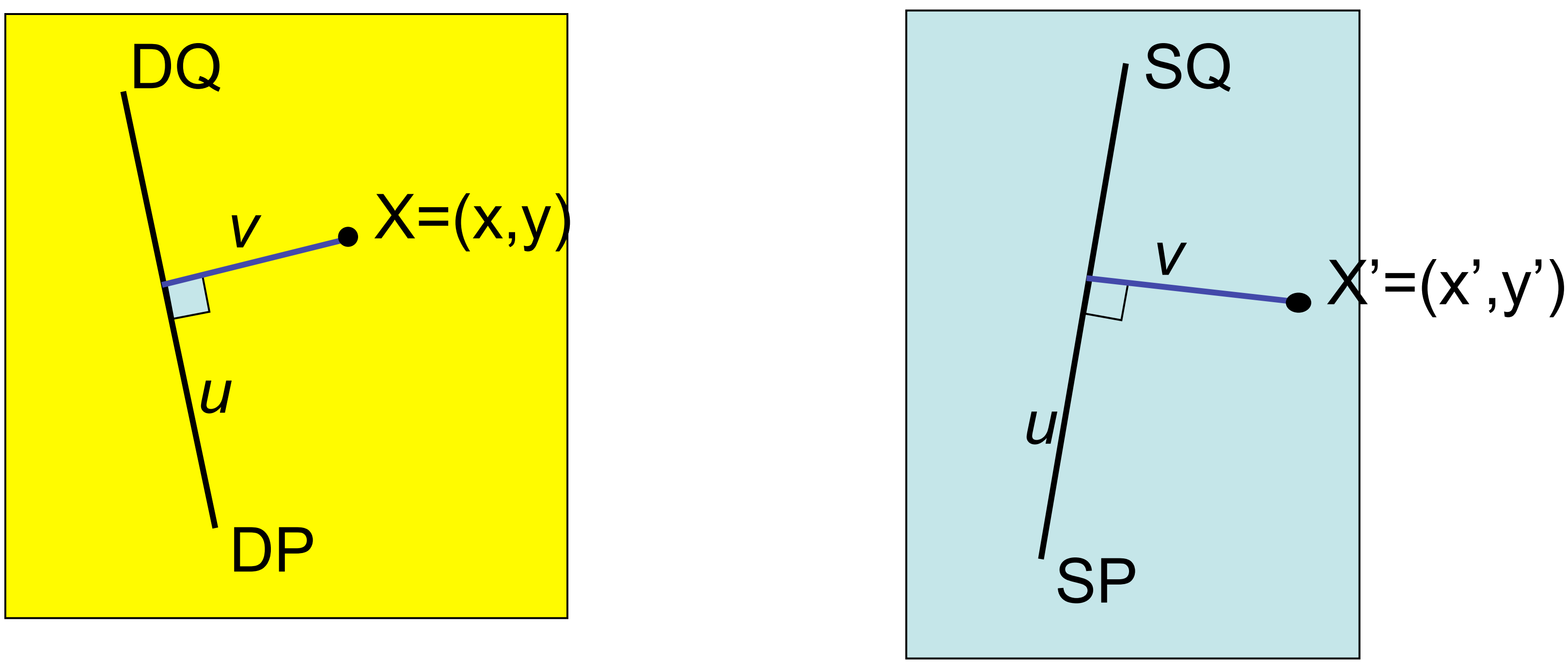


Harriet 276x293

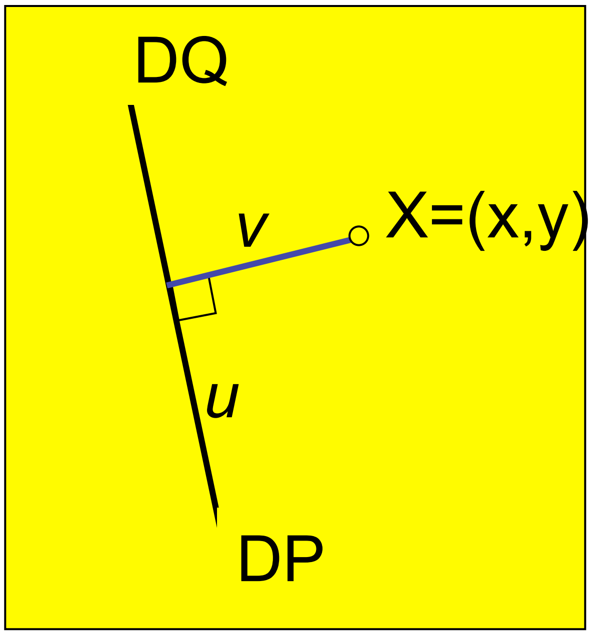
Mandrill 256x256

Line Pair Map

The *line pair map* takes the source image to an image the same size as the destinations and take the line segment in the source to the line segment in the destination.



Finding *u* and *v*



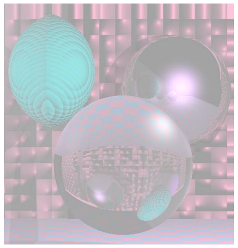
$$u = \frac{(X - DP)\cdot(DQ - DP)}{\|DQ - DP\|^2}$$

$$v = \frac{(X - DP)\cdot perp(DQ - DP)}{\|DQ - DP\|}$$

$$X' = SP + u \times (SQ - SP) + \frac{v \times perp(SQ - SP)}{\|SQ - SP\|}$$

u is the proportion of the distance from DP to DQ.

v is the distance to travel in the perpendicular direction.



linePairMap.m header

```
% linePairMap.m
% Scale image Source to one size DW, DH with line pair mapping
function Dest = forwardMap(Source, DW, DH, SP, SQ, DP, DQ);
% Source is the source image
% DW is the destination width
% DH is the destination height
% SP, SQ are endpoints of a line segment in the Source [y, x]
% DP, DQ are endpoints of a line segment in the Dest [y, x]
```

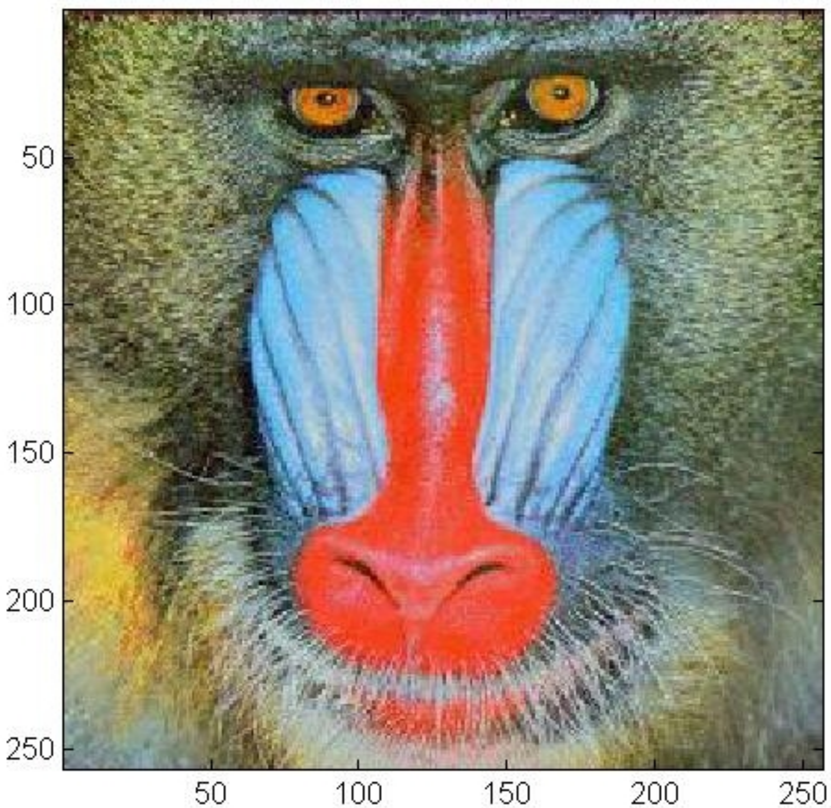
linePairMap.m body

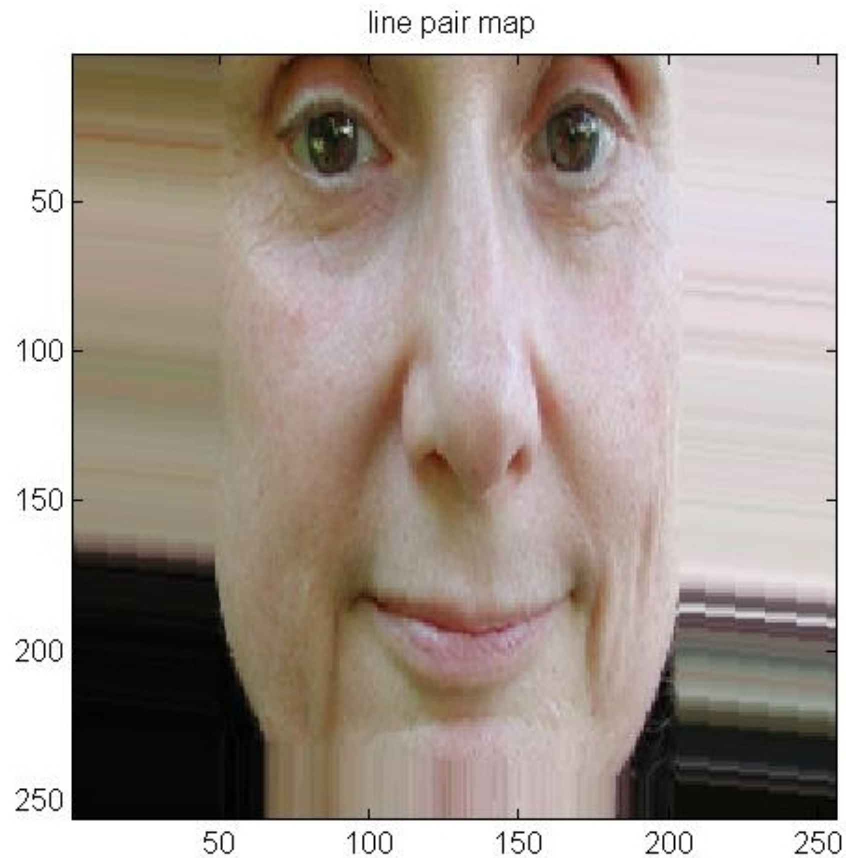
```
Dest = zeros(DH, DW,3);  % rows x columns x RGB
SW = length(Source(1,:,1));  % source width
SH = length(Source(:,1,1));   % source height
for y= 1:DH
   for x = 1:DW
     u = ([x,y]-DP)*(DQ-DP)'/((DQ-DP)*(DQ-DP)');
     v = ([x,y]-DP)*perp(DQ-DP)'/norm(DQ-DP);
     SourcePoint = SP+u*(SQ-SP) + v*perp(SQ-SP)/norm(SQ-SP);
     SourcePoint = max([1,1],min([SW,SH], SourcePoint));

     Dest(y,x,:)=Source(round(SourcePoint(2)),round(SourcePoint(1)),:);
   end;
end;
```

linePairMap.m extras

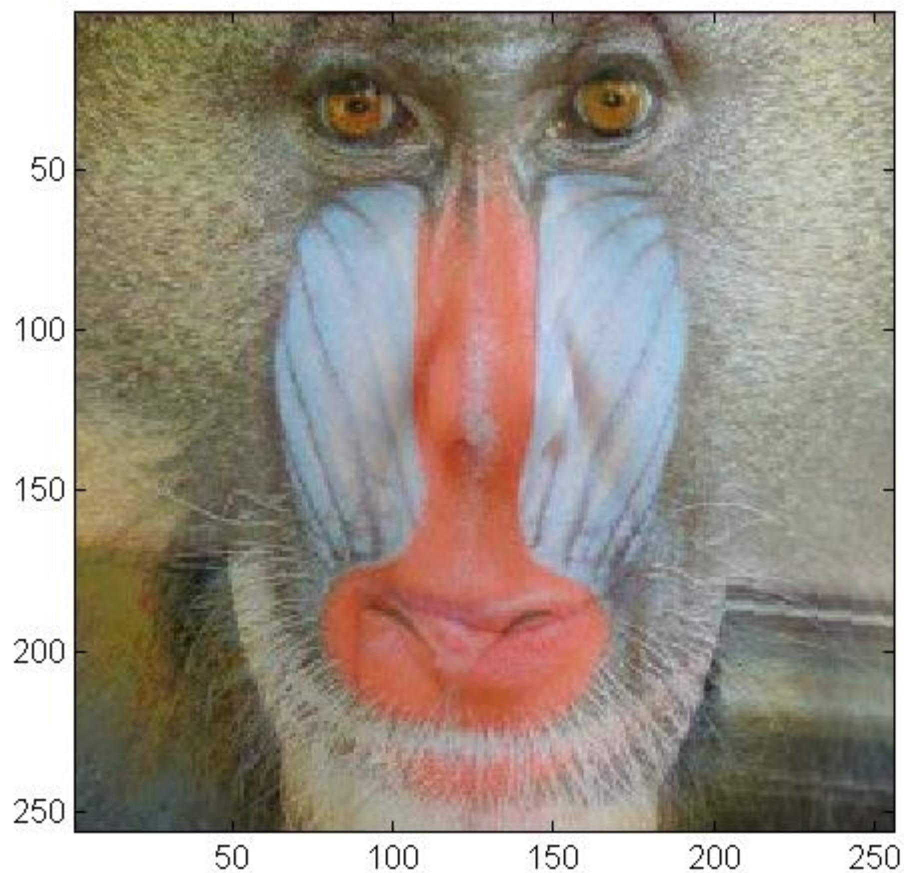
```
% display the image
figure, image(Dest/255,'CDataMapping','scaled');
axis equal;
title('line pair map');
xlim([1,DW]); ylim([1,DH]);


function Vperp = perp(V)
Vperp = [V(2), - V(1)];
```

Line Pair Map

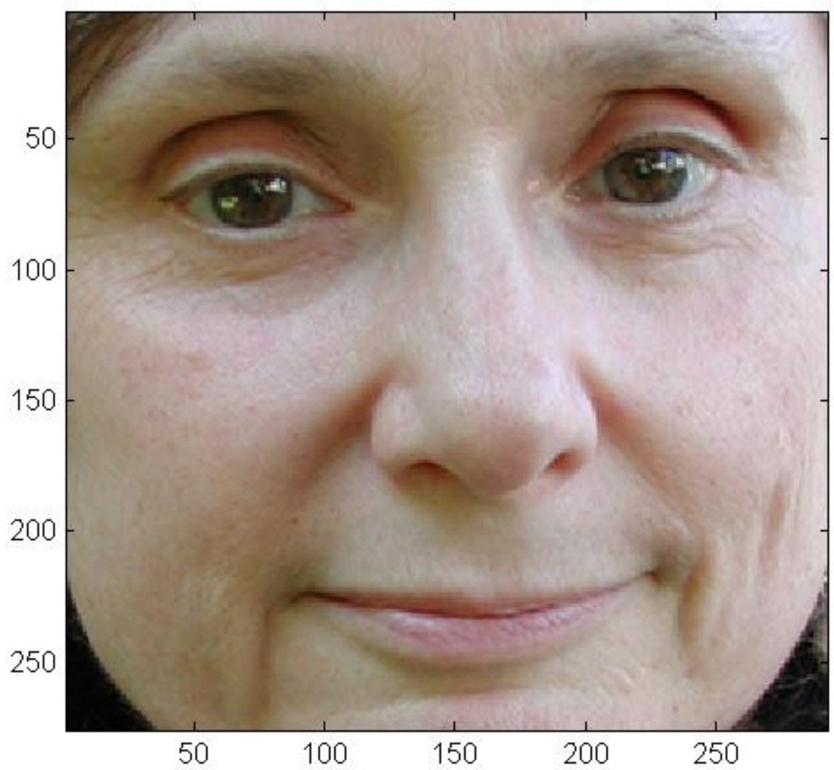


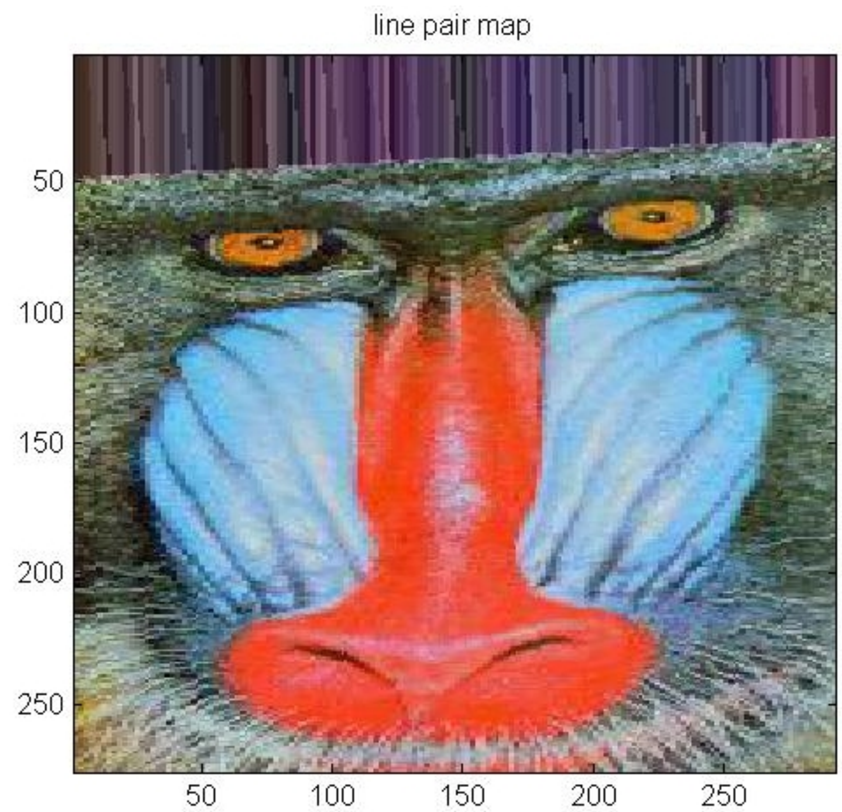


Line Pair Blend

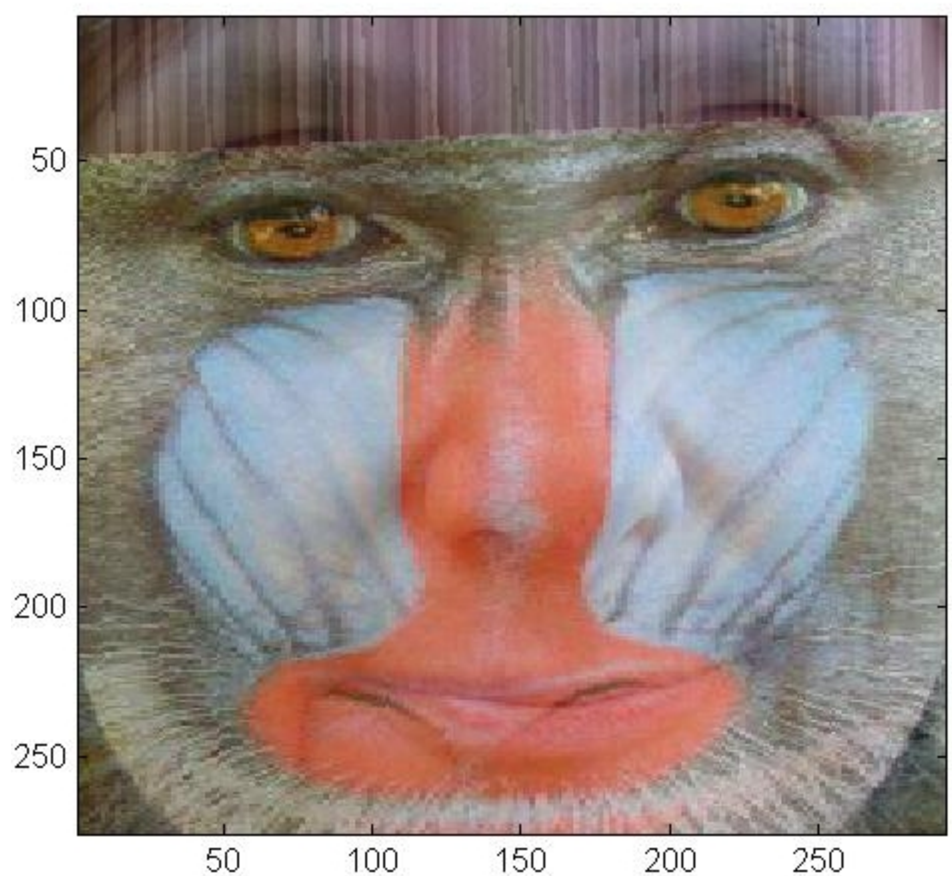


Line Pair Map 2

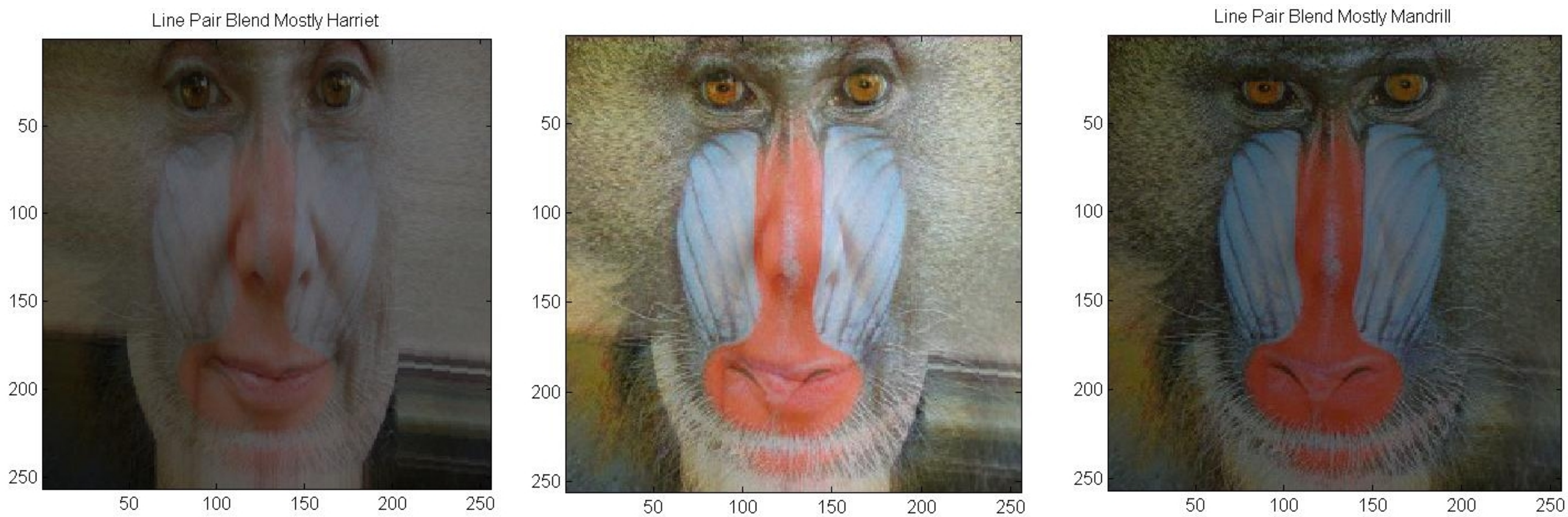




Line Pair Blend 2



Weighted Blends



Multiple Line Pairs

Find Xi' for the ith pair of lines.

Di = Xi' – X

Use a weighted average of the Di.

Weight is determined by the distance from X to the line.

$$weight = \left(\frac{length^{p}}{(a + dist)} \right)^{b}$$

length = length of the line

dist is the distance from the pixel to the line

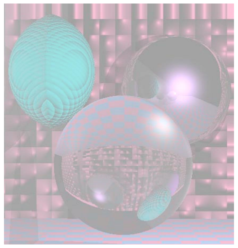
a, b, and p are used to change the relative effect of the lines.

Add average displacement to X to determine X‘.

Let’s Morph

MorphX

Time for a Break



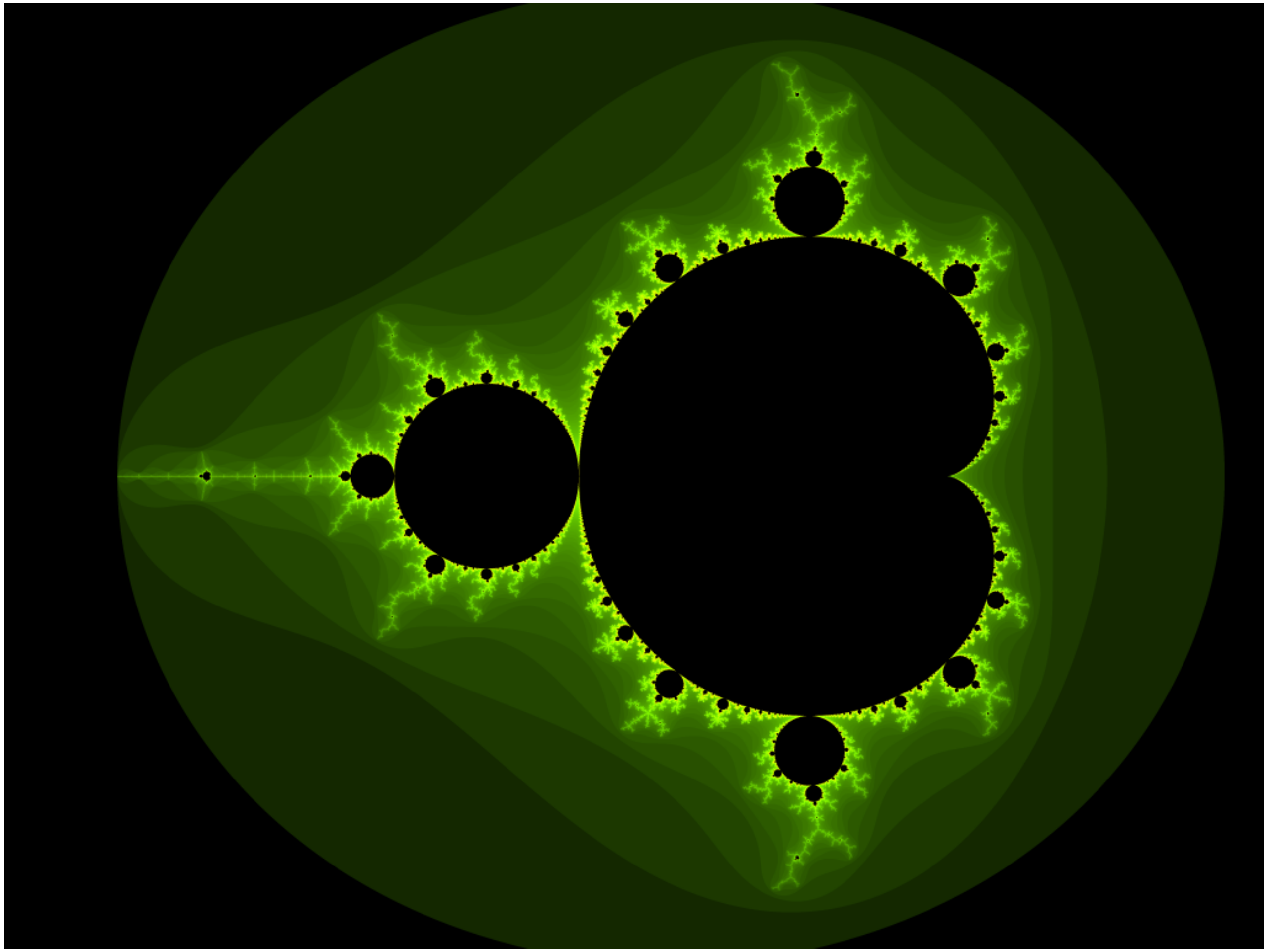
Fractals

The term *fractal* was coined in 1975 by Benoît Mandelbrot, from the Latin *fractus*, meaning "broken" or "fractured".

(colloquial) a shape that is recursively constructed or self-similar, that is, a shape that appears similar at all scales of magnification.

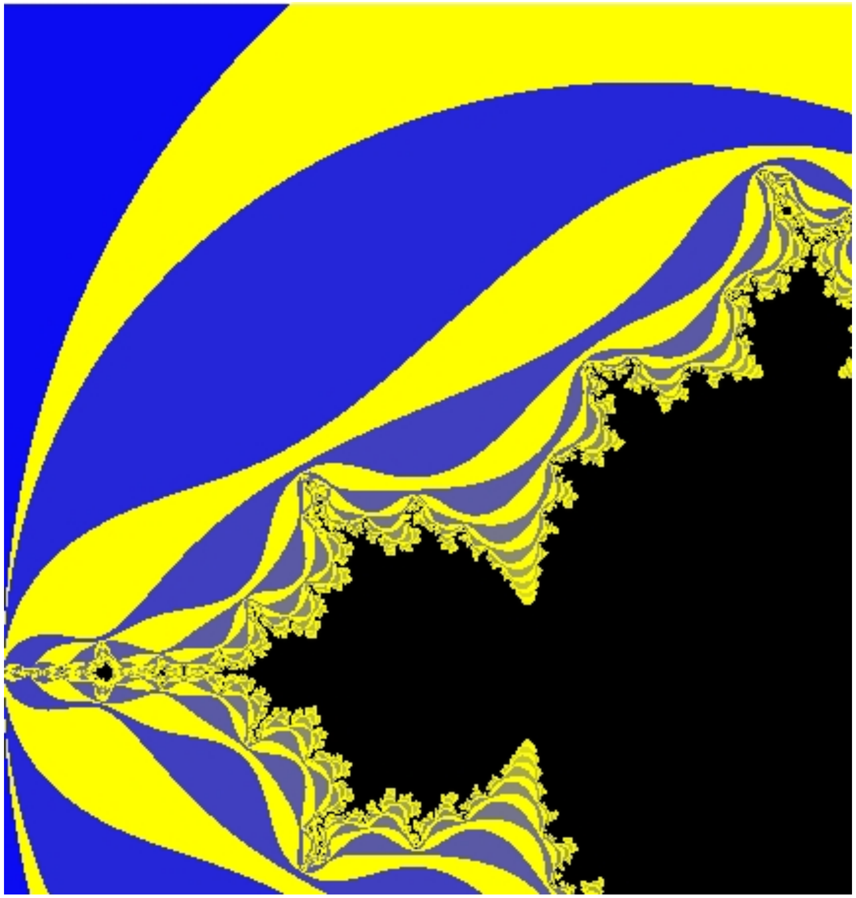
(mathematics) a geometric object that has a Hausdorff dimension greater than its topological dimension.

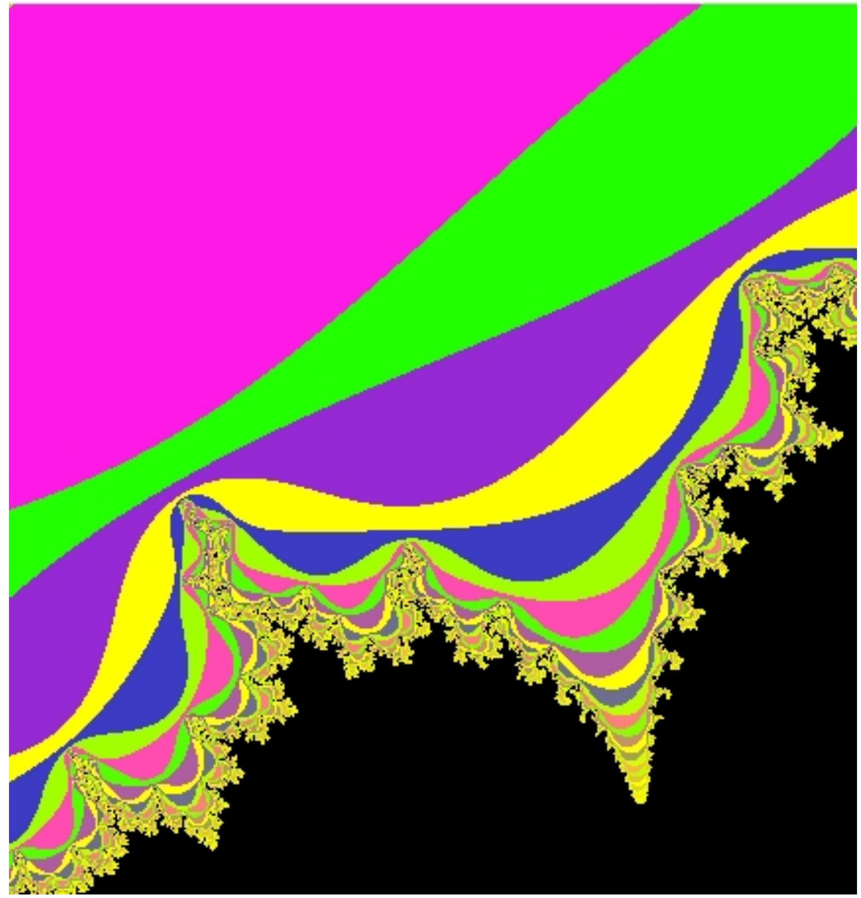
Mandelbrot Set



Mandelbrotset, rendered with Evercat's program.

Mandelbrot Set





What is the Mandelbrot Set?

We start with a quadratic function on the complex numbers.

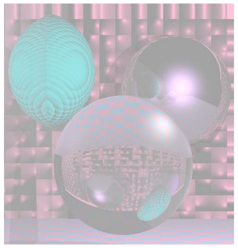
$$f_c : \mathbb{C} \to \mathbb{C}$$

$$z \mapsto z^2 + c$$

The *Mandelbrot Set* is the set of complex c such that

$$f_c^n(0) \not\to \infty$$

where f_c^n is the n-fold composition of f_c with itself.



Example

$$f(z) = z^2 - 1$$

$$f(0) = -1 \qquad f^2(0) = f(-1) = 1 - 1 = 0$$

$$f^n(0) = \begin{cases} -1 & n \text{ odd} \\ 0 & n \text{ even} \end{cases}$$

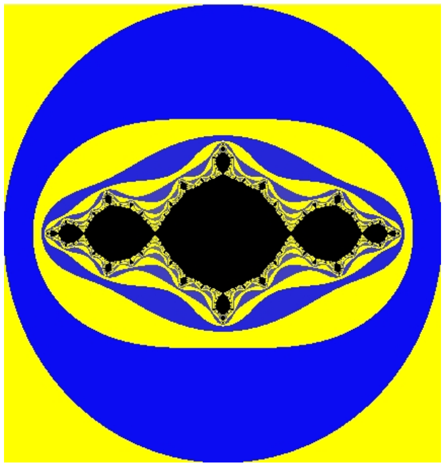
$$f(2) = 4 - 1 = 3 \qquad f^2(2) = f(3) = 9 - 1 = 8$$

$f^n(2)$ tend to ∞ as n tends to ∞.

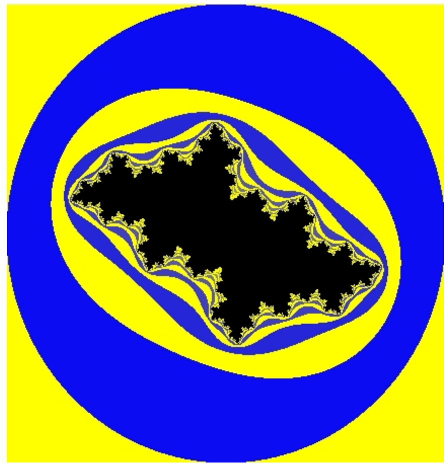
$$f(i) = i^2 - 1 = -2 \qquad f^2(i) = f(-2) = 4 - 1 = 3$$

$f^n(i)$ tend to ∞ as n tends to ∞.

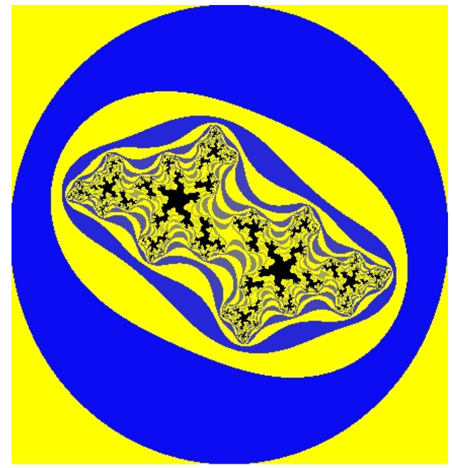
(Filled-in) Julia Sets



$c = -1$



$c = -.5 + .5i$



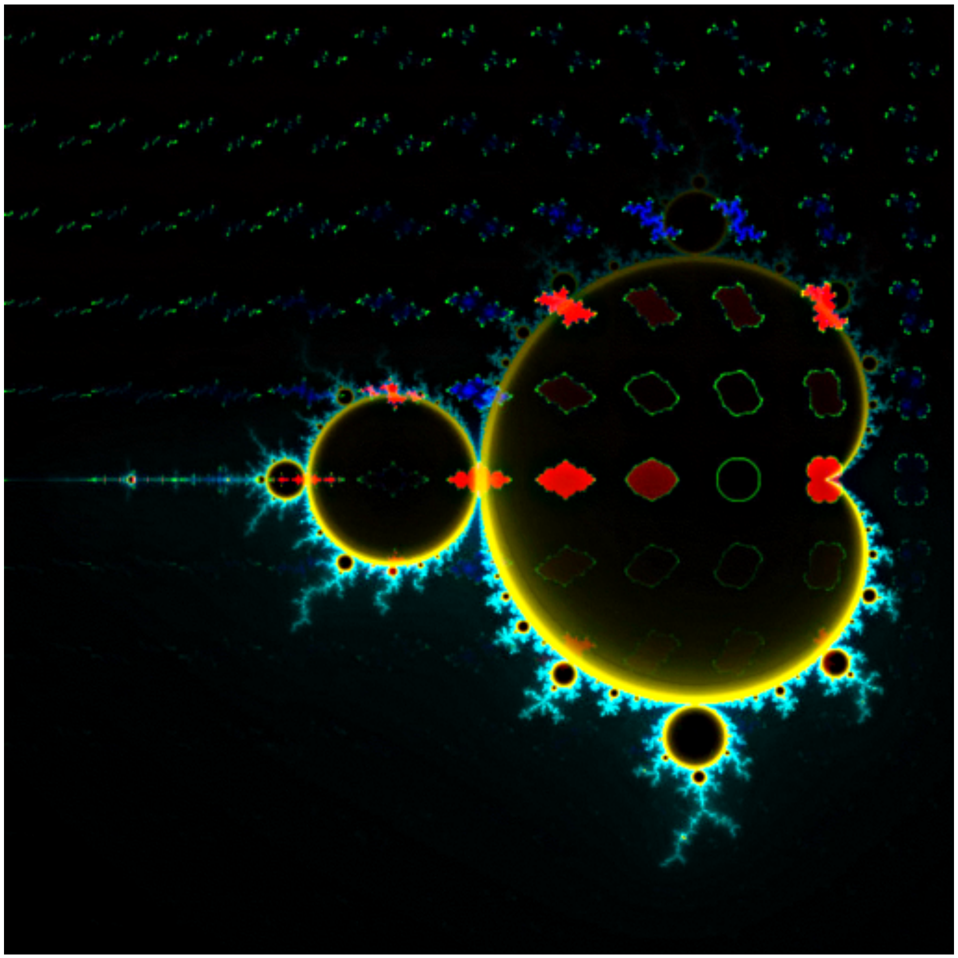
$c = - 5 + .5i$

$f_c : \mathbb{C} \to \mathbb{C}$

$z \mapsto z^2 + c$

The *Julia Set* of f_c is the set of points with 'chaotic' behavior under iteration.
The *filled-in Julia set* (or Prisoner Set), is the set of all *z* whos orbits do not tend towards infinity. The "normal" Julia set is the boundary of the filled-in Julia set.

Julia Sets and the Mandelbrot Set

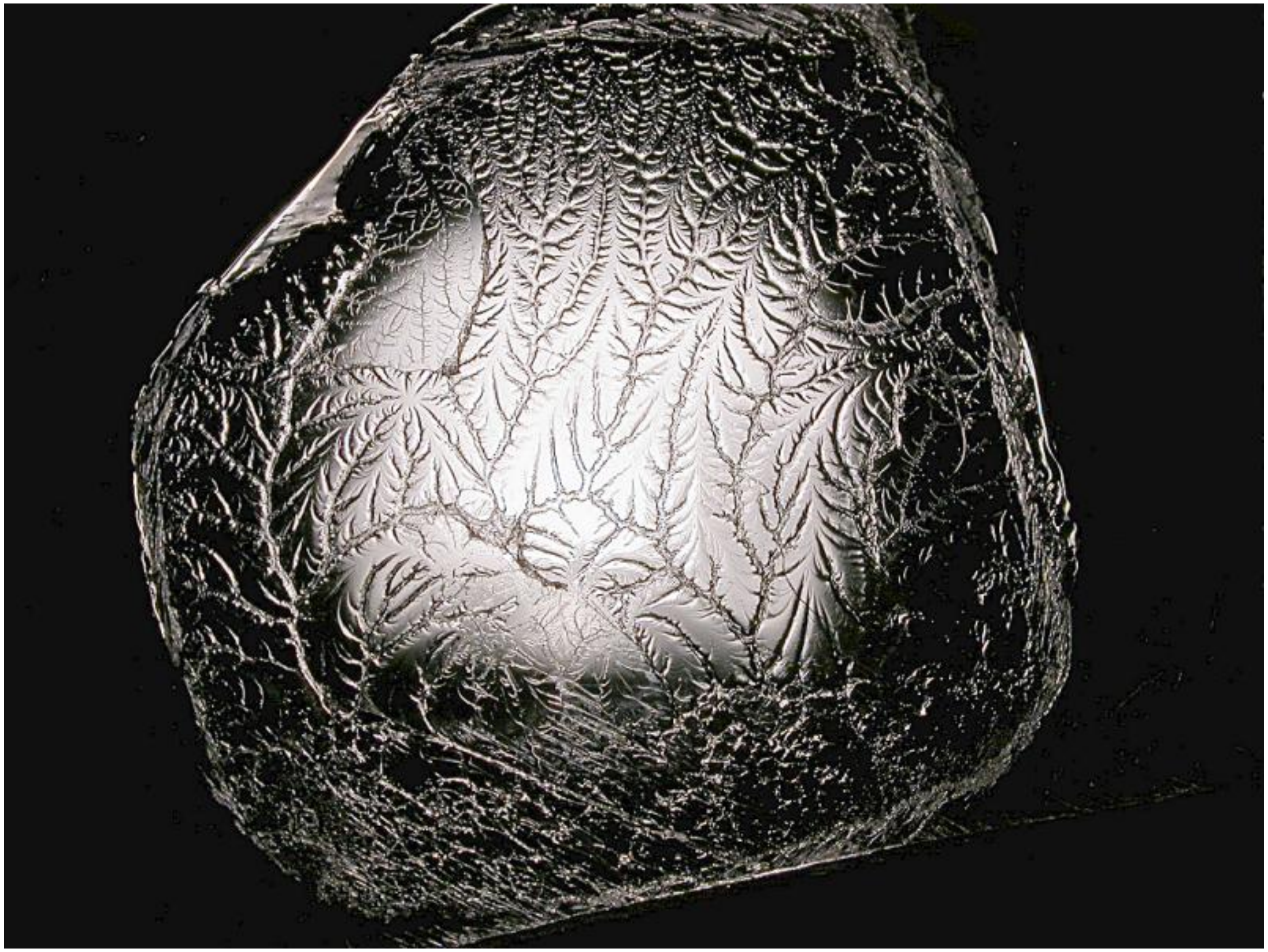


Some Julia sets are connected others are not.

The Mandelbrot set is the set of complex c for which the Julia set of $f_c(z) = z^2 + c$ is connected.

Map of 121 Julia sets in position over the Mandelbrot set (wikipedia)

A fractal is formed when pulling apart two glue-covered acrylic sheets.



Fractal Form of a Romanesco Broccoli

photo by Jon Sullivan

L-Systems

- An *L-system or Lindenmayer system*, after Aristid Lindenmayer (1925–1989), is a formal grammar (a set of rules and symbols) most famously used to model the growth processes of plant development, though able to model the morphology of a variety of organisms.
- L-systems can also be used to generate self-similar fractals such as iterated function systems.

L-System References

- Przemyslaw Prusinkiewicz & Aristid Lindenmayer, “The Algorithmic Beauty of Plants,” Springer, 1996.
- http://en.wikipedia.org/wiki/L-System

L-System Grammar

- **G** = {*V*, *S*, ω, *P*}, where
  - **V** (the *alphabet*) is a set of variables
  - **S** is a set of constant symbols
  - **ω** (*start*, *axiom* or *initiator*) is a string of symbols from **V** defining the initial state of the system
  - **P** is a set of *rules* or *productions* defining the way variables can be replaced with combinations of constants and other variables.
    - A production consists of two strings - the *predecessor* and the *successor*.

L-System Examples

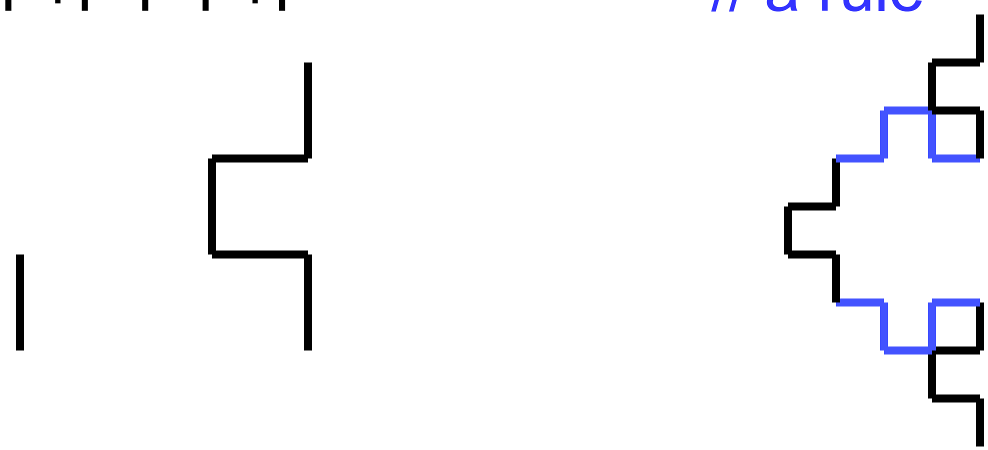
- **Koch curve** (from wikipedia)
- A variant which uses only right-angles.
  - **variables** : F
  - **constants** : + −
  - **start** : F
  - **rules** : (F → F+F−F−F+F)
- Here, *F* means "draw forward", + means "turn left 90°", and - means "turn right 90°" (see turtle graphics).

Turtle Graphics

```
class Turtle {
    double angle;      // direction of turtle motion in degrees
    double X;          // current x position
    double Y;          // current y position
    double step;       // step size of turtle motion
    boolean pen;       // true if the pen is down

 public void forward(Graphics g)
// moves turtle forward distance step in direction angle

public void turn(double ang)
// sets angle = angle + ang;

public void penDown(), public void penUp()
// set pen to true  or  false
}
```

My L-System Data Files

Koch Triangle Form	// title
4	// number of levels to iterate
90	// angle to turn
F	// starting shape
F:F+F-F-F+F	// a rule



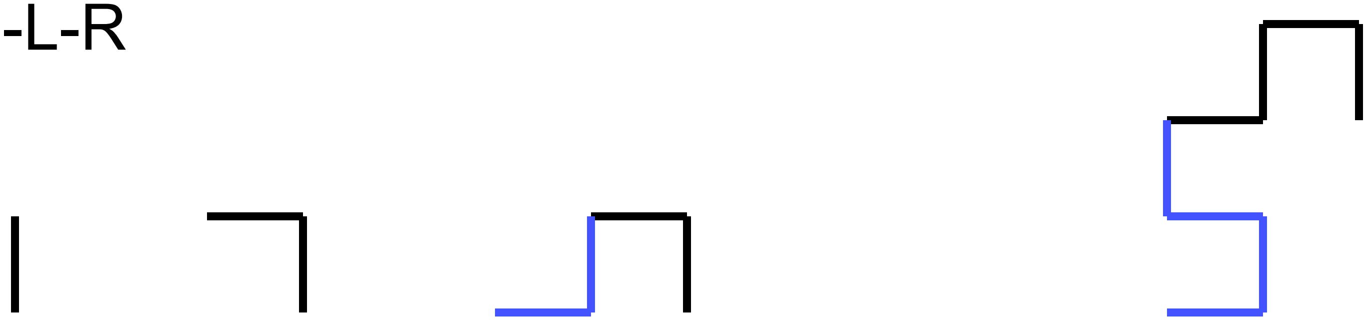
Go to Eclipse

F F+F-F-F+F F+F-F-F+F+F+F-F-F+F-F+F-F-F+F-F+F-F-F+F+F+F-F-F+F

More Variables

Dragon
10
90
L
L:L+R+
R:-L-R

When drawing, treat L and R just like F.

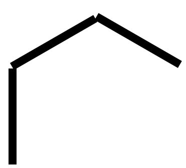


L   L+R+   L+R+ + -L-R +   L+R+ + -L-R + + - L +R+ - -L-R +

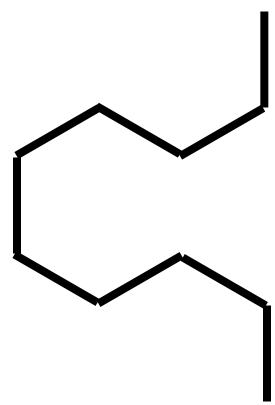
A Different Angle

Sierpinski Gasket
6
60
R
L:R+L+R
R:L-R-L

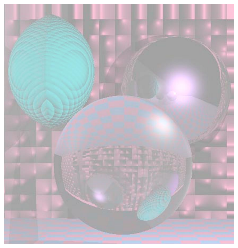
R



L-R-L

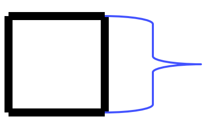


R+L+R- L-R-L -R+L+R



Moving with Pen Up

Islands and Lakes
2
90
F+F+F+F
F:F+f-FF+F+FF+Ff+FF-f+FF-F-FF-Ff-FFF
f:ffffff // f means move forward with the pen up

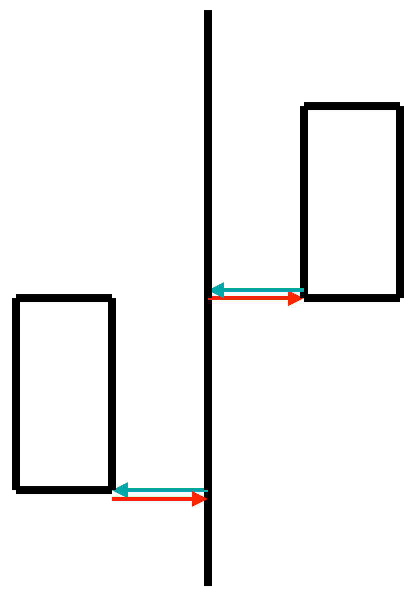


next slide

F+f-FF+F+FF+Ff+FF-f+FF-F-FF-Ff-FFF

F+F+F+F

Islands and Lakes
One Side of the Box

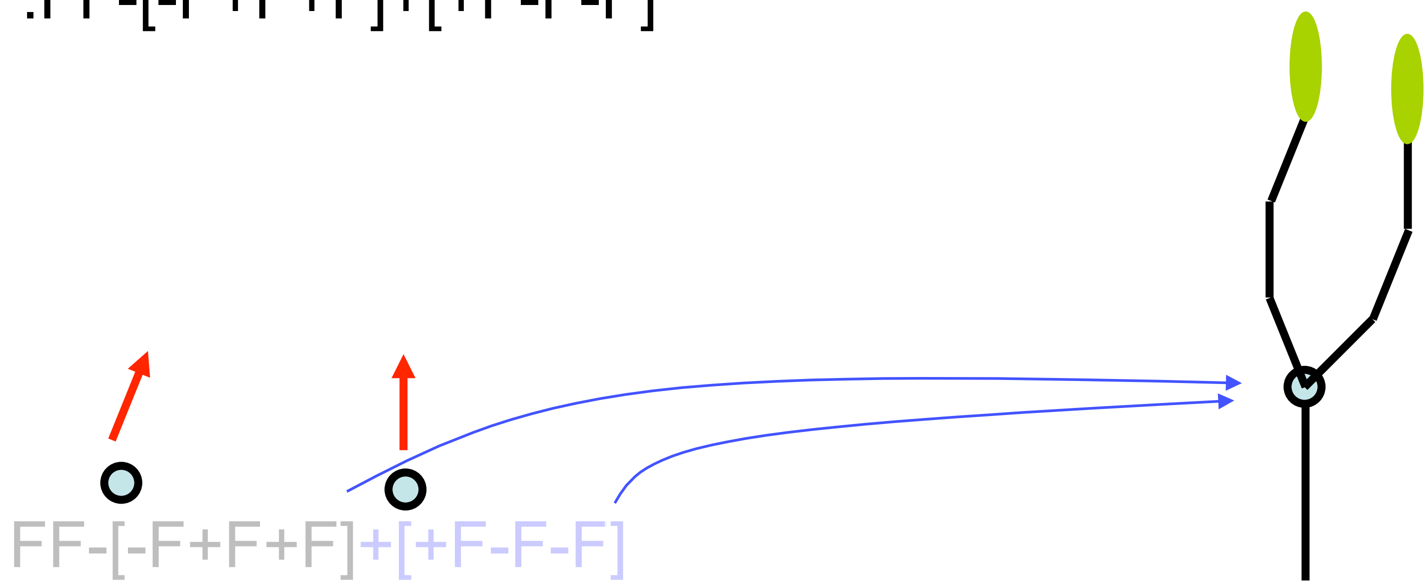


F+f-FF+F+FF+Ff+FF-f+FF-F-FF-Ff-FFF

Using a Stack to Make Trees

Tree1
4
22.5
F
F:FF-[-F+F+F]+[+F-F-F]

[push the turtle state onto the stack
] pop the turtle state from the stack
and I add leaves here



Stochastic L-Systems

http://algorithmicbotany.org/lstudio/CPFGman.pdf

seed: 2454 // different seeds for different trees

derivation length: 3

axiom: F

F--> F[+F]F[-F]F : 1/3

F--> F[+F]F : 1/3

F--> F[-F]F : 1/3

3D Turtle Rotations

Heading, Left, or, Up vector tell turtle direction.

$+(\theta)$ Turn left by angle θ° around the U axis.

$-(\theta)$ Turn right by angle θ° around the U axis.

$\&(\theta)$ Pitch down by angle θ° around the L axis.

$\wedge(\theta)$ Pitch up by angle θ° around the L axis.

$\backslash(\theta)$ Rollleftbyangleθ° around the H axis.

$/(\theta)$ Roll right by angle θ° around the H axis.

| Turn around 180° around the U axis.

@v Roll the turtle around the H axis so that H and U lie in a common vertical plane with U closest to up.

A Mint

http://algorithmicbotany.org/papers/

A model of a member of the mint family that exhibits a basipetal flowering sequence.