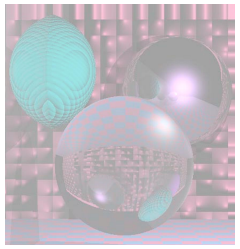
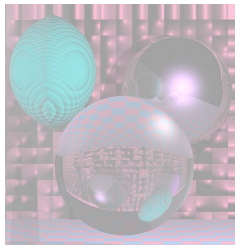


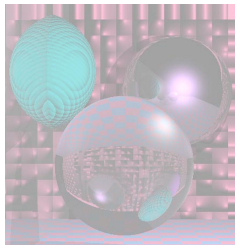
CS G140
Graduate Computer Graphics

Prof. Harriet Fell
Spring 2009
Lecture 10 – March 18, 2009



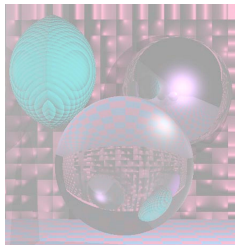
Today's Topics

- Morphing
- Quaternions
- Electronic Theater Program SIGGRAPH2006
  - 19 Guinness 'noitulovE"
  - 26 Foster's Australia 'Big Ad"



Morphing History

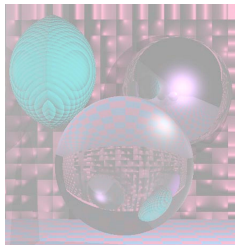
- *Morphing* is turning one image into another through a seamless transition.
- Early films used cross-fading picture of one actor or object to another.
- In 1985, "Cry" by Fodley and Crème, parts of an image fade gradually to make a smother transition.
- Early-1990s computer techniques distorted one 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"
    - » 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…



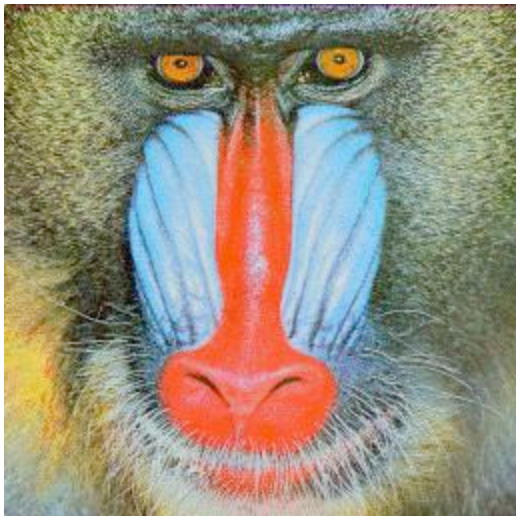
Feature Based Image Metamorphosis

Thaddeus Beier and Shawn Neely 1992

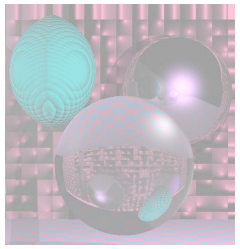
- The morph process consists
  - warping two images so that they have the same "shape"
  - cross dissolving the resulting images
- cross-dissolving is simple
- warping an image is hard

Harriet & Mandrill

Harriet 276x293



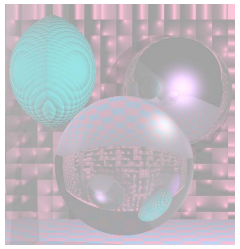
Mandrill 256x256



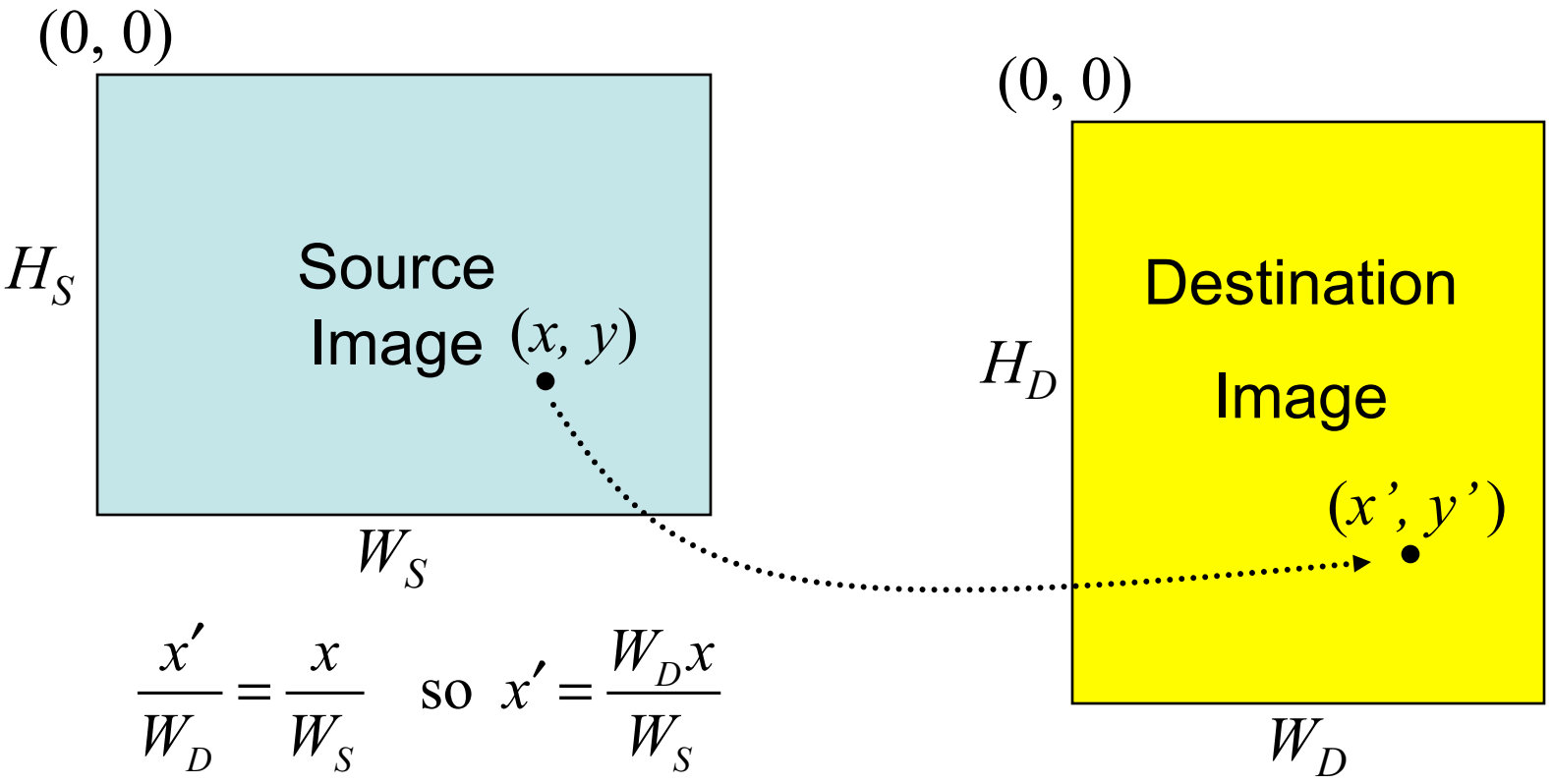
Warping an Image

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.

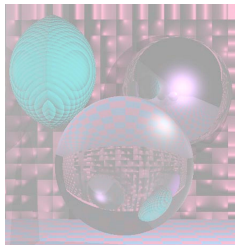


Forward Mapping

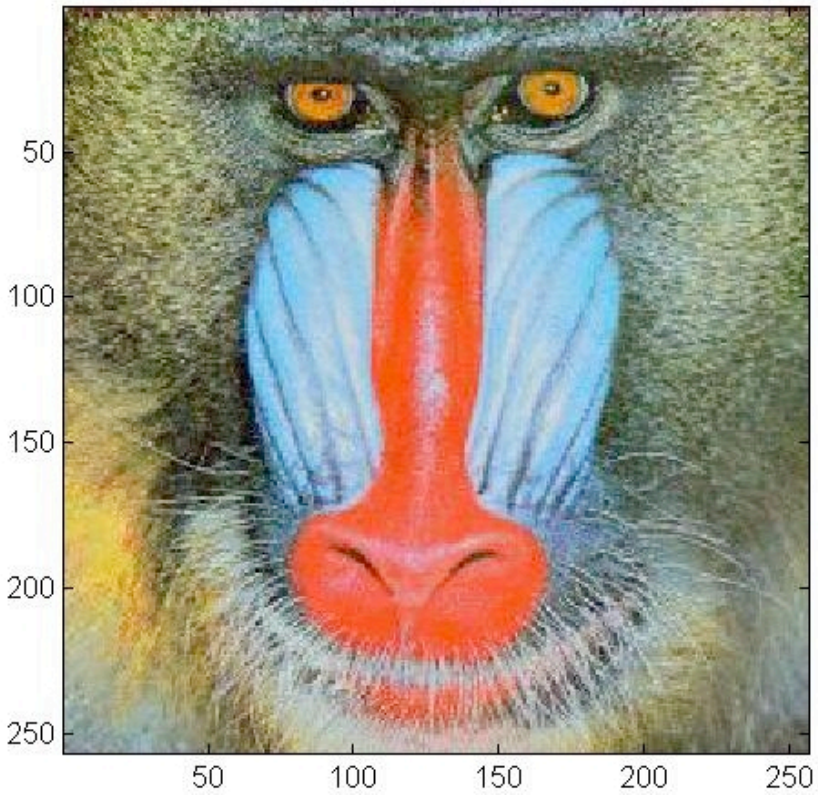


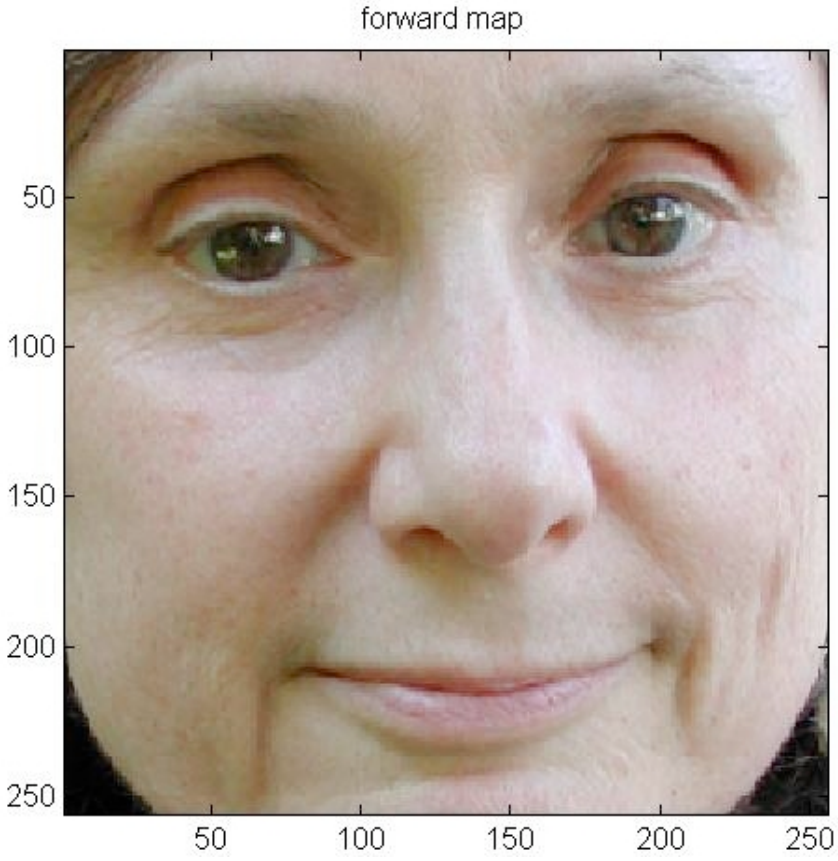
$$\frac{x'}{W_D} = \frac{x}{W_S} \quad \text{so} \quad x' = \frac{W_D x}{W_S}$$

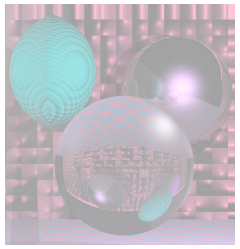
$$\frac{y'}{H_D} = \frac{y}{H_S} \quad \text{so} \quad y' = \frac{H_D y}{H_S}$$



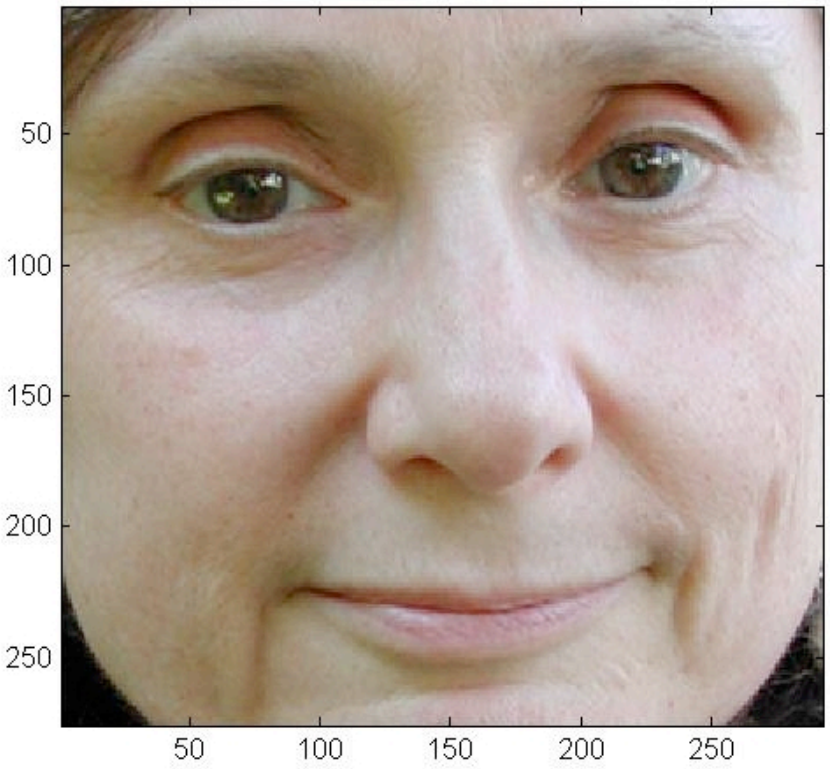
Forward Mapping
Harriet ➔ Mandrill

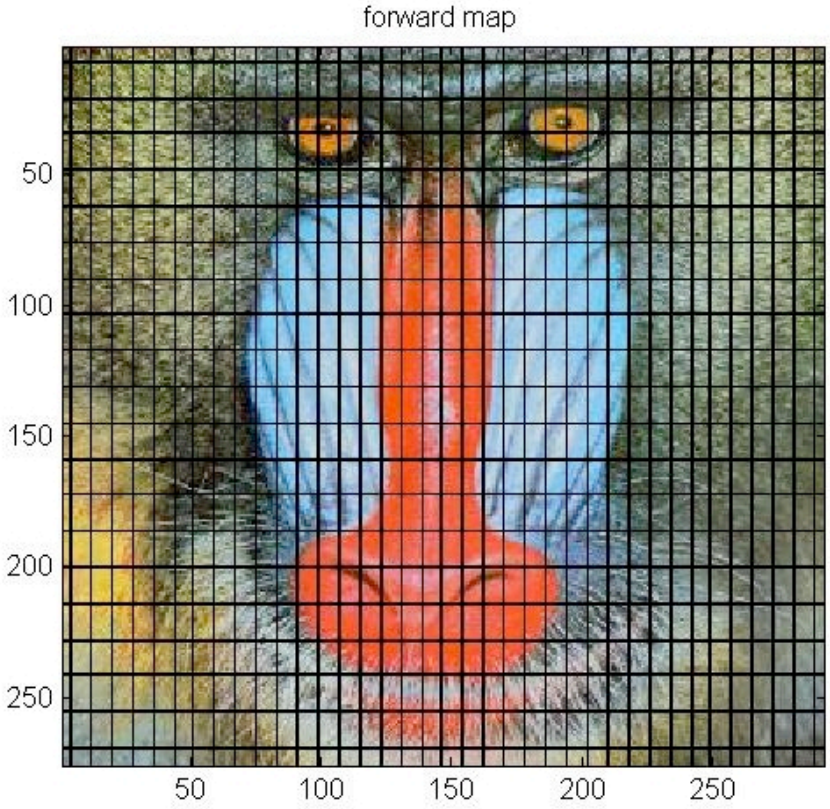


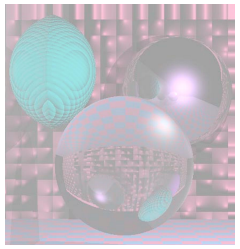




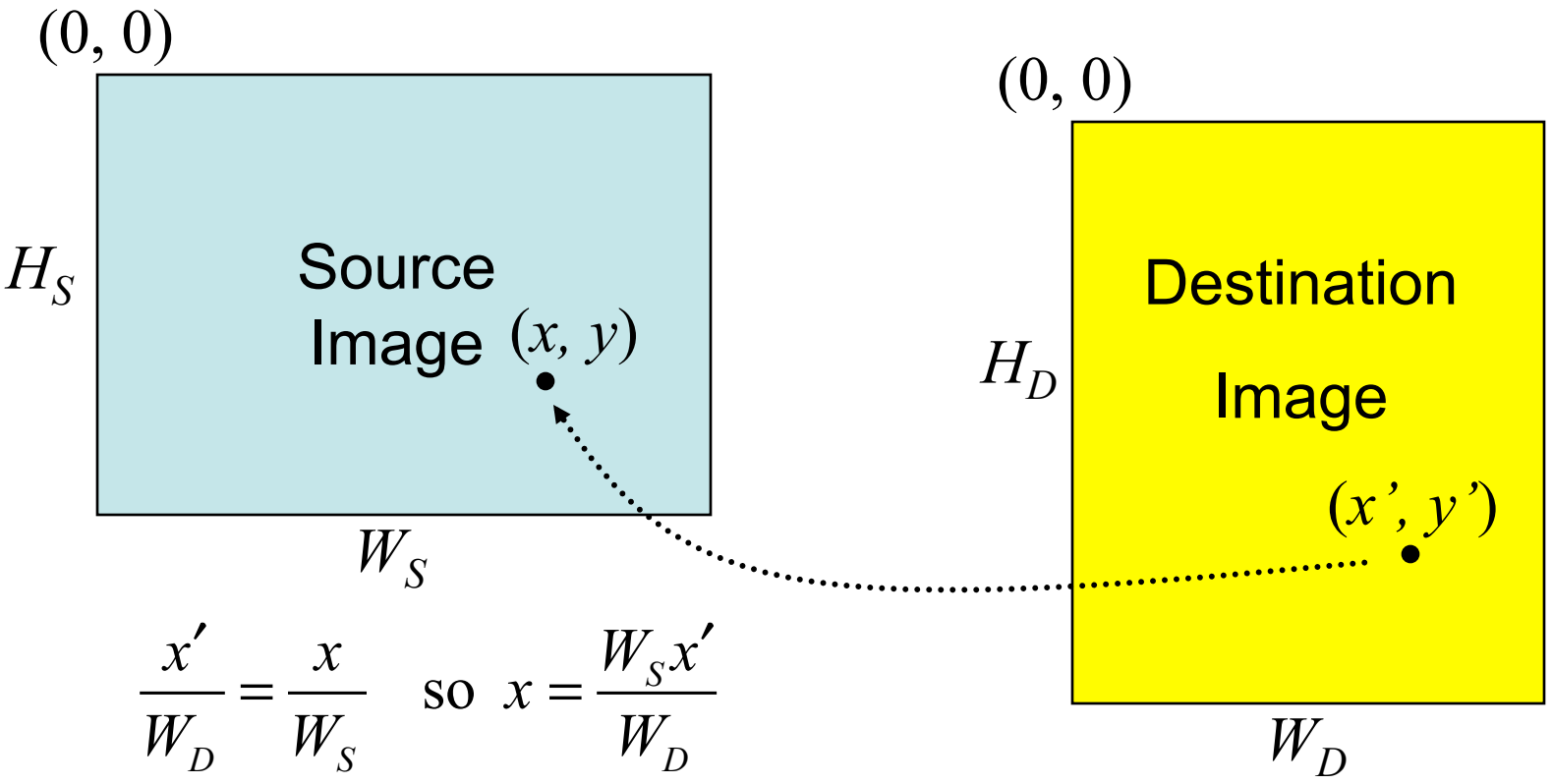
Forward Mapping
Mandrill ➔ Harriet





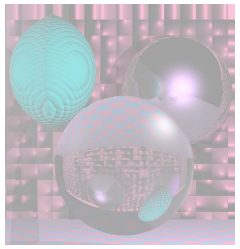


Inverse Mapping

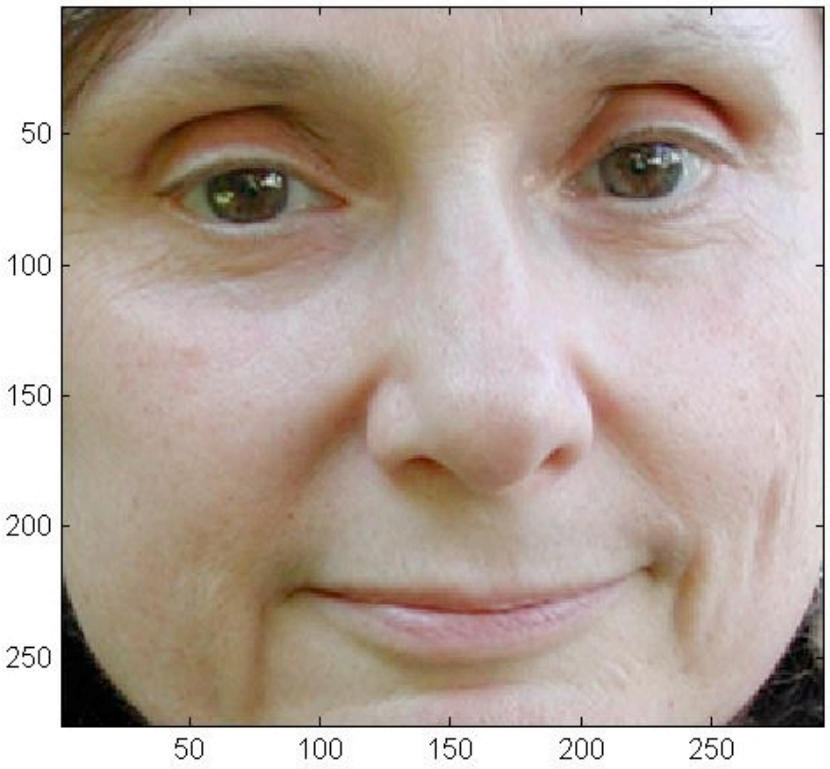


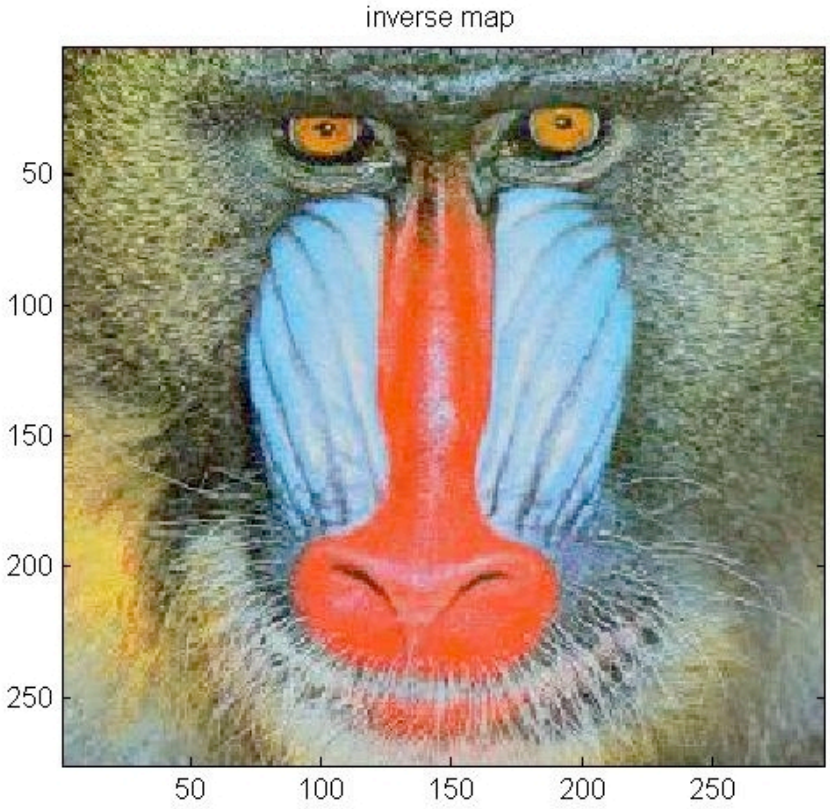
$$\frac{x'}{W_D} = \frac{x}{W_S} \quad \text{so} \quad x = \frac{W_S x'}{W_D}$$

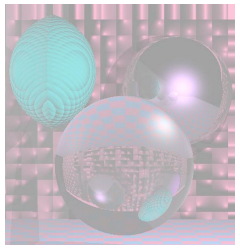
$$\frac{y'}{H_D} = \frac{y}{H_S} \quad \text{so} \quad y = \frac{H_S y'}{H_D}$$



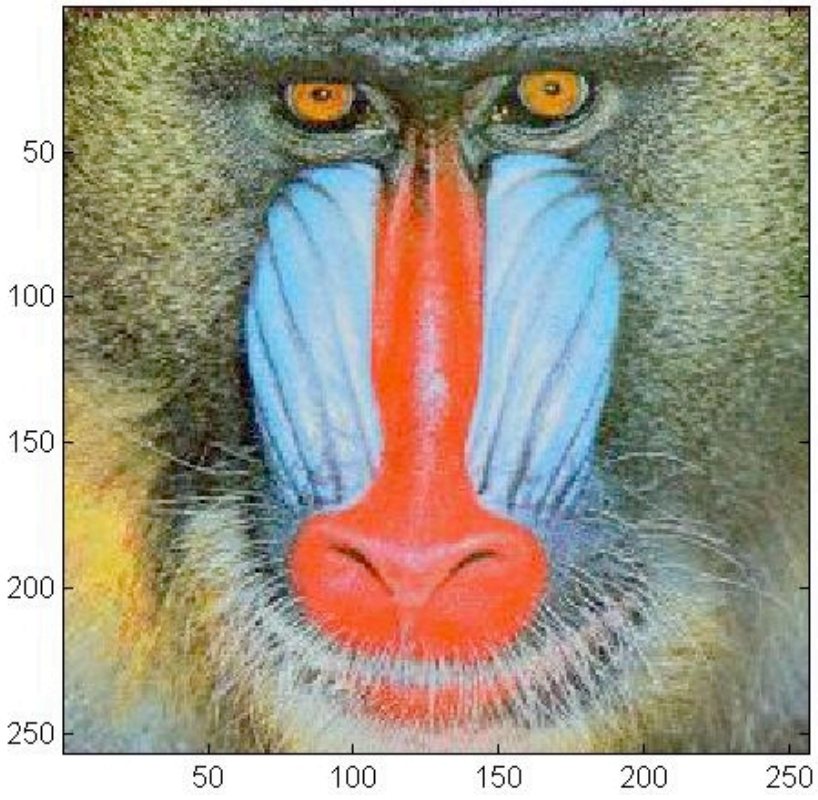
Inverse Mapping
Mandrill ➔ Harriet

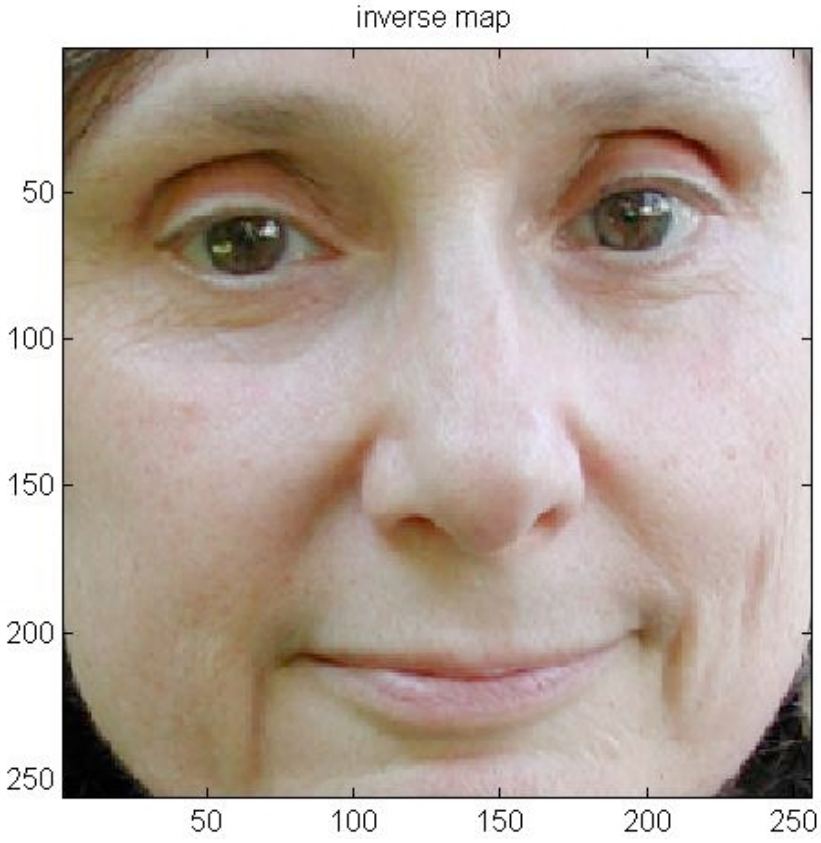


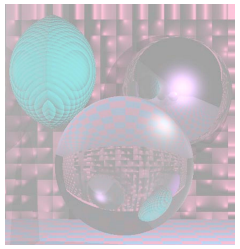




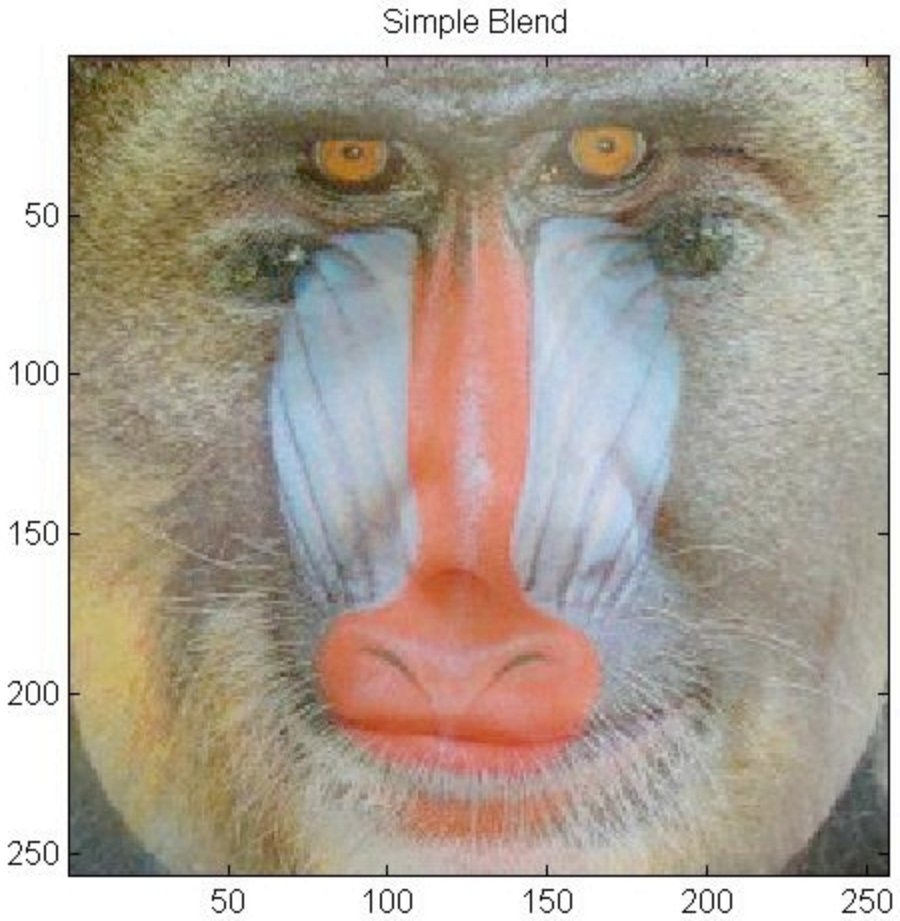
Inverse Mapping
Harriet ➔ Mandrill

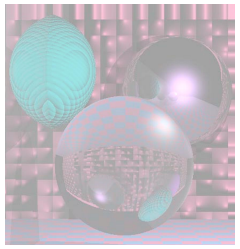




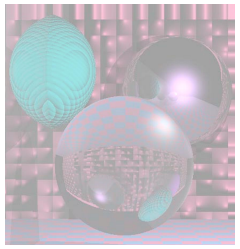


(harrietINV + mandrill)/2

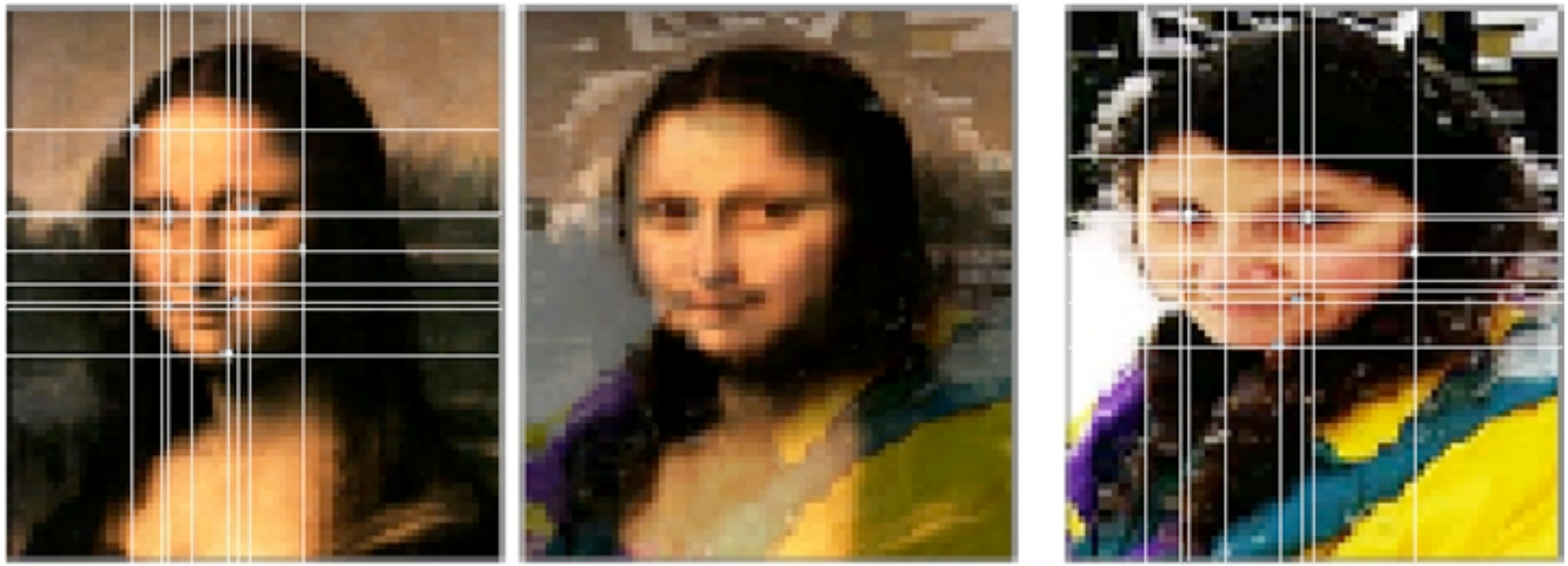


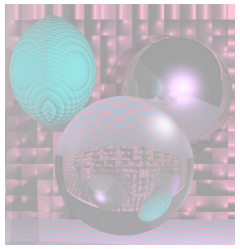


Matching Points



Matching Ponts Rectangular Transforms





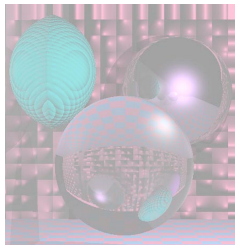
Halfway Blend

Image1

Image2

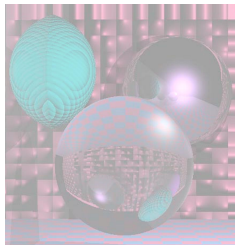
(1-t)Image1 + (t)Image2

T = .5



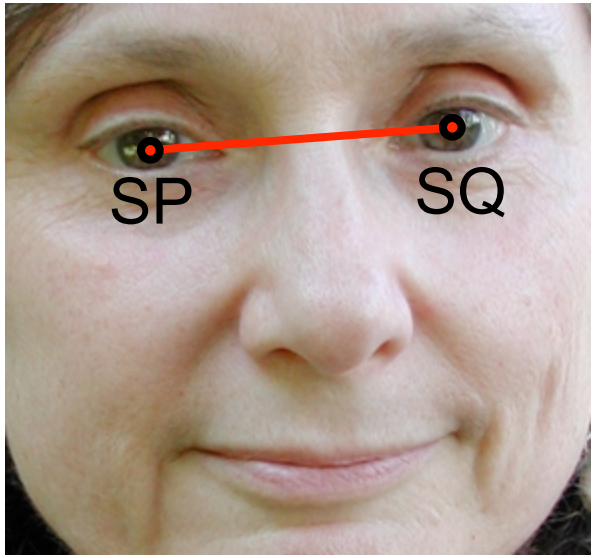
Caricatures
Extreme Blends

t = 1.5

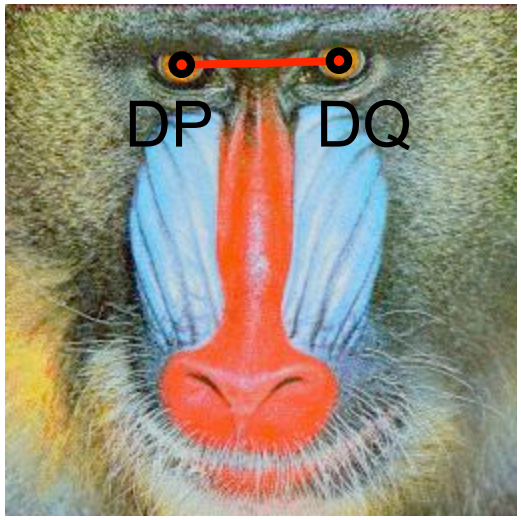


Harriet & Mandrill Matching Eyes

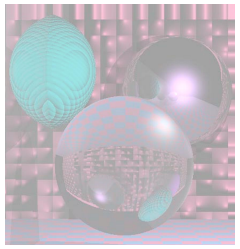
Match the endpoints of a line in the source with the endpoints of a line in the destination.



Harriet 276x293

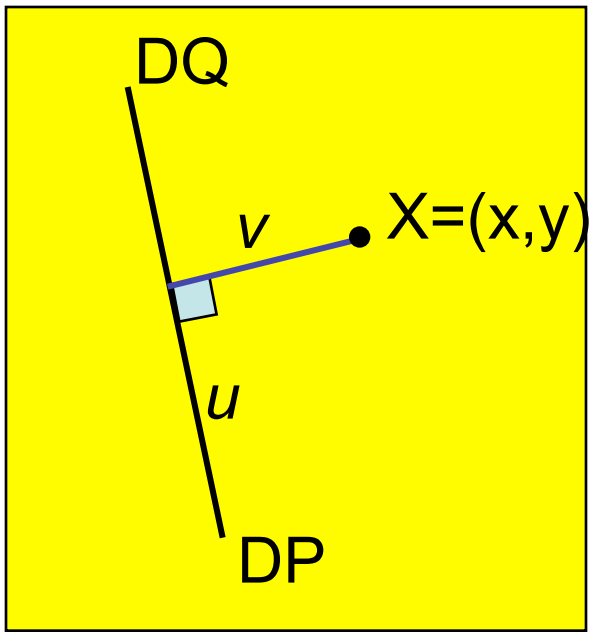


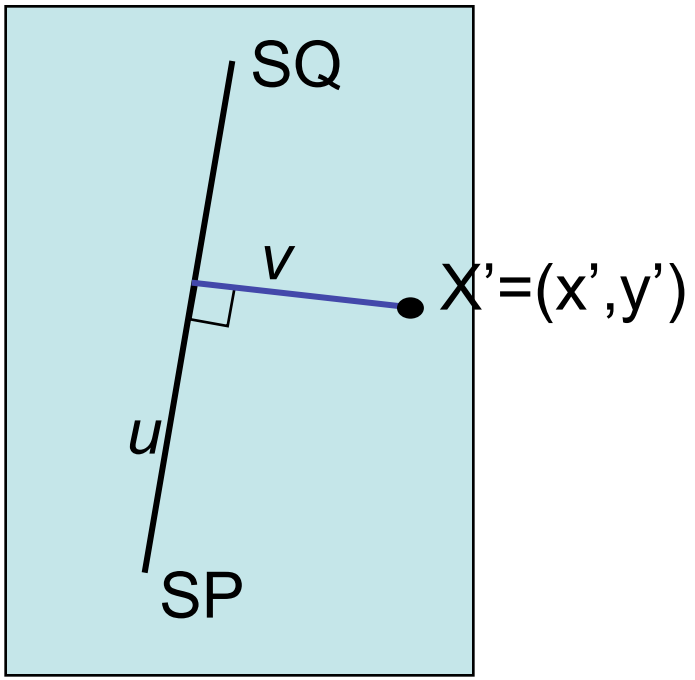
Mandrill 256x256

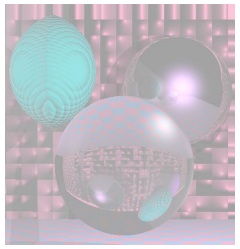


Line Pair Map

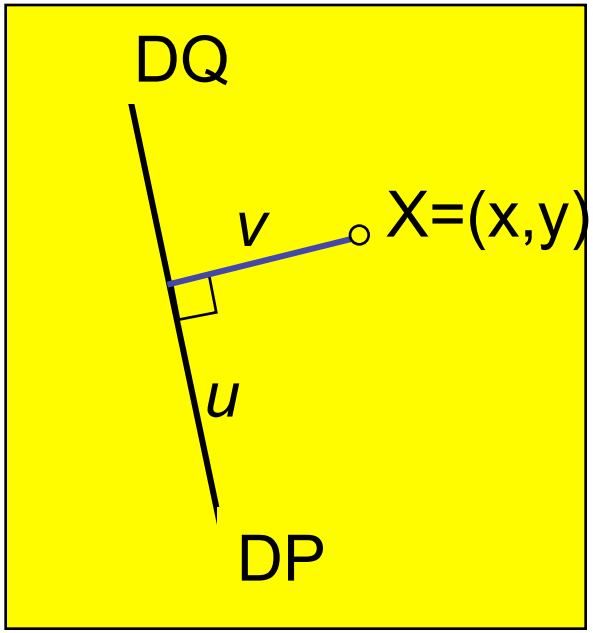
The *line pair map* takes the source image to an image the same size as the destinations and take the line segment in the source to the line segment in the destination.







Finding *u* and *v*



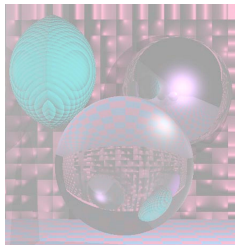
$$u = \frac{(X - DP) \cdot (DQ - DP)}{\|DQ - DP\|^2}$$

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

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

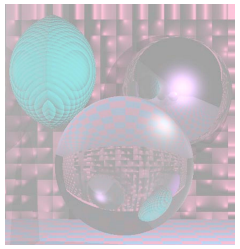
u is the proportion of the distance from DP to DQ.

v is the distance to travel in the perpendicular direction.



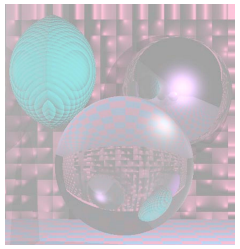
linePairMap.m header

```
% 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]
```



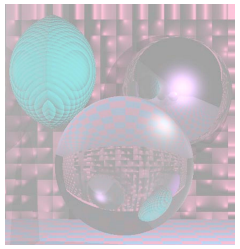
linePairMap.m body

```
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;
```

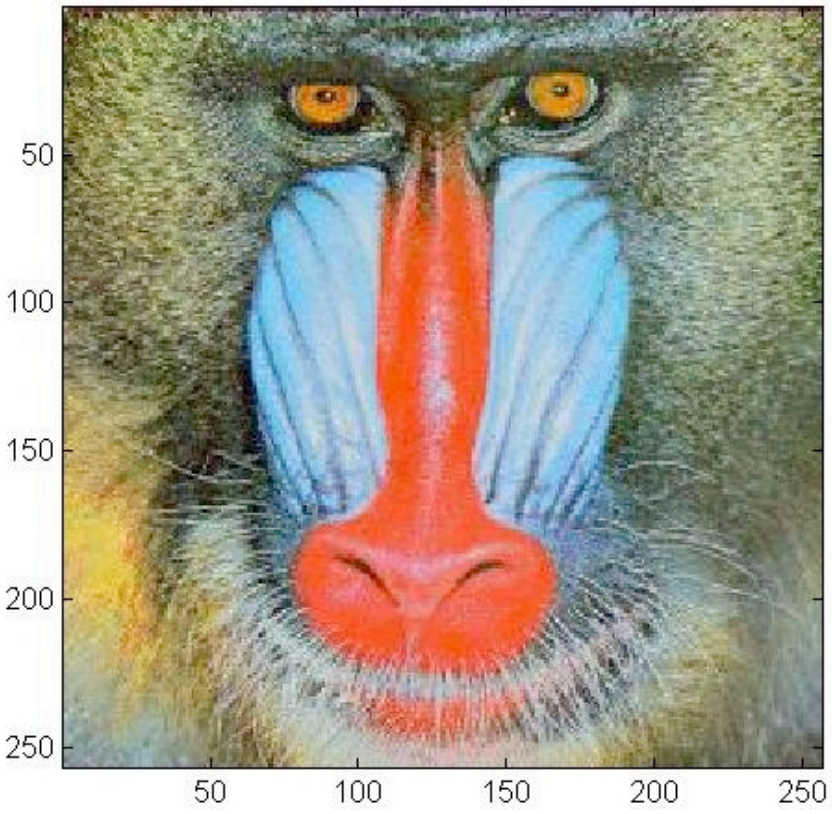


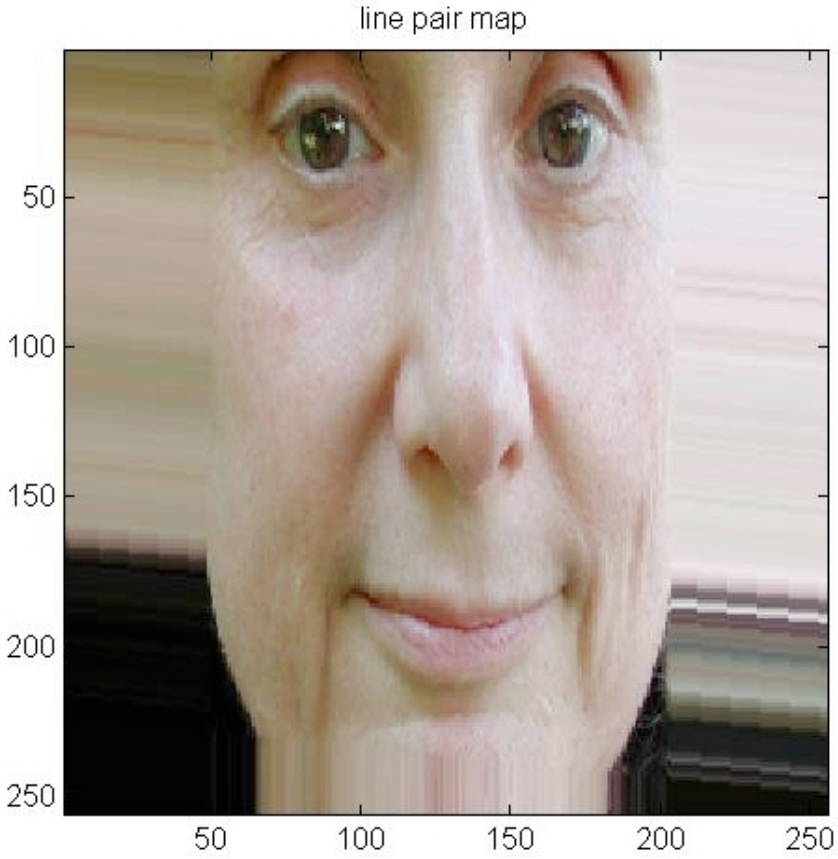
linePairMap.m extras

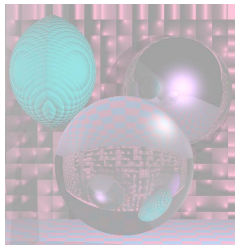
```
% display the image
figure, image(Dest/255,'CDataMapping','scaled');
axis equal;
title('line pair map');
xlim([1,DW]); ylim([1,DH]);


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



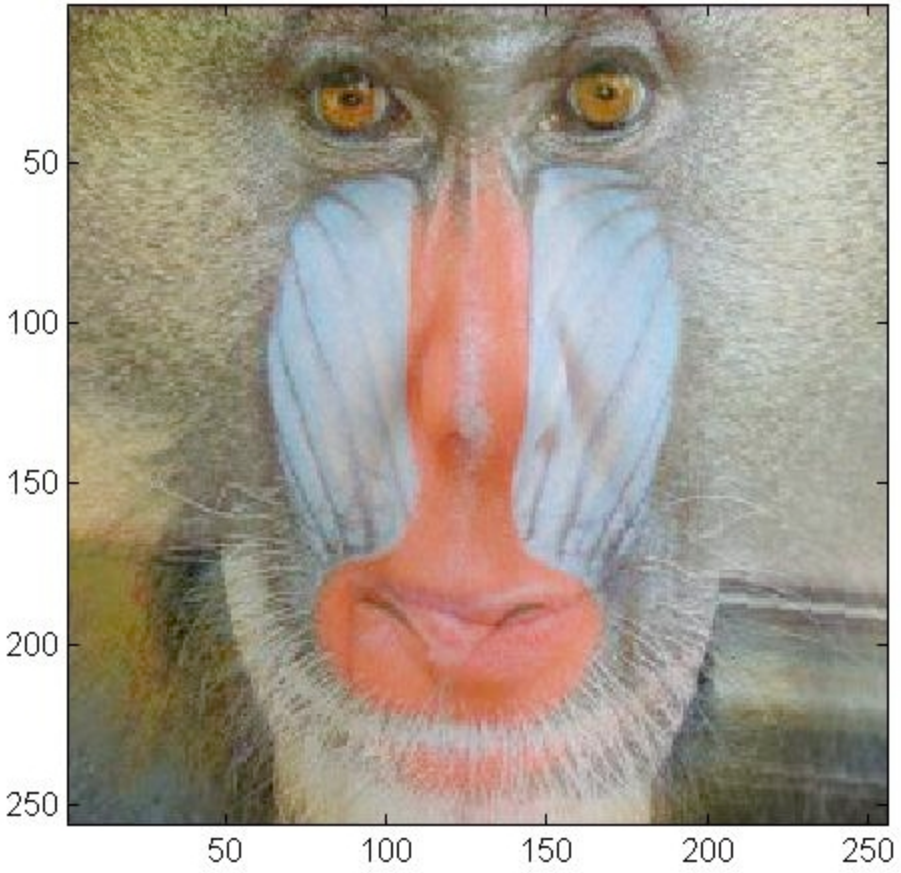
Line Pair Map



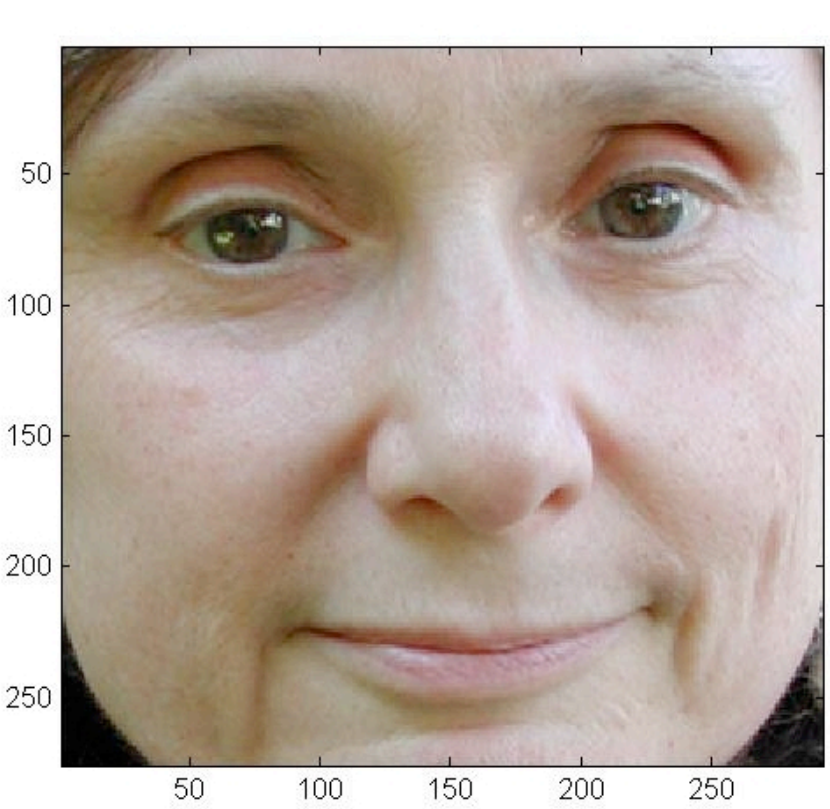


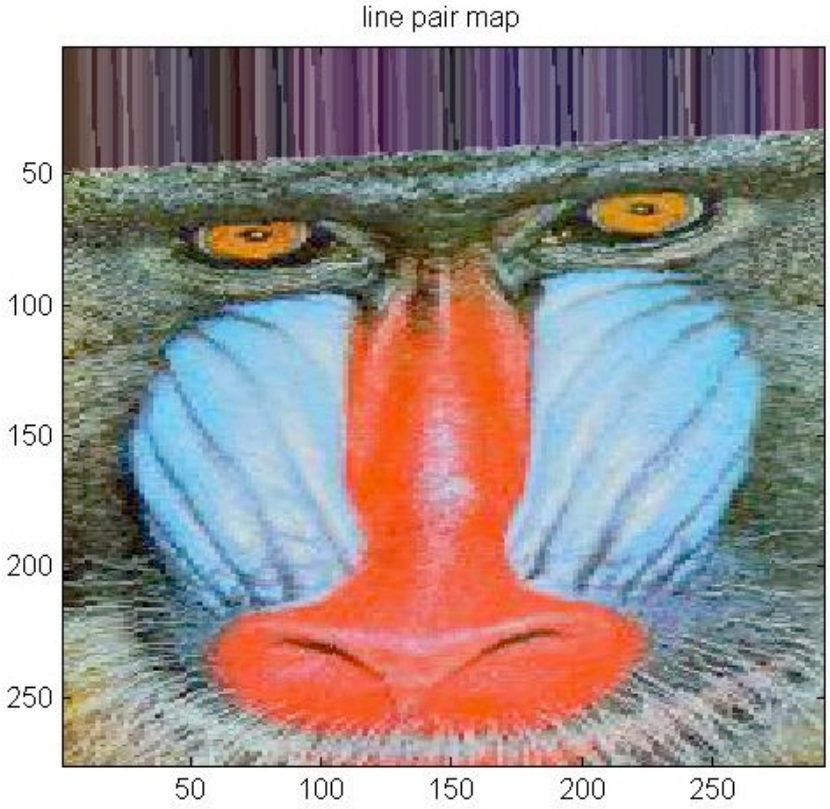


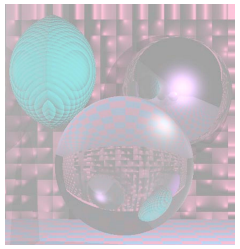
Line Pair Blend



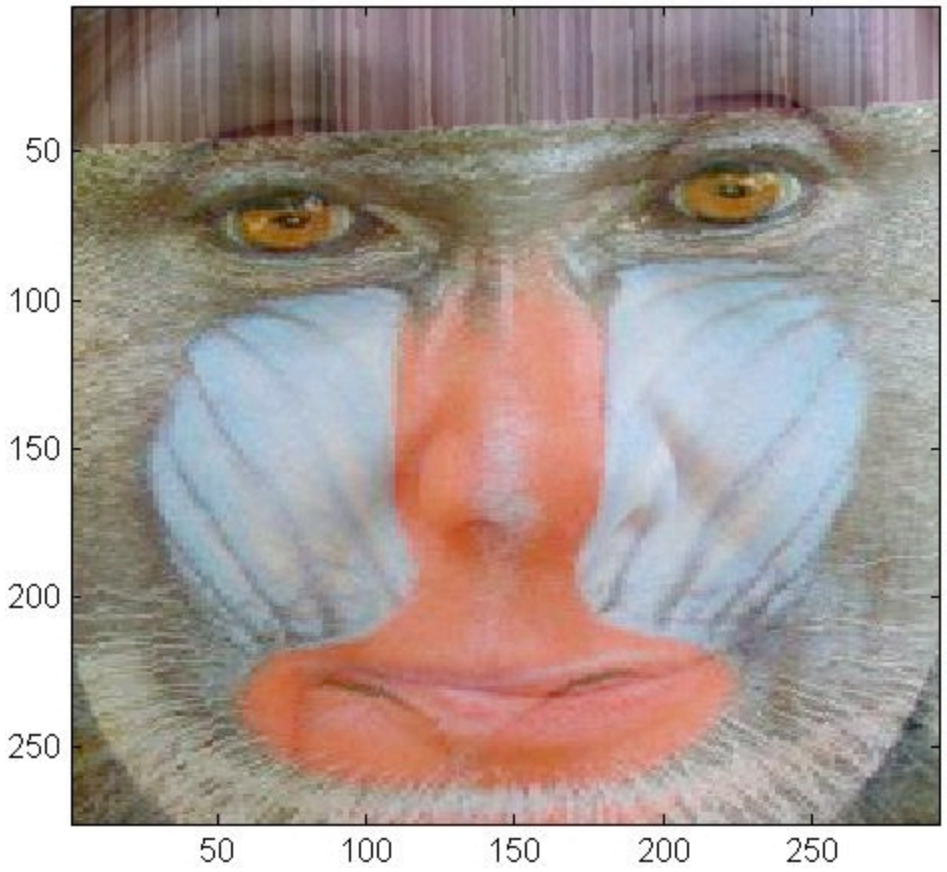
Line Pair Map 2

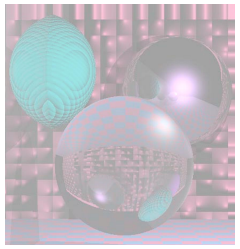




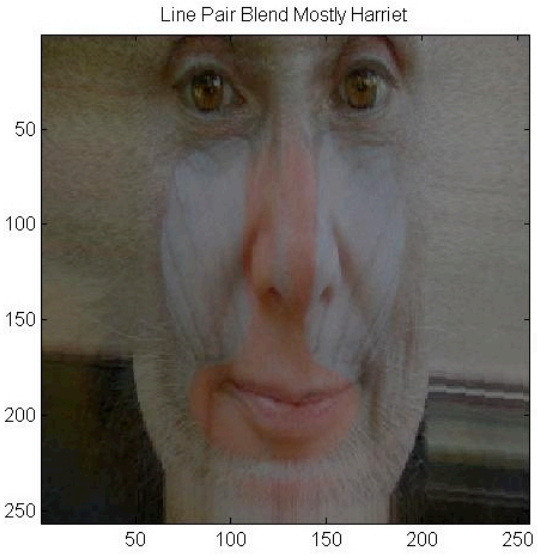


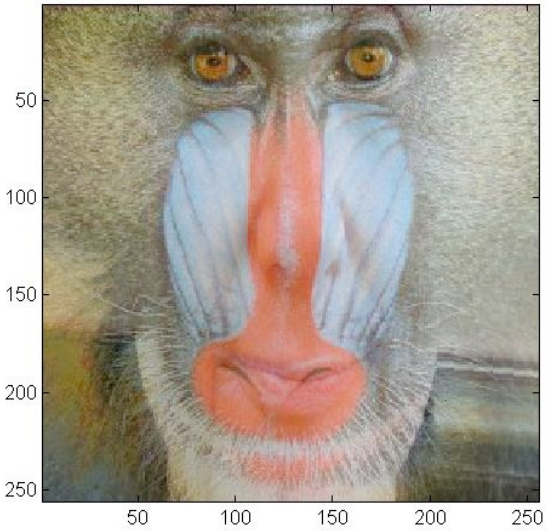
Line Pair Blend 2

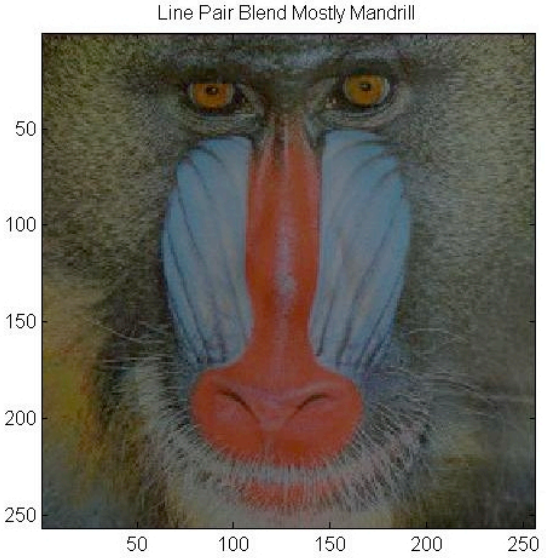


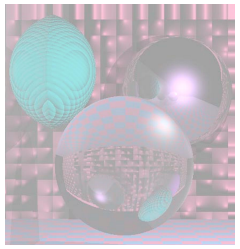


Weighted Blends









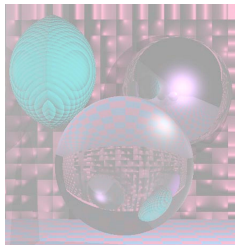
Multiple Line Pairs

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.

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

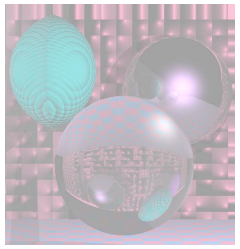
length = length of the line
dist is the distance from the pixel to the line
a, b, and p are used to change the relative effect of the lines.

Add average displacement to X to determine X‘.



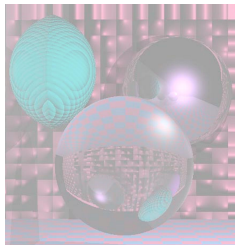
Let's Morph

FantaMorph



Mathematics of Animation

- Lerp
- Spline
  - B-Spline
- Rotate
  - Quaternions
  - Slerp



gimble

`Twas brillig, and the slithy toves

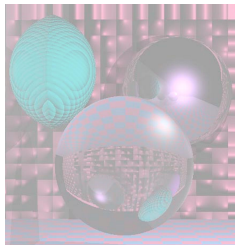
Did gyre and gimble in the wabe:

All mimsy were the borogoves,

And the mome raths outgrabe.

JABBERWOCKY

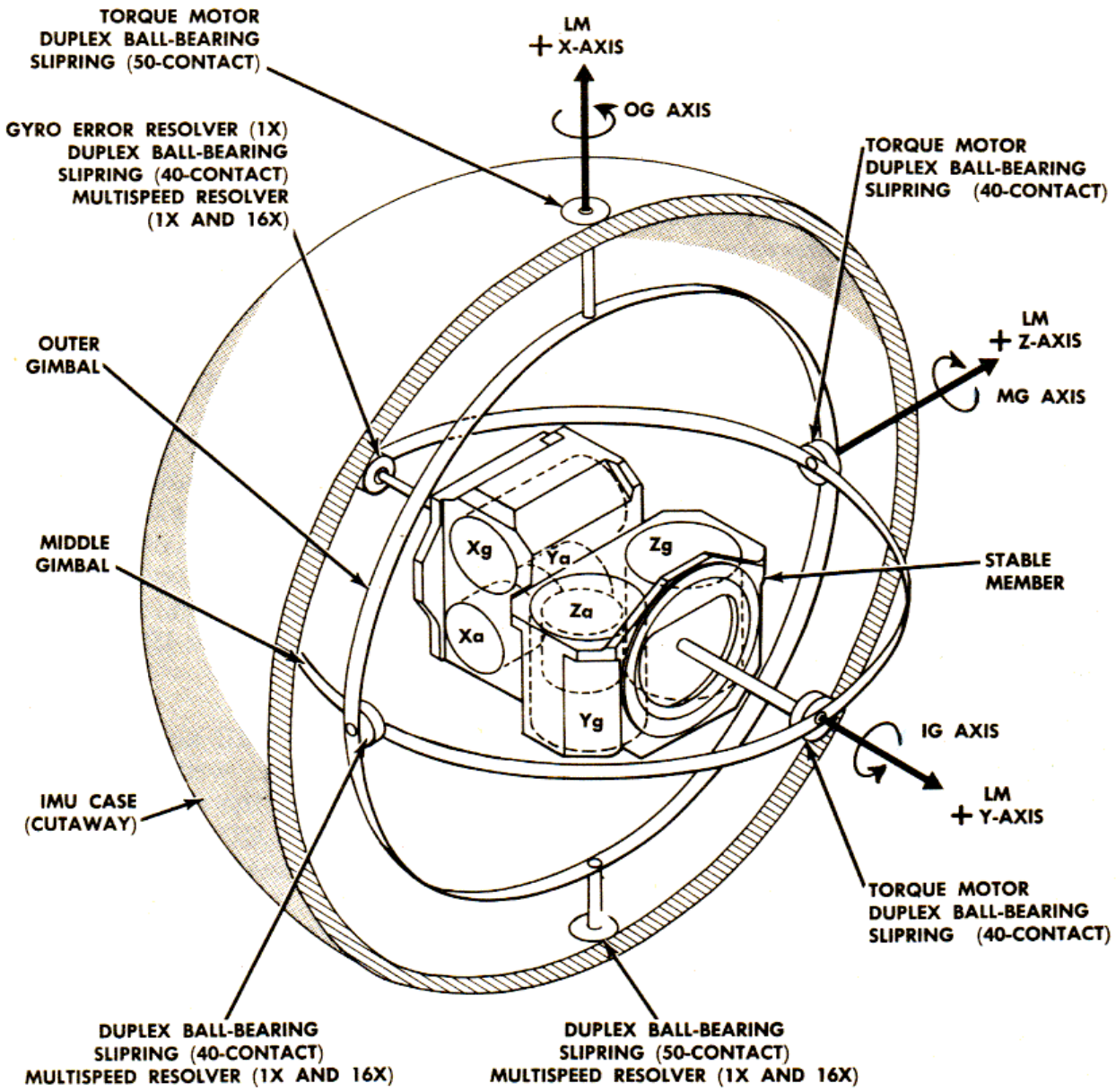
by Lewis Carroll

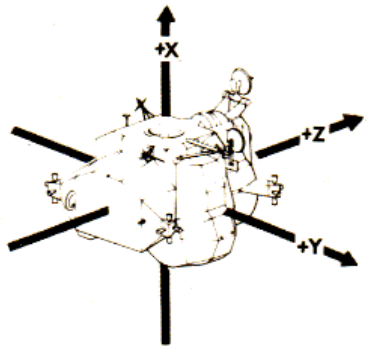


Gimbal

gim·bal
n. A device consisting of two rings mounted on axes at right angles to each other so that an object, such as a ship's compass, will remain suspended in a horizontal plane between them regardless of any motion of its support. Often used in the plural. Also called gimbal ring.

gim·baled or gim·balled, gim·bal·ing or gim·bal·ling, gim·bals
tr.v. To supply with or support on gimbals

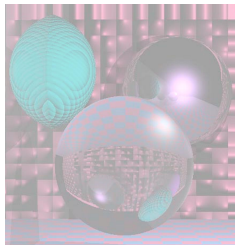




Note:
Xg = X IRIG; Xa = X PIP
Yg = Y IRIG; Ya = Y PIP
Zg = Z IRIG; Za = Z PIP

300LM4-152

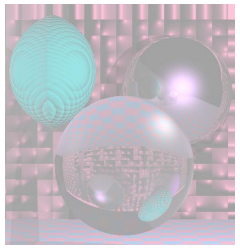
Figure 2.1-24. IMU Gimbal Assembly



Gimbal Lock

Gimbal lock in gyroscopic devices controlled by Euler mechanics or Euler angles is caused by the alignment of two of the three gimbals together so that one of the rotation references (pitch/yaw/roll, often yaw) is cancelled. This would require a reset of the gimbals using an outside reference.

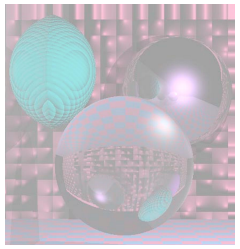
http://en.wikipedia.org/wiki/Gimbal_lock



Gimbal Lock Example

For example, an airplane uses three references, pitch (angle up/down), yaw (angle left/right on a vertical axis) and roll (angle left/right on the horizontal axis). If an airplane heads straight up or down (change of pitch), one other reference (the yaw) is cancelled, one loses a dimension of rotation, because there is always a value for one angle of rotation that yields indefinite values of one of the other two angles (in this case, the yaw).

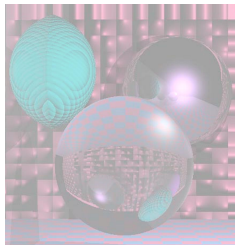
A solution to this problem is the implementation of an extra gimbal in the INS-platform. This reduces the statistical chance of gimbal lock to almost zero.



Apollo 13 Gimbal Lock

However, each gimbal also adds complexity, bulk, and weight.

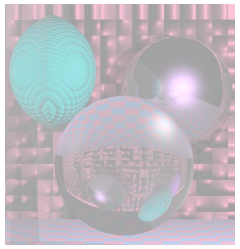
For Apollo, the decision was to use only three gimbals and rely on the astronauts to avoid sequences of rotations that would produce gimbal lock. This was normally not a problem, but became a troublesome complication when Apollo 13 had attitude-control difficulties after the tank rupture.



Gimbal Lock Continued

Another real world comparison is latitude and longitude. At the poles (latitude 90° north or south), the definition of longitude becomes meaningless (as all longitude lines meet at a point or singularity).

Another solution to Gimbal lock is the use of mathematical entities known as quaternions to represent spatial rotations. These are used most often in computer and mathematical contexts, rather than in gyroscopic devices.

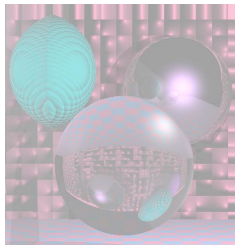


Gimbal Lock in Animation

Any system that uses Eular angles (Maya, Max, Lightwave, Softimage) will have problems with gimbal lock. The reason for this is that Eular angles evaluate each axis independently in a set order.

The order is generally X,Y,Z meaning ... first the object travels down the X axis. When that operation is complete it then travels down the Y axis, and finally the Z axis.

Gimbal lock occurs when you rotate the object down the Y axis, say 90 degrees. Since the X component has already been evaluated it doesn't get carried along with the other two axis. What winds up happening is the X and Z axis get pointed down the same axis.

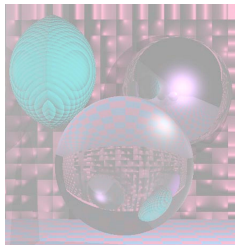


Gimbal Lock Demo

Gimble Lock – Explained

EulerExplained-Desktop

Quaternions and rotation



Quaternions

$$\mathbb{H} = \{a + bi + cj + dk \mid a,b,c,d \in \mathbb{R}\}$$

$$(a_1 + b_1 i + c_1 j + d_1 k) + (a_2 + b_2 i + c_2 j + d_2 k)$$

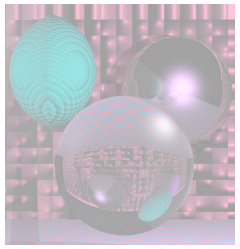
$$= (a_1 + a_2) + (b_1 + b_2)i + (c_1 + c_2)j + (d_1 + d_2)k$$

To evaluate

$$(a_1 + b_1 i + c_1 j + d_1 k)(a_2 + b_2 i + c_2 j + d_2 k)$$

use the distributive law and

$$i^2 = j^2 = k^2 = ijk = -1$$

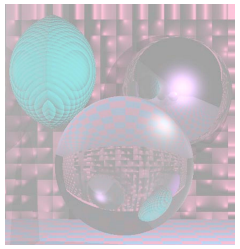


Complex Numbers as Rotations

$$e^{i\theta} = \cos\theta + i\sin\theta$$

$$v = re^{i\varphi} = r\cos\varphi + ri\sin\varphi$$

$$e^{i\theta}v = re^{i\theta}e^{i\varphi} = re^{i(\theta+\varphi)} = r\cos(\theta+\varphi) + ri\sin(\theta+\varphi)$$



Quaternion Rotations

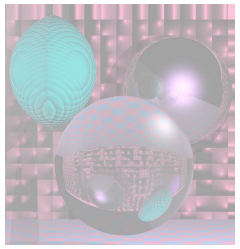
- We can represent an arbitrary vector as

$$v = (x, y, z) = xi + yj + zk$$

- The product of an arbitrary quaternion (of length 1) with a vector is not usually a vector since it has a real part.
- Multiplying the result by the conjugate of the quaternion does result in a vector

$$\Re e(pq) = \Re e(qp) \quad \text{for any quaternions } p \text{ and } q.$$

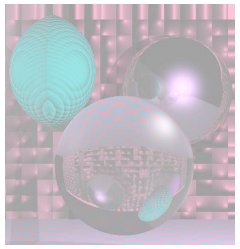
$$\Re e\left(qvq^{-1}\right) = \Re e\left(vq^{-1}q\right) = \Re e(v \cdot 1)$$



Quaternion Rotation

- A counterclockwise rotation through an angle α about an axis *v* can be represented via conjugation by the unit quaternion

$$q = \cos\frac{\alpha}{2} + \sin\frac{\alpha}{2}v$$



Quaternion Rotations

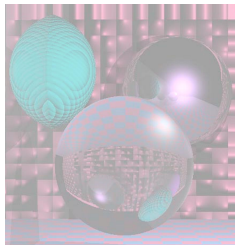
The inverse of a unit quaternion is easy to find.

If $q = a + v = a + v_1 i + v_2 j + v_3 k$ with $a^2 + \|v\| = 1$.

then $q^{-1} = q^* = a - v$.

The composition of two rotations corresponds to quaternion multiplication.

http://en.wikipedia.org/wiki/Quaternions_and_spatial_rotation

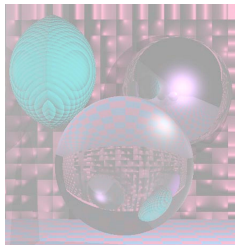


Quaternion Rotation Matrix

- The orthogonal matrix corresponding to a rotation by the unit quaternion

  $z = a + bi + cj + dk$ (with |z| = 1) is given by

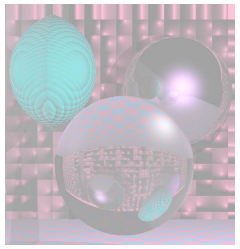
$$\begin{pmatrix} a^2+b^2-c^2-d^2 & 2bc-2ad & 2ac-2bd \\ 2ad-2bc & a^2-b^2+c^2-d^2 & 2cd-2ab \\ 2bd-2ac & 2ab+2cd & a^2-b^2-c^2+d^2 \end{pmatrix}$$



Slerp

- **Slerp** is shorthand for **spherical linear interpolation**, introduced by Ken Shoemake in the context of quaternion interpolation for the purpose of animating 3D rotation. It refers to constant speed motion along a unit radius great circle arc, given the ends and an interpolation parameter between 0 and 1.

  http://en.wikipedia.org/wiki/Slerp



Slerping

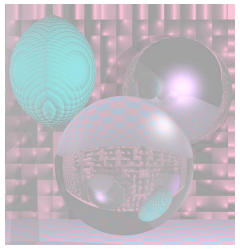
$$e^q = 1 + q + \frac{q^2}{2} + \frac{q^3}{6} + \cdots + \frac{q^n}{n!} + \cdots$$

$$q = \cos\Omega + v\sin\Omega$$

$$e^{v\Omega} = q \quad \text{and} \quad q^t = \cos t\Omega + v\sin t\Omega$$

$$Slerp(q_0, q_1, t) = q_0\left(q_0^{-1}q_1\right)^t = \left(q_1 q_0^{-1}\right)^t q_0$$

When Slerp is applied to unit quaternions, the quaternion path maps to a path through 3D rotations in a standard way. The effect is a rotation with uniform angular velocity around a fixed rotation axis.



Air on Dirac Strings

Air on Dirac Strings