Computational Medicine
Computational Anatomy
Homework 1

Seraj Grimes, Kavita Krishnan, Chantelle Lim, Joshua Popp, Tony Wei

September 2019

1 Orthogonal Transformations

1.1 2D Rotations

Consider the 2D rotation matrix

$$\phi_\theta = \begin{pmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{pmatrix}$$

We have shown that the inverse of this matrix is its transpose, and that the identity matrix corresponds to $\theta = 0$. To demonstrate that these rotations form a group, show that they are closed under matrix multiplication. That is, show that $\phi_\theta \circ \phi_{\theta'} = \phi_\gamma$ for some γ.

$$\begin{pmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{pmatrix} \begin{pmatrix} \cos(\theta') & -\sin(\theta') \\ \sin(\theta') & \cos(\theta') \end{pmatrix} =$$

$$= \begin{pmatrix} \cos(\theta)\cos(\theta') - \sin(\theta)\sin(\theta') & -\cos(\theta)\sin(\theta') - \sin(\theta)\cos(\theta') \\ \sin(\theta)\cos(\theta') + \cos(\theta)\sin(\theta') & -\sin(\theta)\sin(\theta') + \cos(\theta)\cos(\theta') \end{pmatrix}$$

$$= \begin{pmatrix} \cos(\theta+\theta') & -\sin(\theta+\theta') \\ \sin(\theta+\theta') & \cos(\theta+\theta') \end{pmatrix}$$

Note that

$$\cos(\theta)\cos(\theta') - \sin(\theta)\sin(\theta') = \cos(\theta+\theta')$$

and

$$\sin(\theta)\cos(\theta') + \sin(\theta')\cos(\theta) = \sin(\theta+\theta')$$

from trigonometric angle identities.

The set of 2D rotation matrices forms a group because (in addition to the already proven properties) we observe that for any real θ, θ' we have $\phi_\theta \circ \phi_{\theta'} = \phi_\gamma$ for $\gamma = \theta + \theta'$

1.2 3D Rotations with Euler Angles

We have parameterized the 3D rotations as 3 rotations around the x, y and z axes.

$$
\begin{aligned}
O &= \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\psi & -\sin\psi \\ 0 & \sin\psi & \cos\psi \end{pmatrix} \begin{pmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{pmatrix} \begin{pmatrix} \cos\phi & -\sin\phi & 0 \\ \sin\phi & \cos\phi & 0 \\ 0 & 0 & 1 \end{pmatrix} \\
&= \begin{pmatrix} \cos\theta\cos\phi & -\cos\theta\sin\phi & \sin\theta \\ \cos\psi\sin\phi + \sin\psi\sin\theta\cos\phi & \cos\psi\cos\phi - \sin\psi\sin\theta\sin\phi & -\sin\psi\cos\theta \\ \sin\psi\sin\phi - \cos\psi\sin\theta\cos\phi & \sin\psi\cos\phi + \cos\psi\sin\theta\sin\phi & \cos\psi\cos\theta \end{pmatrix}
\end{aligned}
$$

Show that the above transformation is orthogonal (i.e. $O^T O = Id$ for Id the identity matrix), and also show that it has determinant 1 (Hint: It will take a lot of work to show this for the full rotation matrix directly. Instead, show that the product of orthogonal matrices is also orthogonal, and show that each of the three on-axis rotations is orthogonal. Also, use the fact that the determinant of a product is the product of determinants).

First, prove that the product of orthogonal matrices is orthogonal: Let $\mathbf{M}$ and $\mathbf{N}$ be matrices such that

$$\mathbf{M}\mathbf{M}^\top = \mathbf{M}^\top\mathbf{M} = \mathbf{Id}$$

$$\mathbf{N}\mathbf{N}^\top = \mathbf{N}^\top\mathbf{N} = \mathbf{Id}$$

Then

$$(\mathbf{M}\mathbf{N})^\top(\mathbf{M}\mathbf{N}) = \mathbf{N}^\top\mathbf{M}^\top\mathbf{M}\mathbf{N} = \mathbf{N}^\top\mathbf{I}\mathbf{d}\mathbf{N} = \mathbf{Id}$$

Therefore if each matrix in O is orthogonal, so is O. Let O = ABC where

$$
\begin{aligned}
\mathbf{A} &= \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\psi & -\sin\psi \\ 0 & \sin\psi & \cos\psi \end{pmatrix} \\
\mathbf{B} &= \begin{pmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{pmatrix} \\
\mathbf{C} &= \begin{pmatrix} \cos\phi & -\sin\phi & 0 \\ \sin\phi & \cos\phi & 0 \\ 0 & 0 & 1 \end{pmatrix}
\end{aligned}
$$

Let us prove that each of A, B, and C are orthogonal.

$$
\begin{aligned}
\mathbf{A}\mathbf{A}^\top &= \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\psi & -\sin\psi \\ 0 & \sin\psi & \cos\psi \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\psi & \sin\psi \\ 0 & -\sin\psi & \cos\psi \end{pmatrix} \\
&= \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos^2\psi + \sin^2\psi & 0 \\ 0 & 0 & \sin^2\psi + \cos^2\psi \end{pmatrix} \\
&= \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}
\end{aligned}
$$

$$\begin{aligned}\mathbf{B}\mathbf{B}^\top &= \begin{pmatrix}\cos\theta & 0 & \sin\theta\\ 0 & 1 & 0\\ -\sin\theta & 0 & \cos\theta\end{pmatrix}\begin{pmatrix}\cos\theta & 0 & -\sin\theta\\ 0 & 1 & 0\\ \sin\theta & 0 & \cos\theta\end{pmatrix}\\ &= \begin{pmatrix}\cos^2\theta+\sin^2\theta & 0 & 0\\ 0 & 1 & 0\\ 0 & 0 & \sin^2\theta+\cos^2\theta\end{pmatrix}\\ &= \begin{pmatrix}1 & 0 & 0\\ 0 & 1 & 0\\ 0 & 0 & 1\end{pmatrix}\end{aligned}$$

$$\begin{aligned}\mathbf{C}\mathbf{C}^\top &= \begin{pmatrix}\cos\phi & -\sin\phi & 0\\ \sin\phi & \cos\phi & 0\\ 0 & 0 & 1\end{pmatrix}\begin{pmatrix}\cos\phi & \sin\phi & 0\\ -\sin\phi & \cos\phi & 0\\ 0 & 0 & 1\end{pmatrix}\\ &= \begin{pmatrix}\cos^2\phi+\sin^2\phi & 0 & 0\\ 0 & \sin^2\phi+\cos^2\phi & 0\\ 0 & 0 & 1\end{pmatrix}\\ &= \begin{pmatrix}1 & 0 & 0\\ 0 & 1 & 0\\ 0 & 0 & 1\end{pmatrix}\end{aligned}$$

As $O = \mathbf{ABC}$ and $\mathbf{A}$, $\mathbf{B}$, and $\mathbf{C}$ are orthogonal, matrix O is orthogonal.
To prove that det(O) = 1, we use the property that

$$det(\mathbf{ABC}) = det(\mathbf{A}) \cdot det(\mathbf{B}) \cdot det(\mathbf{C})$$

The determinants of each component matrix of O are calculated.

$$\begin{aligned}det(\mathbf{A}) &= 1 \cdot (\cos^2\psi - -\sin^2\psi) = 1\\ det(\mathbf{B}) &= \cos\theta \cdot (\cos\theta) + \sin\theta \cdot *(0 - -\sin\theta) = 1\\ det(\mathbf{C}) &= \cos\phi \cdot (\cos\phi) - -\sin\phi \cdot (\sin\phi) = 1\\ det(O) &= det(\mathbf{ABC}) = 1 \cdot 1 \cdot 1 = 1\end{aligned}$$

2 Homogeneous Coordinates

2.1 Closure under matrix multiplication

Affine transformations are represented in the following form. Let

$$A = \begin{pmatrix}a_{11} & a_{12} & a_{13} & a_{14}\\ a_{21} & a_{22} & a_{23} & a_{24}\\ a_{31} & a_{32} & a_{33} & a_{34}\\ 0 & 0 & 0 & 1\end{pmatrix}$$

and let

$$B = \begin{pmatrix} b_{11} & b_{12} & b_{13} & b_{14} \\ b_{21} & b_{22} & b_{23} & b_{24} \\ b_{31} & b_{32} & b_{33} & b_{34} \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Show that AB has the form of an affine matrix transformation.

$$AB = \begin{pmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} b_{11} & b_{12} & b_{13} & b_{14} \\ b_{21} & b_{22} & b_{23} & b_{24} \\ b_{31} & b_{32} & b_{33} & b_{34} \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$= \begin{pmatrix} a_{11}b_{11} + a_{12}b_{21} + a_{13}b_{31} & a_{11}b_{12} + ... & a_{11}b_{13} + ... & a_{11}b_{14} + a_{12}b_{24} + a_{13}b_{34} \\ a_{21}b_{11} + a_{22}b_{21} + a_{23}b_{31} & a_{11}b_{12} + ... & a_{21}b_{13} + ... & a_{21}b_{14} + a_{22}b_{24} + a_{23}b_{34} \\ a_{31}b_{11} + a_{32}b_{21} + a_{33}b_{31} & a_{11}b_{12} + ... & a_{31}b_{13} + ... & a_{31}b_{14} + a_{32}b_{24} + a_{33}b_{34} \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

As seen above, AB retains the form of an affine transformation.

2.2 Action on homogeneous coordinates

Homogeneous coordinates are represented in the following form

$$x = \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ 1 \end{pmatrix}$$

Show that Ax is has the form of a homogeneous coordinate for any A defined as above.

$$Ax = \begin{pmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ 1 \end{pmatrix}$$

$$= \begin{pmatrix} a_{11}x1 + a_{12}x2 + a_{13}x3 + a_{14} \\ a_{21}x1 + a_{22}x2 + a_{23}x3 + a_{24} \\ a_{31}x1 + a_{32}x2 + a_{33}x3 + a_{34} \\ 1 \end{pmatrix}$$

As shown above, the product Ax retains the form of a homogeneous coordinate.

3 Estimating linear transformations from landmark images

Suppose we have a 2D template landmark image x_i for $i \in \{1, ..., N\}$. We would like to transform it to match a target landmark image y_i as closely as possible by minimizing sum of square error.

$$\sum_{i=1}^{N} |y_i - Ax_i|^2$$

where A is a 2×2 matrix, and the norm is defined by $|b|^2 = b \cdot b = b^T b = \sum_i b_i b_i$. To approach this, rewrite the expression $Ax_i = X_i a$ where X_i is a 2×4 matrix and a a 4×1 column vector.

$$a = \begin{pmatrix} a_{11} \\ a_{12} \\ a_{21} \\ a_{22} \end{pmatrix}$$

$$X_i = \begin{pmatrix} x_{i1} & x_{i2} & 0 & 0 \\ 0 & 0 & x_{i1} & x_{i2} \end{pmatrix}$$

First show that $Ax = Xa$. Then you can solve the problem using a standard vector calculus approach. Show that the minimizer is $a = (\sum_i X_i^T X_i)^{-1}(\sum_i X_i^T y_i)$

Let $A = \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix}$ and $x_i = \begin{pmatrix} x_{i1} \\ x_{i2} \end{pmatrix}$

$$A \cdot x_i = \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix} \begin{pmatrix} x_{i1} \\ x_{i2} \end{pmatrix} = \begin{pmatrix} a_{11}x_{i1} + a_{12}x_{i2} \\ a_{21}x_{i1} + a_{22}x_{i2} \end{pmatrix}$$

$$X_i \cdot a = \begin{pmatrix} x_{i1} & x_{i2} & 0 & 0 \\ 0 & 0 & x_{i1} & x_{i2} \end{pmatrix} \begin{pmatrix} a_{11} \\ a_{12} \\ a_{21} \\ a_{22} \end{pmatrix} = \begin{pmatrix} a_{11}x_{i1} + a_{12}x_{i2} \\ a_{21}x_{i1} + a_{22}x_{i2} \end{pmatrix}$$

As shown above, $Ax = Xa$. Now minimize the sum of squared error:

$$\epsilon^T \epsilon = \sum_{i=1}^{N} ((y_i - X_i a)^T (y_i - X_i a)) = \sum_{i=1}^{N} (y_i^T y_i - y_i^T X_i a - a^T X_i^T y_i + a^T X_i^T X_i a)$$

Note: $y^T X_i a = a^T X_i^T y_i$ (the transpose of a scalar is the same scalar)
To minimize, take derivative and set equal to zero:

$$\frac{\partial(\epsilon^T \epsilon)}{\partial a} = 0 = \sum_{i=1}^{N} (\frac{\partial(y_i^T y_i)}{\partial a} - \frac{\partial(2a^T X_i^T y_i)}{\partial a} + \frac{\partial(a^T X_i^T X_i a)}{\partial a})$$

$$\sum_{i=1}^{N} (-2X_i^T y_i + 2X_i^T X_i a) = 2a \sum_{i=1}^{N} (X_i^T X_i) - 2 \sum_{i=1}^{N} (X_i^T y_i) = 0$$

$$a \sum_{i=1}^{N} (X_i^T X_i) = \sum_{i=1}^{N} (X_i^T y_i)$$

$$a (\sum_{i=1}^{N} (X_i^T X_i))^{-1} \sum_{i=1}^{N} (X_i^T X_i) = (\sum_{i=1}^{N} (X_i^T X_i))^{-1} \sum_{i=1}^{N} (X_i^T y_i)$$

$$a = (\sum_{i=1}^{N} (X_i^T X_i))^{-1} \sum_{i=1}^{N} (X_i^T y_i)$$

4 Transforming points and images

In this homework you will calculate a transformation that matches one set of landmarks to another, and apply it to transform an image according to the nonlinear observer equation. The assignment should be performed in matlab. Please hand in any code you write, in addition to anything else the problems ask for.

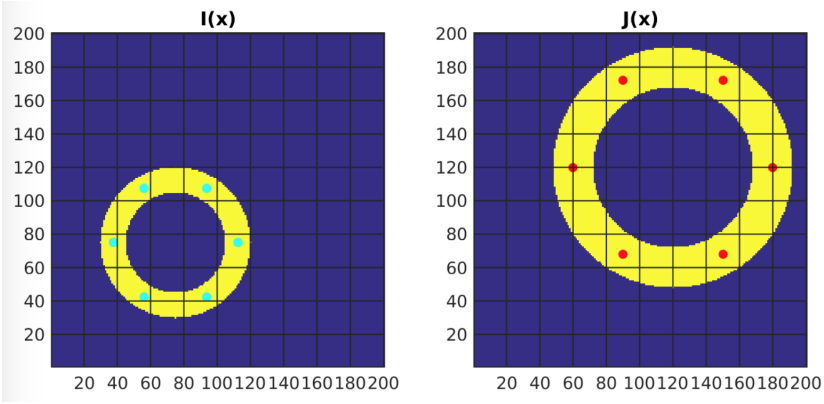


Figure 1: Images of two annuli with 6 corresponding labelled landmarks.

4.1 Generate the image $I(x)$

Use this code to generate your image of an annulus, corresponding to the left side of Fig. 1, in matlab.

```
% The location of each pixel
nX = 200; % number of columns
nY = 200; % number of rows
xj = 1 : nX; % x location of each column
yi = 1 : nY; % y location of each row
[xij,yij] = meshgrid(xj,yi); % x,y location of each pixel
% define an image of an annulus
cx = 75; % x component of center
cy = 75; % y component of center
r1 = 30; % inner radius in pixels
r2 = 45; % outer radius in pixels
% define the image using binary operations
I=((xij - cx).^2 + (yij - cy).^2 <= r2^2 )-((xij - cx).^2 + (yij - cy).^2 < r1^2 );
```

Use this code or something similar to display your image.

```
% display it
figure;
imagesc(I);
axis image;
```

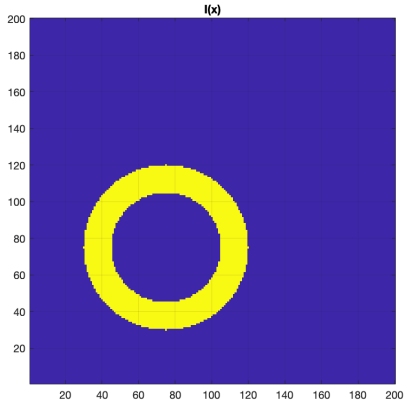
```
title('I(x)');
set(gca,'ydir','normal'); % put origin at bottom left
```

Our version of the code:

```
% 4.1 DISPLAY THE ANNULUS
% The location of each pixel
nX = 200; % number of columns
nY = 200; % number of rows
xj = 1 : nX; % x location of each column
yi = 1 : nY; % y location of each row
[xij,yij] = meshgrid(xj,yi); % x,y location of each pixel
% define an image of an annulus
cx = 75; % x component of center
cy = 75; % y component of center
r1 = 30; % inner radius in pixels
r2 = 45; % outer radius in pixels
% define the image using binary operations
I=((xij - cx).^2 + (yij - cy).^2 <= r2^2 )-((xij - cx).^2 + (yij - cy).^2 < r1^2 );

% display it
figure;
imagesc(I);
axis image;
title("I(x)");
grid on
set(gca,"ydir","normal"); % put origin at bottom left
hold on;
```

Our version of the annulus:



4.2 Record landmarks

Estimate the position of the 6 landmarks on I, and store them in the (2 rows by 6 columns) variable `X`. Estimate the position of the 6 corresponding landmarks on J, and store them in the (2 rows by 6 columns) variable `Y`. Let the rightmost landmark be the first in your list, and proceed adding the others counterclockwise. Write down the values you use and turn them in with your assignment.

X	Y
(111, 68)	(180, 120)
(92, 107)	(150, 172)
(56, 107)	(90, 172)
(39, 68)	(60, 120)
(56, 41)	(90, 63)
(92, 41)	(150, 63)

4.3 Find scale factor

Find the scale factor relating one to the other finding the value of s that minimizes the error

$$E = \sum_{i=1}^{6} \sum_{j=1}^{2} |sX_j(i) - Y_j(i)|^2$$

We can set the derivative with respect to s equal to zero and solve:

$$
\begin{aligned}
E &= \sum_{i=1}^{6}\sum_{j=1}^{2} |sX_j(i) - Y_j(i)|^2 \\
\frac{\partial E}{\partial s} &= \frac{\partial}{\partial s}\sum_{i=1}^{6}\sum_{j=1}^{2} |sX_j(i) - Y_j(i)|^2 \\
&= \sum_{i=1}^{6}\sum_{j=1}^{2} \frac{\partial}{\partial s}|sX_j(i) - Y_j(i)|^2 \\
&= \sum_{i=1}^{6}\sum_{j=1}^{2} 2|sX_j(i) - Y_j(i)|(X_j(i)) \\
0 &= \sum_{i=1}^{6}\sum_{j=1}^{2} 2|sX_j(i) - Y_j(i)|(X_j(i)) \\
0 &= \sum_{i=1}^{6}\sum_{j=1}^{2} |sX_j(i) - Y_j(i)| \\
\sum_{i=1}^{6}\sum_{j=1}^{2} |sX_j(i)| &= \sum_{i=1}^{6}\sum_{j=1}^{2} |Y_j(i)| \\
s\sum_{i=1}^{6}\sum_{j=1}^{2} |X_j(i)| &= \sum_{i=1}^{6}\sum_{j=1}^{2} |Y_j(i)| \\
s &= \frac{\sum_{i=1}^{6}\sum_{j=1}^{2} |Y_j(i)|}{\sum_{i=1}^{6}\sum_{j=1}^{2} |X_j(i)|}
\end{aligned}
$$

(From Matlab, based on values defined in 4.2)

$$s = 1.6287$$

4.4 Transform the landmarks X

Apply your scale factor to the landmarks in `X`, call this variable `sX`. Use a scatter plot to display the landmarks `X` (cyan, 'c'), the landmarks `sX` (blue, 'b'), and the landmarks `Y` (red, 'r').

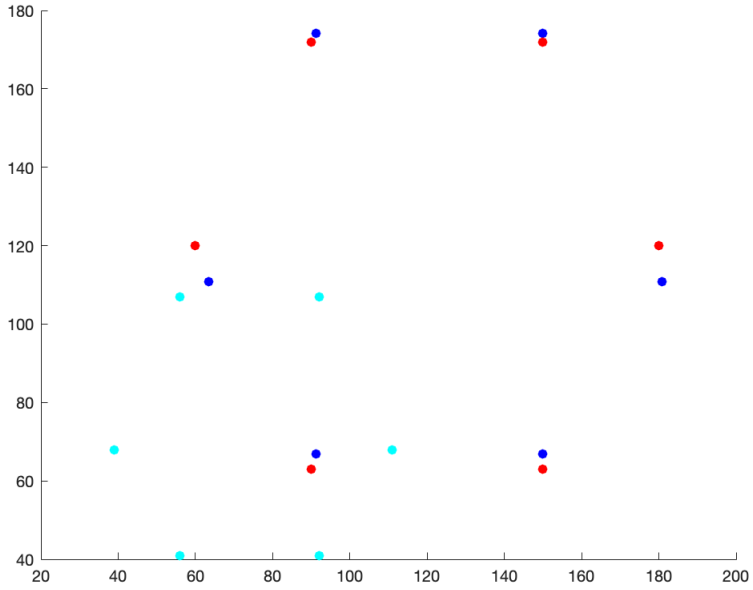


Figure 2: Images transformed

Comment on the proximity of `sX` and `Y`. Do they match exactly? Would you expect them to? Write down the transformed landmarks and turn them in with your assignment.

They are very close, and you would not expect them to be identical: the expected scale factor was calculated based on points defined by visual inspection, which is prone to error.

sX
(180.785876993166, 110.751708428246)
(149.840546697039, 174.271070615034)
(91.2072892938497, 174.271070615034)
(63.5193621867882, 110.751708428246)
(91.2072892938497, 66.7767653758542)
(149.840546697039, 66.7767653758542)

4.5 Transform the image "naively" $I(x) \rightarrow I(sx)$

Transform your image in the variable I according to the incorrect equation $I(x) \rightarrow I(sx)$ using the following pseudocode. You can reuse this code for a variety of different transformations, just change the lines that say `% you should implement this line` to match your transformation.

```
% initialize an image of all zeros
ITransformed = zeros(size(I));
for i = 1 : nY % loop through each row
```

```
for j = 1 : nX % loop through each column
% we are looking for the value to assign to Isx(j,i)
% find the position to look at in the image J
% iLook = % you should implement this line
% jLook = % you should implement this line
% round them to the nearest integer
iLookRound = round(iLook);
jLookRound = round(jLook);
% check if we're out of bounds,
if iLookRound < 1 || iLookRound > nY || jLookRound < 1 || jLookRound > nX
% if so, fill the image with the value zero
ITransformed(j,i) = 0;
else
% otherwise, assign the value in our image at this point
ITransformed(j,i) = I(jLookRound,iLookRound);
end
% don't forget to index your images by (row,column) and not (x,y) !
end
end
```

Our code:

```
% 4.5 NAIVE TRANSFORMATION
% initialize an image of all zeros
ITransformed = zeros(size(I));
for i = 2 : nY % loop through each row
    for j = 2 : nX % loop through each column
        % we are looking for the value to assign to Isx(j,i)
        % find the position to look at in the image J
        iLook = i*s;  % the inverse
        jLook = j*s;
        % round them to the nearest integer
        iLookRound = round(iLook);
        jLookRound = round(jLook);
        % check if we're out of bounds,
        if iLookRound < 1 || iLookRound > nY || jLookRound < 1 || jLookRound > nX
            % if so, fill the image with the value zero
            ITransformed(j,i) = 0;
            else
            % otherwise, assign the value in our image at this point
            ITransformed(j,i) = I(jLookRound,iLookRound);
        end
        % don't forget to index your images by (row,column) and not (x,y) !
    end
end

% display the new guy
```

```
figure;
imagesc(ITransformed);
axis image;
title("I'(x)");
grid on
set(gca,"ydir","normal"); % put origin at bottom left
```

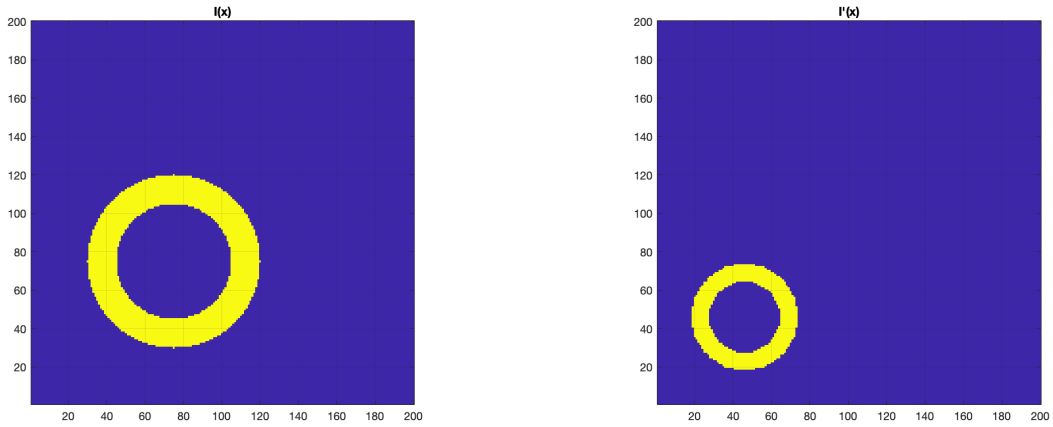


Figure 3: Image transformed via naive transformation

Comment on what happened to the image. Did it transform to match the image J?

Nope, we forgot to use the inverse, so it had the opposite effect from what you'd expect.

Print out a figure showing this transformed image and turn it in with your assignment

4.6 Transform the image with the observer equation $I(x) \rightarrow I(s^{-1}x)$

Transform your image in the variable I according to the nonlinear observer equation $I(x) \rightarrow I(s^{-1}x)$ using the above pseudocode.

Our code:

```
% 4.6 OBSERVER EQN TRANSFORMATION
% initialize an image of all zeros
ITransformed = zeros(size(I));
for i = 2 : nY % loop through each row
    for j = 2 : nX % loop through each column
        % we are looking for the value to assign to Isx(j,i)
        % find the position to look at in the image J
        iLook = i/s;  % the inverse
```

```
        jLook = j/s;
        % round them to the nearest integer
        iLookRound = round(iLook);
        jLookRound = round(jLook);
        % check if we?re out of bounds,
        if iLookRound < 1 || iLookRound > nY || jLookRound < 1 || jLookRound > nX
            % if so, fill the image with the value zero
            ITransformed(j,i) = 0;
            else
            % otherwise, assign the value in our image at this point
            ITransformed(j,i) = I(jLookRound,iLookRound);
        end
        % don't forget to index your images by (row,column) and not (x,y) !
    end
end

% display the new guy
figure;
imagesc(ITransformed);
axis image;
title("I'(x)");
grid on
set(gca,"ydir","normal"); % put origin at bottom left

hold on;
scatter(Y(:,1),Y(:,2),'r', 'filled');
```

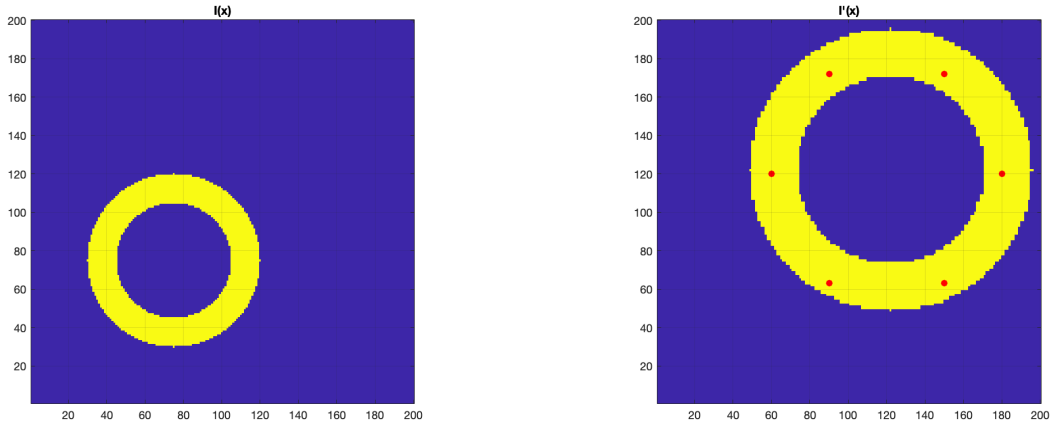


Figure 4: Image transformed via the observer equation

Comment on what happened to the image? Did it transform to match the image J?

Yes it did. Pictured is the transformed image, along with (my estimations of) the original landmark points, shown in red.

Print out a figure showing this transformed image and turn it in with your assignment.

5 Working with rigid transformations

In this homework you will gain familiarity working with rigid transformations, by manually aligning surfaces contouring the medial temporal lobe structures hippocampus and amygdala.

5.1 Open manual alignment tool

In Matlab, run the function `align_v2` provided in the m file `align_v2.m`. A window like that shown in Fig 2. The top left allows you to load files, and will display a rigid transformation matrix later.

5.2 Load atlas

Click on `load atlas`, in the `Files of Type`: dropbox select `Surfaces (*.byu)` and open the file `hippocampus_and_amygdala_2.byu` provided. This is shown in Fig. 5. The top right shows a 2D view of your 3D data seen in the XZ plane. The bottom right shows a 2D view of your 3D data seen in the XY plane. The bottom left shows a 2D view of your 3D data seen in the YZ plane. Each of these three panels has buttons to move the atlas up (U), down(D), left (L), right (R), clockwise (CW), or counterclockwise (CCW) with respect to the view in that panel. Note that this will rotate the atlas about its center, not about the origin (0,0,0).

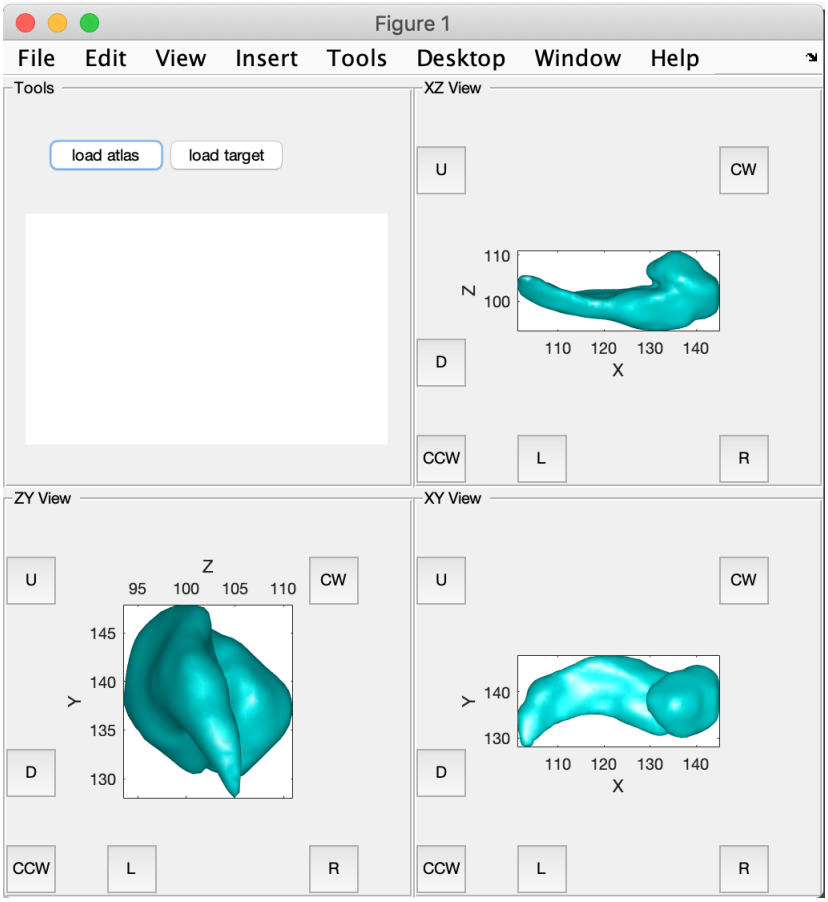


Figure 5: The atlas surface is loaded and shown in cyan.

5.3 Load target

Click on `load atlas`, in the `Files of Type`: dropbox select `Surfaces (*.byu)` and open the file `hippocampus_and_amygdala` provided. This is shown in Fig. 4

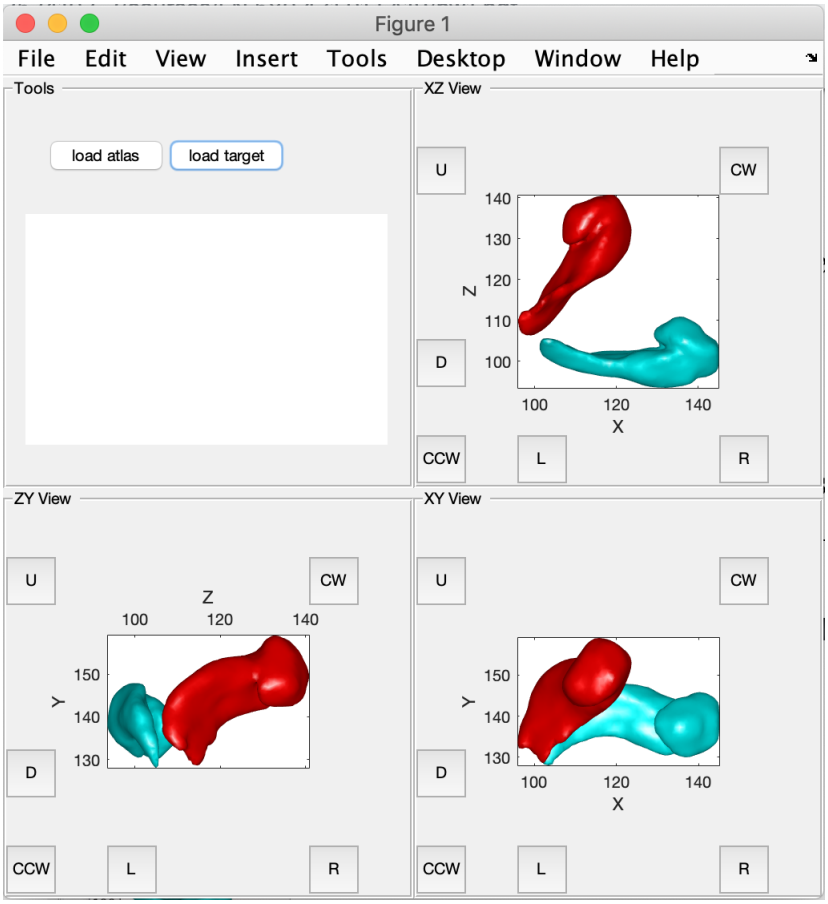


Figure 6: The target surface is loaded and shown in red.

5.4 Note corresponding features

Note the amygdala is a large bulge at near the head of the hippocampus (large x), and the hippocampus has a tapering tail (small x). Notice in the XY view that the hippocampus is concave on its medial margin (small y), and convex on its lateral margin (large y). Note that the amygdala is slightly superior to the hippocamus (larger z).

5.5 Calculate an alignment

Click on the tools in each panel to move the cyan atlas in register with the target. As you work, the linear transformation applied to the atlas will be printed in the top left panel as shown in Fig. 5. This is written as a 4 by 4 matrix with the rotation part as the upper left 3 by 3 matrix, and the translation part the right 3 by 1 column vector.

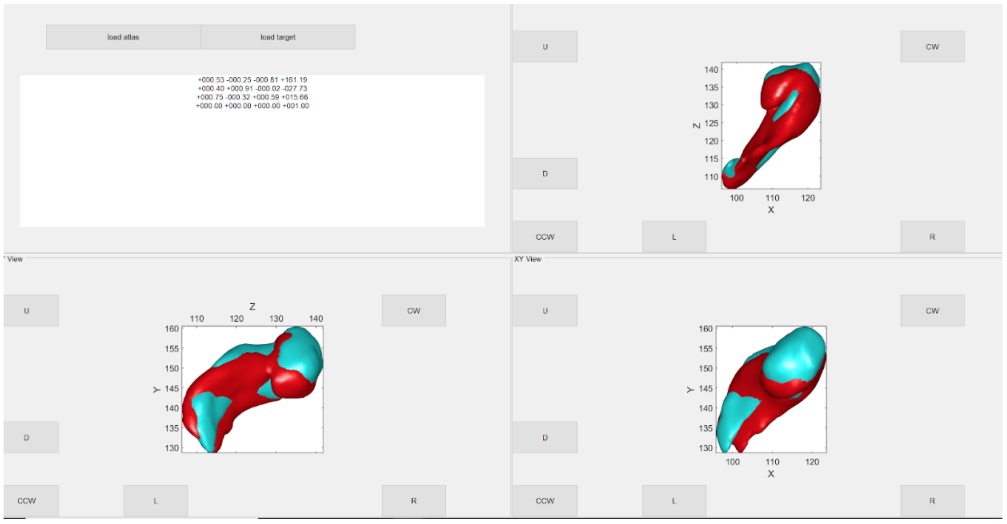


Figure 7: Linear rigid transformation matrix appears at top left.

5.6 Record your transformation

After you are satisfied with your work, record the transformation matrix printed in the top left panel, take a screenshot of the alignment, and submit it.

$$\begin{pmatrix} +000.53 & -000.25 & -000.81 & +161.19 \\ +000.40 & +000.91 & -000.02 & -027.73 \\ +000.75 & -000.32 & +000.59 & +015.66 \\ +000.00 & +000.00 & +000.00 & +001.00 \end{pmatrix}$$