CS 217 Homework 2

Zachary DeStefano, 15247592

Due Date: May 12, 2015

(a) Left Image (b) Right Image

Figure 1: Original Pictures of Stuffed Squirtle

Problem 1

The following are examples of running the SIFT matching code on my own objects

Squirtle Example

Figure 1 shows the original pictures that I took of a stuffed Pokemon named Squirtle. Figure 2 show 50 random points found by running SIFT. Figure 3 shows some of the matches found using SIFT.

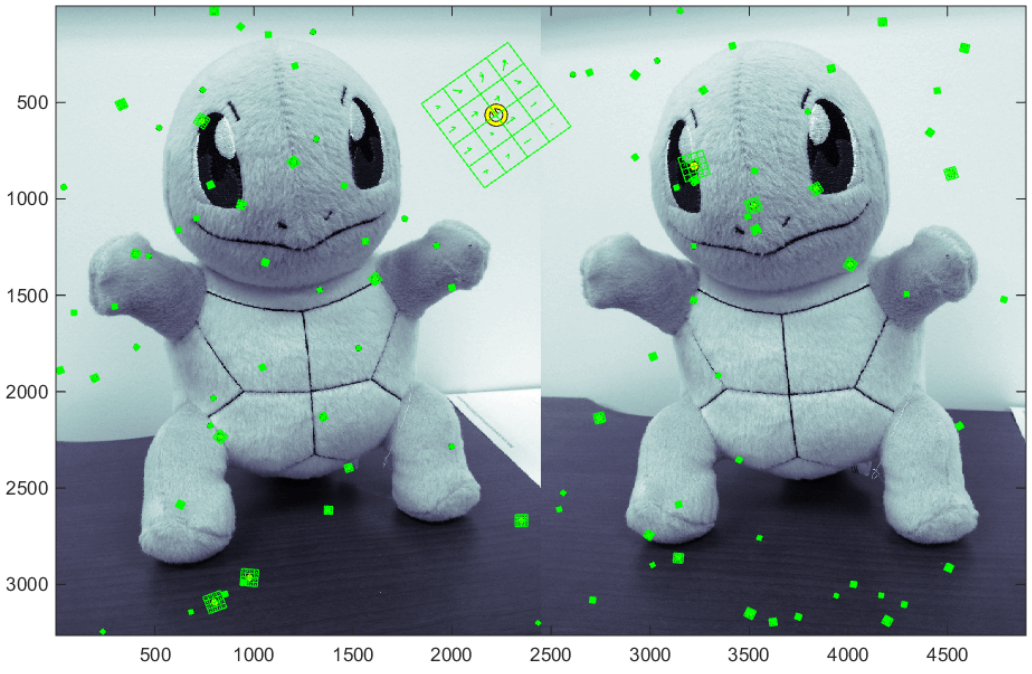


Figure 2: Squirtle Images With Points Marked

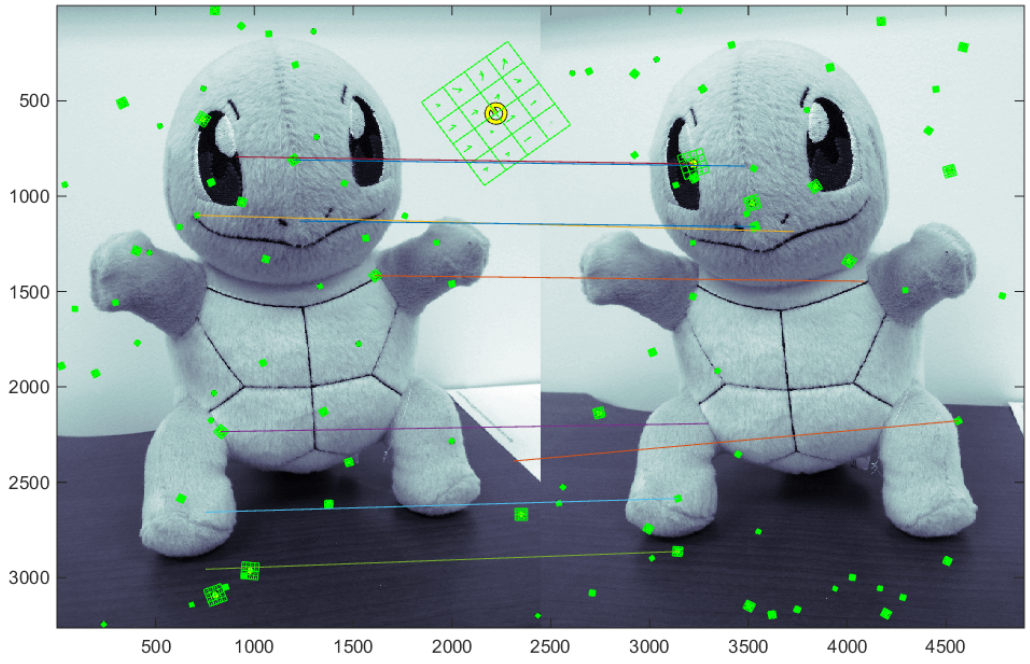
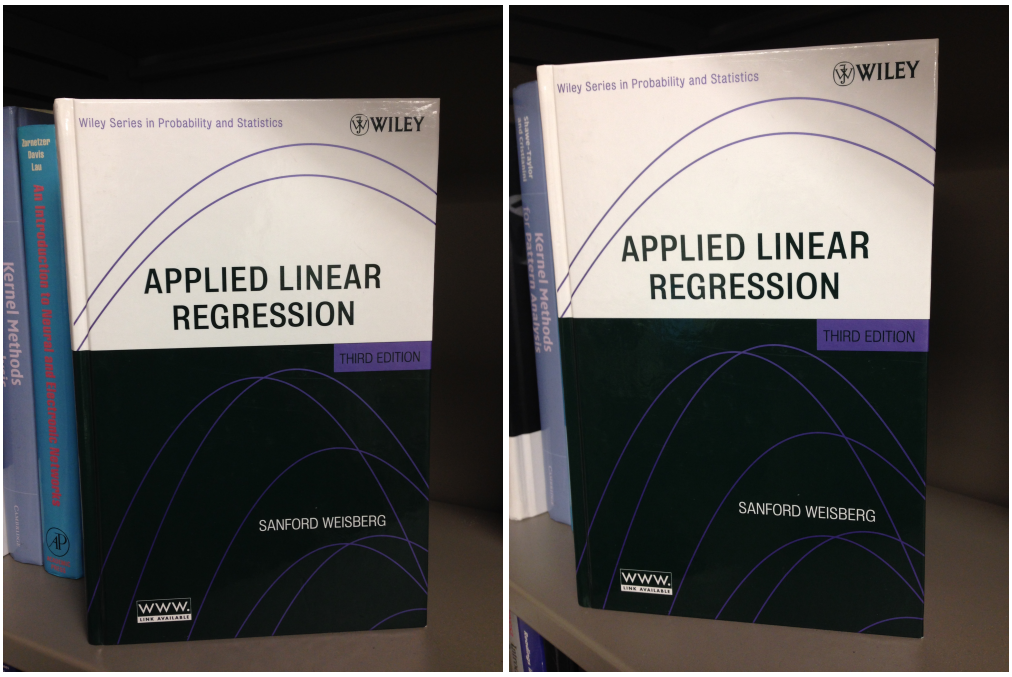


Figure 3: Squirtle Images With Points Marked and Some Matches Shown



(a) Left Image (b) Right Image

Figure 4: Original Pictures of Textbook

Book Example

Figure 4 shows the original pictures that I took of a textbook. Figure 5 show 50 random points found by running SIFT. Figure 6 shows some of the matches found using SIFT.

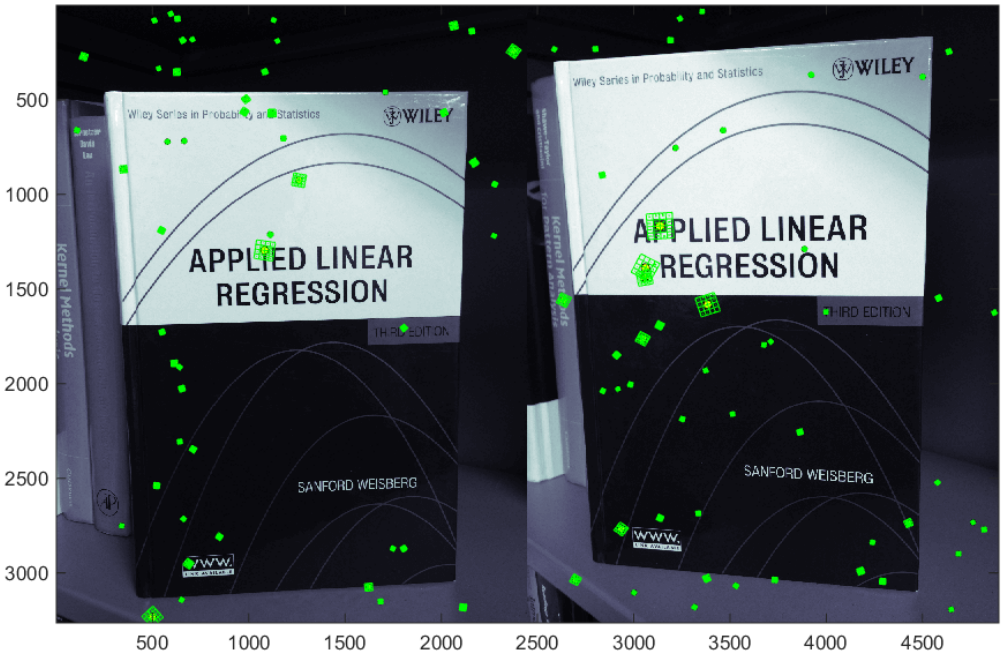


Figure 5: Textbook Images With Points Marked

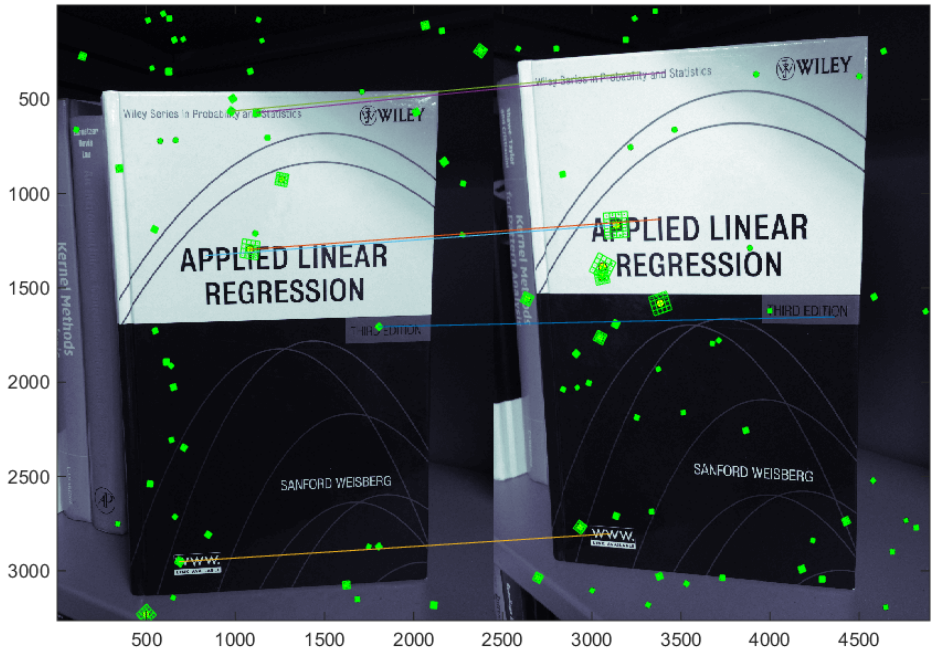


Figure 6: Textbook Images With Points Marked and Some Matches Shown

(a) Left Image (b) Right Image

Figure 7: Original Pictures of Coffee Can

Coffee Can Example

Figure 7 shows the original pictures that I took of a coffee can. Figure 8 show 50 random points found by running SIFT. Figure 9 shows some of the matches found using SIFT.

Figure 8: Coffee Can Images With Points Marked

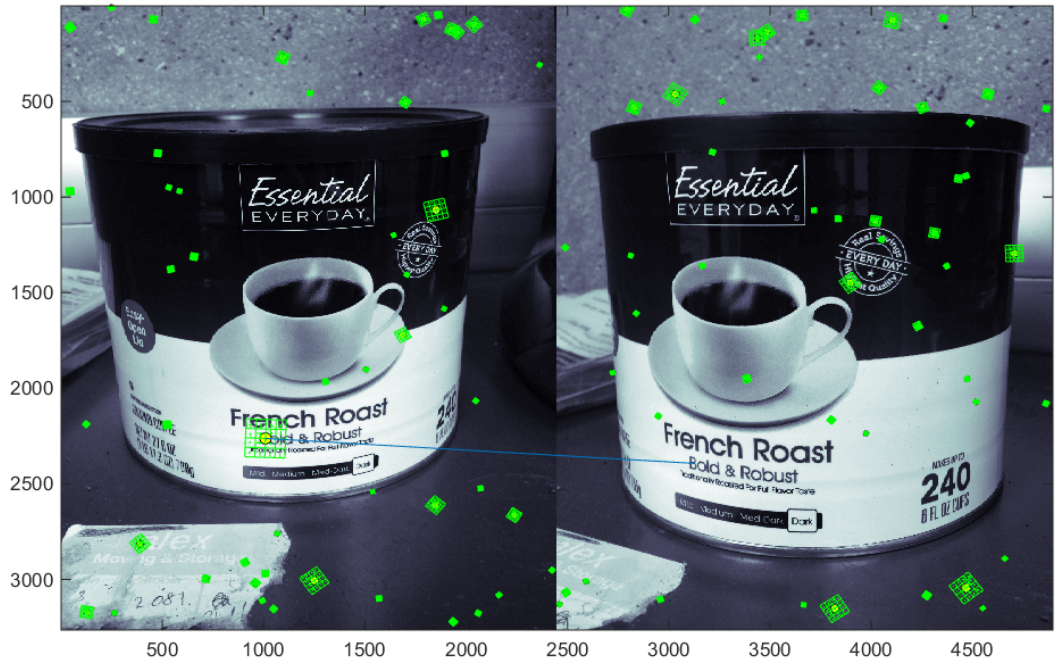


Figure 9: Coffee Can Images With Points Marked and Some Matches Shown

Code for Problem 1

Here is the code I used for this problem

```
%{
This is the script used for problem 1.

Much of the code was inspired by the tutorial here:
http://www.vlfeat.org/overview/sift.html
%}

%specify the image in the folder
%imageName = 'squirtle';
%imageName = 'book';
imageName = 'coffeeCan';
image1 = strcat(imageName,'1.JPG');
image2 = strcat(imageName,'2.JPG');

%run SIFT on image1 and image2
I1 = imread(image1);
I1 = single(rgb2gray(I1));
[f1,d1] = vl_sift(I1);
I2 = imread(image2);
I2 = single(rgb2gray(I2));
[f2,d2] = vl_sift(I2);

%select random subsets fo the points
perm = randperm(size(f1,2)) ;
sel1 = perm(1:50) ;
perm = randperm(size(f2,2)) ;
sel2 = perm(1:50) ;

%does basic matching
[matches, scores] = vl_ubcmatch(d1, d2) ;

%goes through the points visualized in sel1 and sel2
%   and finds their corresponding matches in the other images
image1matches = matches(1,:);
image2matches = matches(2,:);
numPoints = size(sel1,2);
image1Points = [];
image2Points = [];
```

```
for i = 1:numPoints
    matchingCol = find(image1matches==sel1(i));
    if(~isempty(matchingCol))
       colNum = matchingCol(1);
       image1index = matches(1,colNum);
       image1Points = [image1Points f1(1:2,image1index)];
       img2MatchingIndex = matches(2,colNum);
       image2Points = [image2Points f2(1:2,img2MatchingIndex)];
    end
end

numPoints = size(sel2,2);
for i = 1:numPoints
    matchingCol = find(image2matches==sel2(i));
    if(~isempty(matchingCol))
       colNum = matchingCol(1);
       image2index = matches(2,colNum);
       image2Points = [image2Points f2(1:2,image2index)];
       img1MatchingIndex = matches(1,colNum);
       image1Points = [image1Points f1(1:2,img1MatchingIndex)];
    end
end

%plots the left and right image side-by-side
doubleImage = [I1 I2];
figure
imagesc(doubleImage);
colormap bone;

%plots the SIFT points for left image
h1 = vl_plotframe(f1(:,sel1)) ;
h2 = vl_plotframe(f1(:,sel1)) ;
set(h1,'color','k','linewidth',3) ;
set(h2,'color','y','linewidth',2) ;
h3 = vl_plotsiftdescriptor(d1(:,sel1),f1(:,sel1)) ;
set(h3,'color','g') ;

%plots the SIFT points for the right image
width = size(I1,2);
f2Original = f2(:,sel2);
f2Offset = repmat([width;0;0;0],1,50);
```

```
f2New = f2Original+f2Offset;
h1 = vl_plotframe(f2New) ;
h2 = vl_plotframe(f2New) ;
set(h1,'color','k','linewidth',3) ;
set(h2,'color','y','linewidth',2) ;
h3 = vl_plotsiftdescriptor(d2(:,sel2),f2New) ;
set(h3,'color','g') ;

%plots the matches for the visualized points in each image
hold on
for i = 1:size(image1Points,2)
    Xvals = [image1Points(1,i) image2Points(1,i)+width];
    Yvals = [image1Points(2,i) image2Points(2,i)];
    plot(Xvals,Yvals);
end
```

Problem 2

Part A

For problem 2, I used the same images as the previous problem but I chose my own points. Figures 10, 11, and 12 show the points that I chose in each image. After running my own implementation of RANSAC, I got the correspondences that are shown in figures 13, 14, and 15.

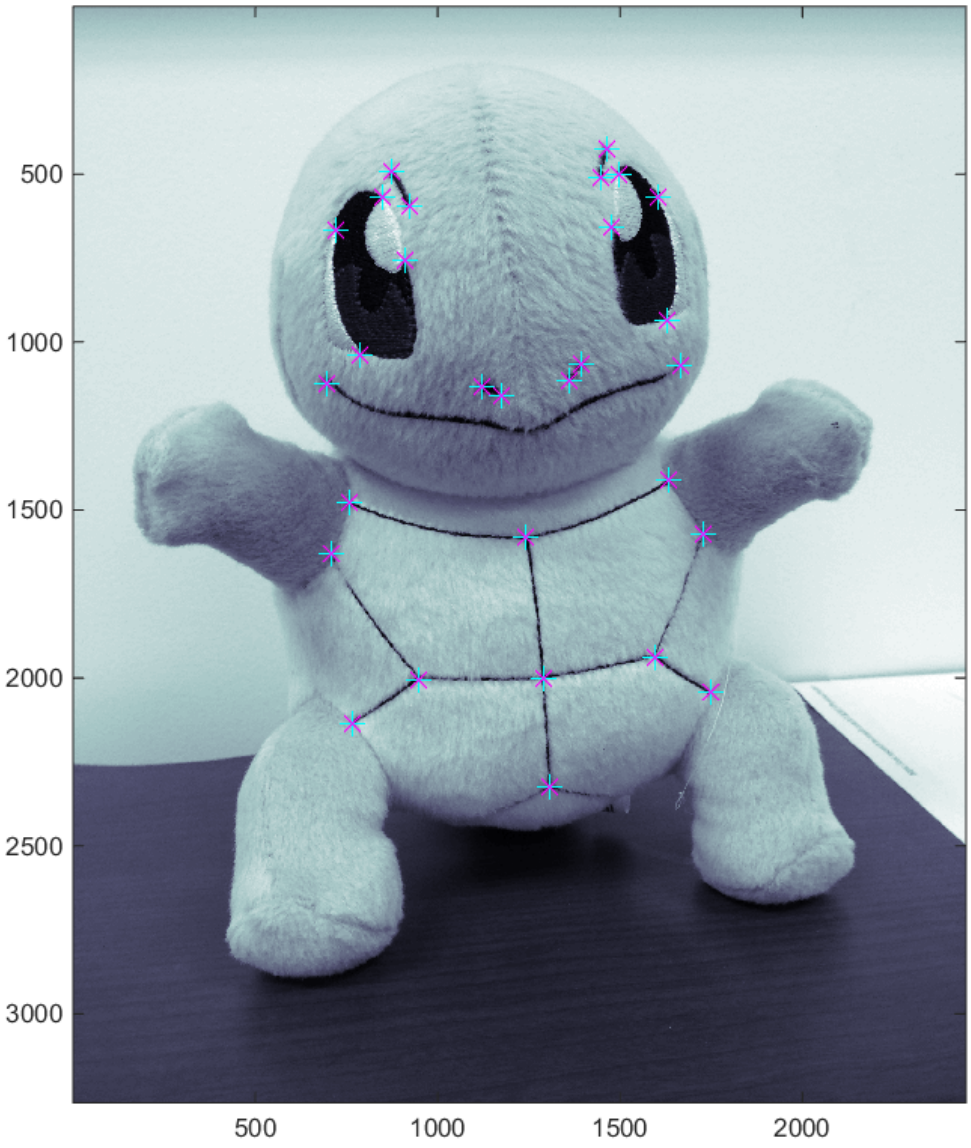


Figure 10: Squirtle Image with Points Marked

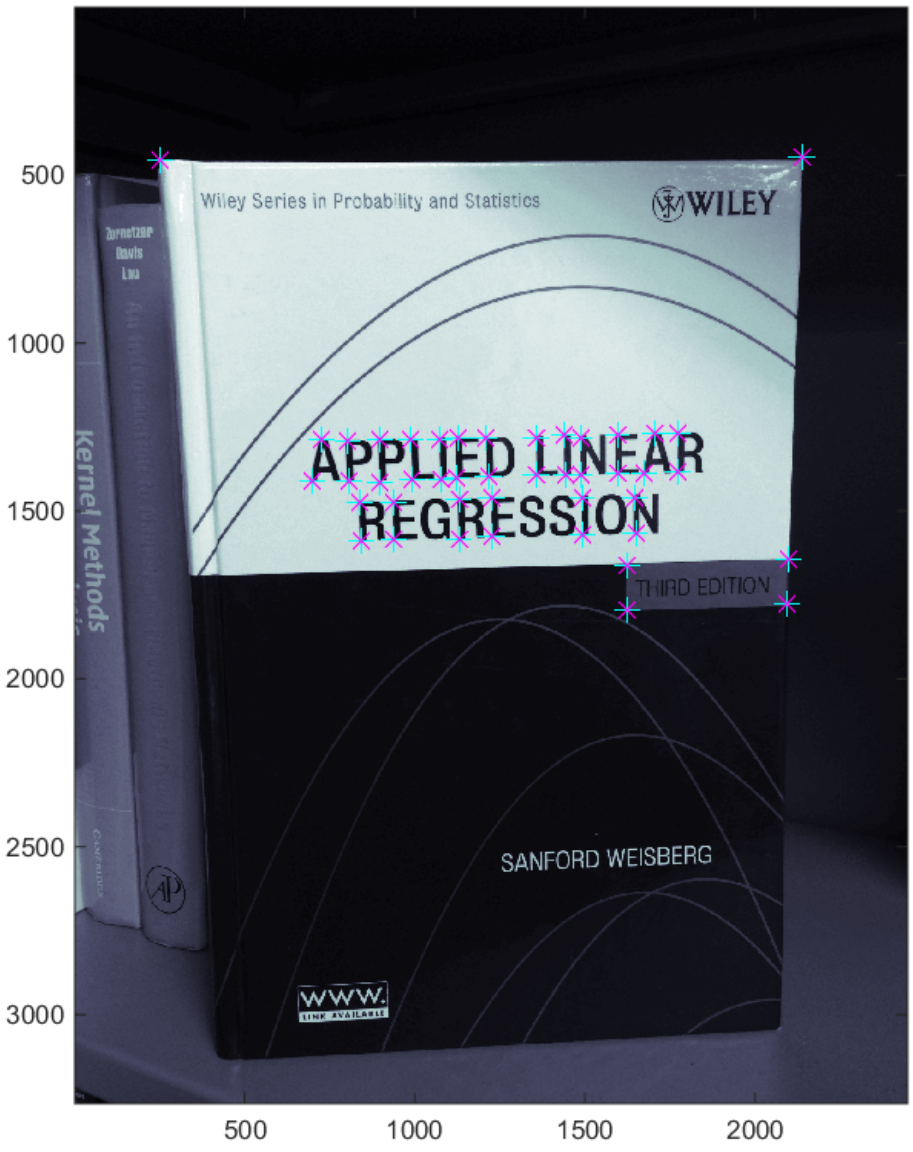


Figure 11: Textbook Image with Points Marked

Figure 12: Coffee Can Image with Points Marked

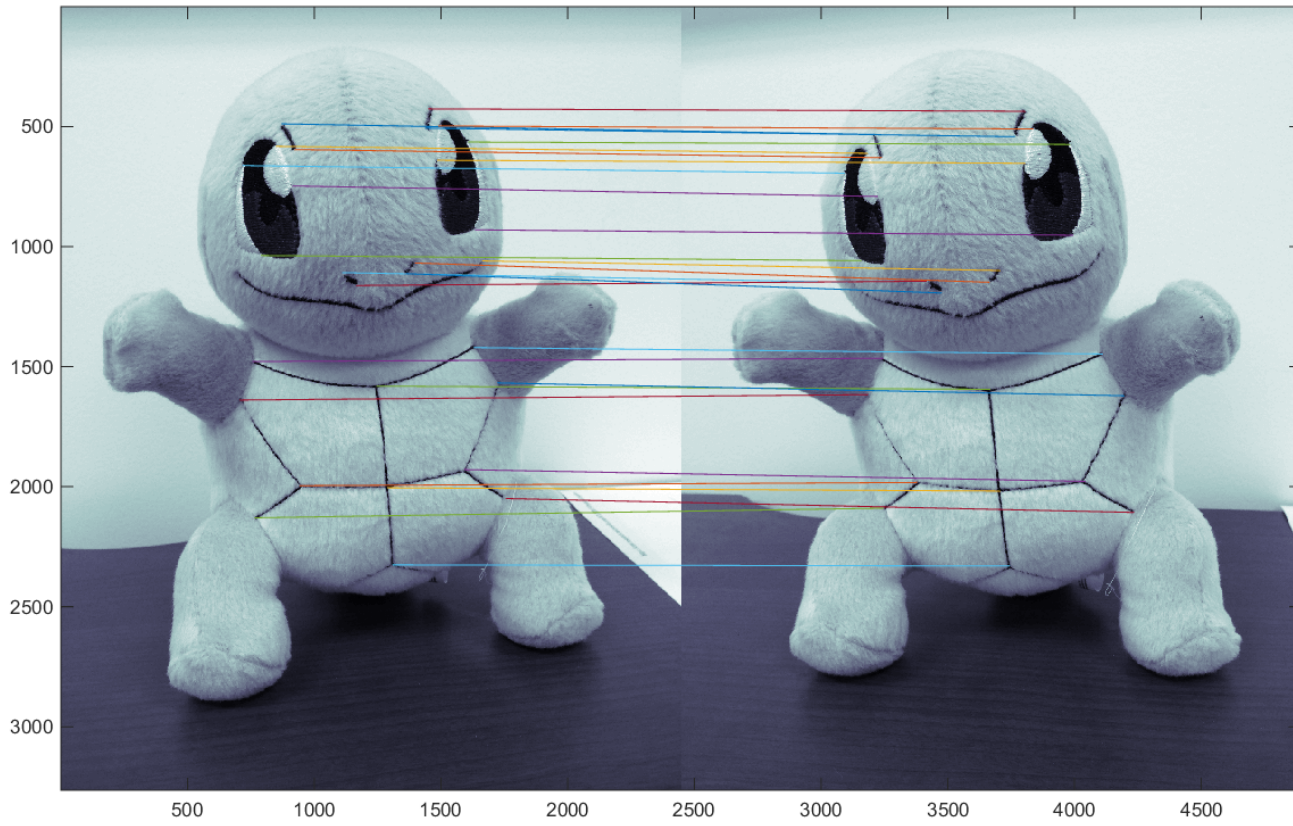


Figure 13: Squirtle Images with matching points

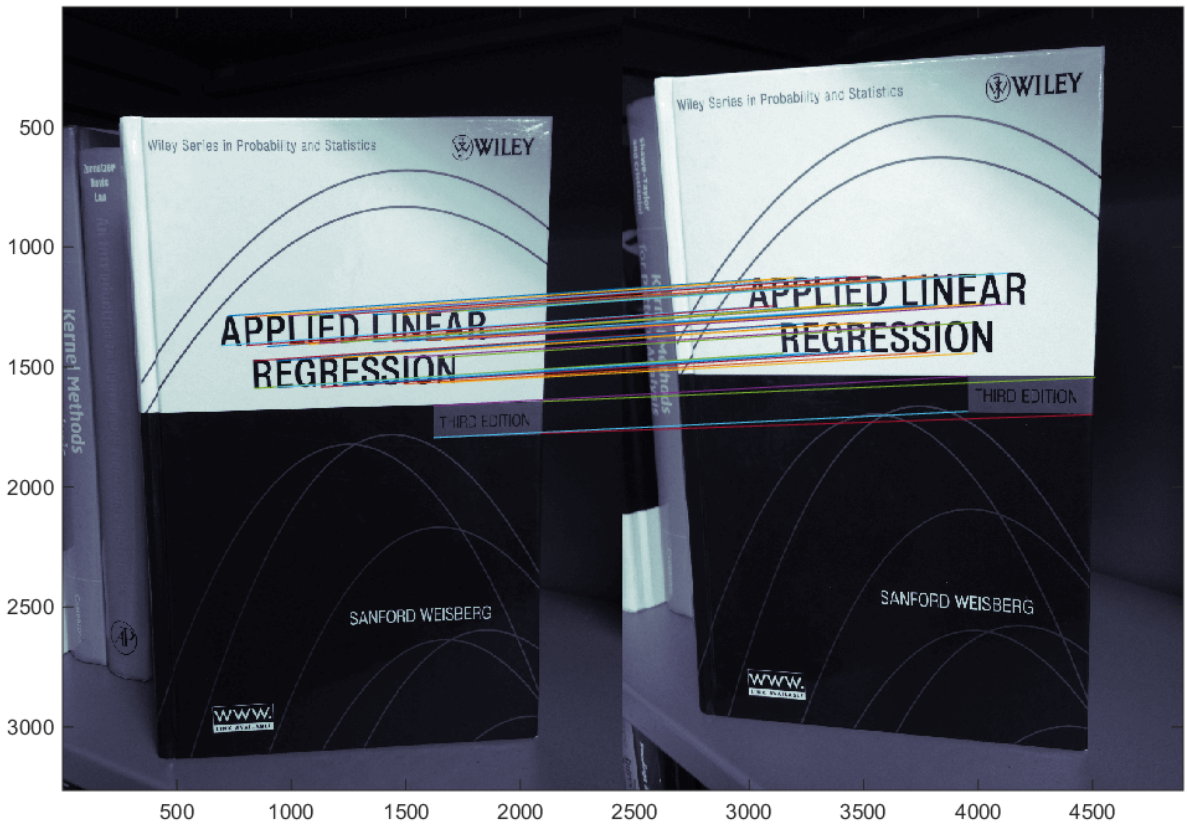


Figure 14: Textbook Images with matching points

Figure 15: Coffee Can Images with matching points

Part b

With the squirtle image, the epipole for the left image lies slightly above the epipole for the right image, suggesting that the left image was taken from a slightly raised viewpoint as compared to the right image. With the coffee can image, the epipole of the left image is below the epipole for right image, suggesting that it was taken from a lowered viewpoint.

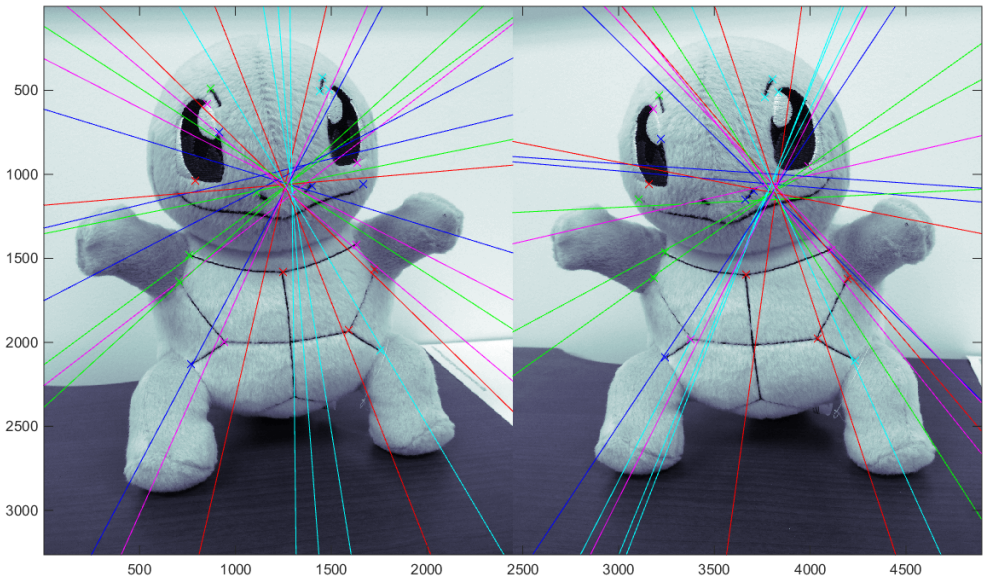


Figure 16: Squirtle Image with Epipolar Lines and their points

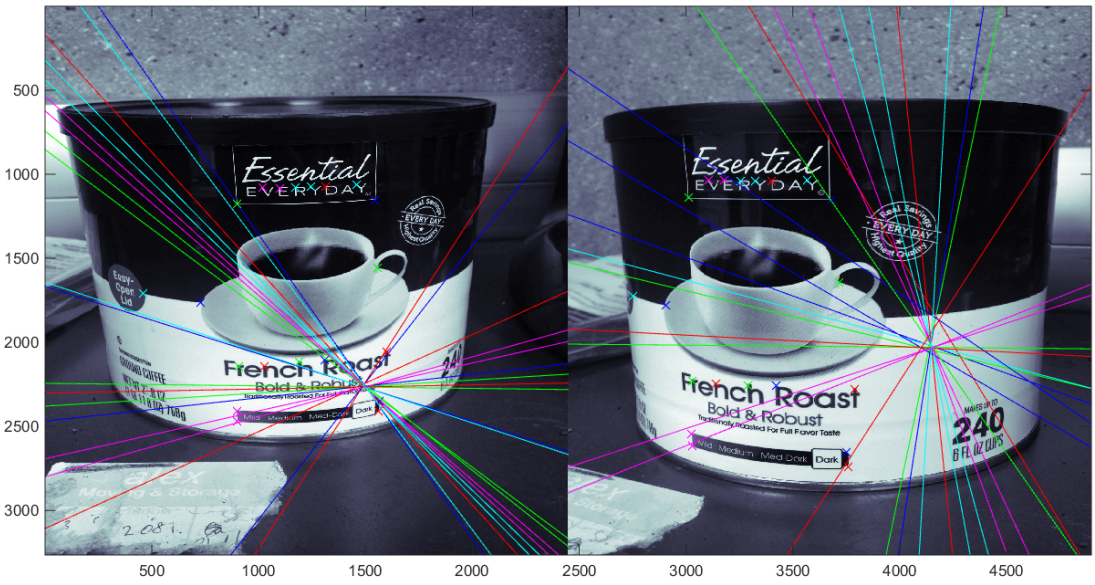


Figure 17: Coffee Can Image with Epipolar Lines and their points

Part c

The estimate for F is quite stable. I decided to compute it multiple times and find the standard deviation of the entries in the F matrix. I then summed the standard deviation.

For squirtle, the sum was near zero and 4 computations of the F matrix were done.
For the textbook, the sum was 0.9567 and 6 computations of the F matrix were done.
For the coffee can, the sum was 1.0140 and 6 computations of the F matrix were done.

The best way to make F more robust would have been to add more points and then see if they correspond. With more data, the estimates get more robust.

Code for Problem 2

```
%This code was used to select image points to use
%imageName = 'coffeeCan';
%imageName = 'squirtle';
imageName = 'book';
image1 = strcat(imageName,'1.JPG');
image2 = strcat(imageName,'2.JPG');
I1 = imread(image2);
%I1 = imread(image1);
I1 = single(rgb2gray(I1));
figure(1)
curImg = imagesc(I1)
colormap bone;
%[X1 Y1] = getpts(1);
[X2 Y2] = getpts(1);

%%
save('coffeeCanPts.mat','X1','X2','Y1','Y2','-v7.3');
%%
load('squirtlePts.mat');
%%
load('bookPts.mat');
%%
load('coffeeCanPts.mat');
%%

%number of iterations of RANSAC
numIter=5;
```

```
%gets an initial randomly sampled set
numInitPts = 8;
randIndices = randperm(size(X1,1));
eightPtsIndices = randIndices(1:numInitPts);
equMatrix = getAmatrix(X1,X2,Y1,Y2,eightPtsIndices);
vecs = null(equMatrix);
ff = vecs./norm(vecs);
fMatrix = reshape(ff,[3 3]);

%performs iterations of RANSAC
%threshold = 0.45; %for squirtle
%threshold = 0.6; %for book
threshold = 0.55; %for coffee can
for iter=1:numIter

    %figures out the inlier and outlier indices
    inlierIndices = [];
    outlierIndices = [];
    for index = 1:size(X1,1)
        img1Vec = [X1(index) Y1(index) 1];
        img2Vec = [X2(index);Y2(index);1];
        value = abs(img1Vec*fMatrix*img2Vec);
        if(value<threshold)
            inlierIndices = [inlierIndices index];
        else
            outlierIndices = [outlierIndices index];
        end
    end

    %recompute F based on Inlier Indices
    equMatrix = getAmatrix(X1,X2,Y1,Y2,inlierIndices);
    [U,S,V] = svd(equMatrix);
    calib = V(:,end);
    ff2 = calib./norm(calib);
    fMatrix = reshape(ff2,[3 3]);
end

%%

%imageName = 'squirtle';
```

```
%imageName = 'coffeeCan';
imageName = 'book';
image1 = strcat(imageName,'1.JPG');
image2 = strcat(imageName,'2.JPG');
I1 = imread(image1);
I2 = imread(image2);
I1 = single(rgb2gray(I1));
I2 = single(rgb2gray(I2));
width = size(I1,2);

%plots the left and right image side-by-side
doubleImage = [I1 I2];
figure
imagesc(doubleImage);
colormap bone;

%plots the matches for the visualized points in each image for Part A
hold on
for i = 1:size(inlierIndices,2)
    index = inlierIndices(i);
   Xvals = [X1(index) X2(index)+width];
   Yvals = [Y1(index) Y2(index)];
   plot(Xvals,Yvals);
end

%%

%Plots the Epipolar lines for Part B
randInds = randperm(size(inlierIndices,2));
randInds = inlierIndices(randInds);

doubleImage = [I1 I2];
figure
imagesc(doubleImage);
colormap bone;

x = linspace(0,width,100);

%plots the matches for the visualized points in each image
hold on
colors = 'bgrcmbgrcmbgrcmbgrcm';
```

```
for i = 1:20
    index = randInds(i);

    %gets the point in image 2 and its line in image 1
    abcVector = fMatrix*[X2(index);Y2(index);1];
    A = abcVector(1); B = abcVector(2); C = abcVector(3);
    curY = -(A.*x + C)./B;

    %gets the point in image 1 and its line in image 2
    abcVector2 = [X1(index) Y1(index) 1]*fMatrix;
    A2 = abcVector2(1); B2 = abcVector2(2); C2 = abcVector2(3);
    curY2 = -(A2.*x + C2)./B2;

    %plots the lines found above
    plot(x,curY,strcat(colors(i),'-'));
    plot(x+width,curY2,strcat(colors(i),'-'));

    %plots the points found above
    plot(X1(index),Y1(index),strcat(colors(i),'x'));
    plot(X2(index)+width,Y2(index),strcat(colors(i),'x'));
end

%%

%This records good fMatrices for use in part C
fMatInd = 0;
%%
fMatInd = fMatInd+1;
fMatrices(fMatInd,:,:) = fMatrix;

%%

%calculates STD to get sense of stability
stdMat = std(fMatrices,1);
sumStd = sum(sum(stdMat));

%for squirtle, sumStd is near zero and 4 computations of fMatrix were done
%for book, sumStd was 0.9567 and 6 computations of fMatrix were done
%for coffee can, sumStd was 1.0140 and 6 computations of fMatrix were done
```

Problem 3

After running everything, these are the two rotation matrices:

$$\begin{pmatrix} 0.9920 & 0.1264 & \text{-3.7932e-5} \\ -0.1264 & 0.9920 & \text{-9.8549e-5} \\ \text{-2.5167e-5} & -0.0001 & -0.9999 \end{pmatrix}$$

$$\begin{pmatrix} -0.9920 & -0.1264 & \text{-1.1563e-4} \\ 0.1264 & -0.9920 & \text{1.8543e-5} \\ \text{1.1704e-4} & \text{-.7748e-6} & -0.9999 \end{pmatrix}$$

The two position vectors are as follows
(7.6779e-5, 4.0003e-5,0.9999)
(-7.6779e-5, -4.0003e-5,-0.9999)

According to these results, it appears that the camera was moved back (or forward) and then rotated to make the second image. This appears consisent with what the images look like. I would have expected slightly more movement in the x, y direction though based on my perception of the image, but perhaps most of the translational movement was in the z direction. That is quite possible since there are only two camera positions we are considering. Figure 18 shows the image correspondences found through SIFT that were used to compute the above matrices and vectors.

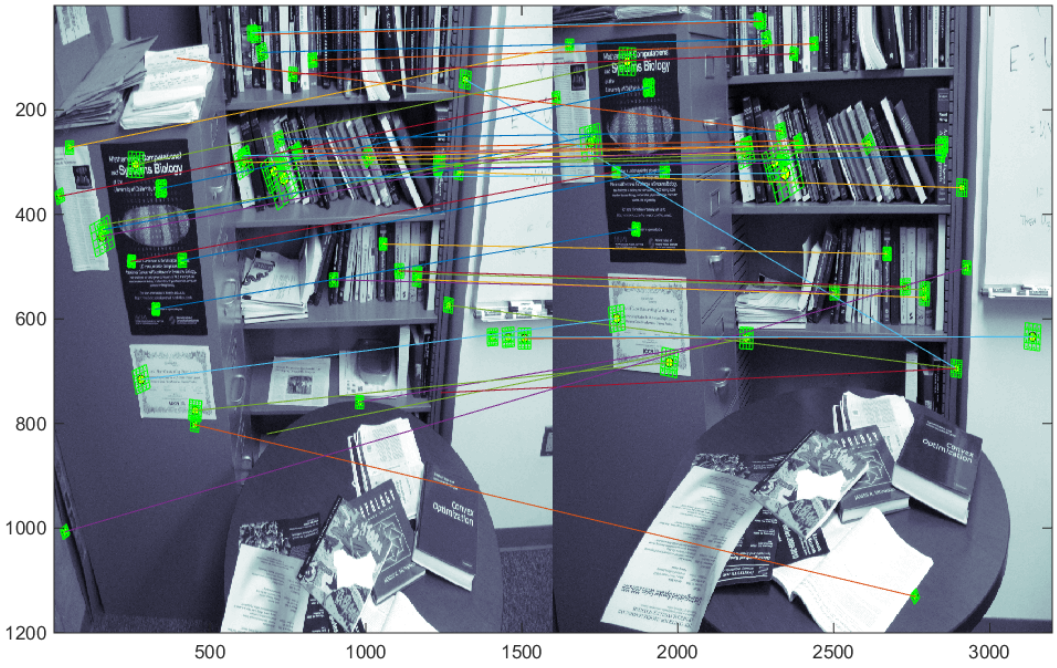


Figure 18: Image Correspondences

Here is the code to complete problem 3

```
%{
Calibration results
from running the calibration toolbox
%}
%— Focal length:
fc = [ 1661.869788486886800 ; 1661.878228311368000 ];
%— Principal point:
cc = [ 836.558147655548620 ; 607.339755380394990 ];
%— Image size:
nx = 1600;
ny = 1200;

%generates instrinsic matrix from these results
K = [nx*fc(1) 0 cc(1);0 ny*fc(2) cc(2); 0 0 1];

%{
This is the script used for problem 1.
```

```
Much of the code was inspired by the tutorial here:
http://www.vlfeat.org/overview/sift.html
%}

image1 = 'sfm/IMG_1434.JPG';
image2 = 'sfm/IMG_1435.JPG';

%run SIFT on image1 and image2
I1 = imread(image1);
I1 = single(rgb2gray(I1));
[f1,d1] = vl_sift(I1);
I2 = imread(image2);
I2 = single(rgb2gray(I2));
[f2,d2] = vl_sift(I2);

%does basic matching
[matches, scores] = vl_ubcmatch(d1, d2) ;
%%
[bestScores,bestScoreInds] = sort(scores);
sel1 = matches(1,bestScoreInds);
sel1 = sel1(1:50);
sel2 = matches(2,bestScoreInds);
sel2 = sel2(1:50);
%%
%goes through the points visualized in sel1 and sel2
%   and finds their corresponding matches in the other images
image1matches = matches(1,:);
image2matches = matches(2,:);
numPoints = size(sel1,2);
image1Points = [];
image2Points = [];
for i = 1:numPoints
    matchingCol = find(image1matches==sel1(i));
    if(~isempty(matchingCol))
        colNum = matchingCol(1);
        image1index = matches(1,colNum);
        image1Points = [image1Points f1(1:2,image1index)];
        img2MatchingIndex = matches(2,colNum);
        image2Points = [image2Points f2(1:2,img2MatchingIndex)];
    end
```

```
end

numPoints = size(sel2,2);
for i = 1:numPoints
    matchingCol = find(image2matches==sel2(i));
    if(~isempty(matchingCol))
       colNum = matchingCol(1);
       image2index = matches(2,colNum);
       image2Points = [image2Points f2(1:2,image2index)];
       img1MatchingIndex = matches(1,colNum);
       image1Points = [image1Points f1(1:2,img1MatchingIndex)];
    end
end

%plots the left and right image side-by-side
doubleImage = [I1 I2];
figure
imagesc(doubleImage);
colormap bone;

%plots the SIFT points for left image
h1 = vl_plotframe(f1(:,sel1)) ;
h2 = vl_plotframe(f1(:,sel1)) ;
set(h1,'color','k','linewidth',3) ;
set(h2,'color','y','linewidth',2) ;
h3 = vl_plotsiftdescriptor(d1(:,sel1),f1(:,sel1)) ;
set(h3,'color','g') ;

%plots the SIFT points for the right image
width = size(I1,2);
f2Original = f2(:,sel2);
f2Offset = repmat([width;0;0;0],1,50);
f2New = f2Original+f2Offset;
h1 = vl_plotframe(f2New) ;
h2 = vl_plotframe(f2New) ;
set(h1,'color','k','linewidth',3) ;
set(h2,'color','y','linewidth',2) ;
h3 = vl_plotsiftdescriptor(d2(:,sel2),f2New) ;
set(h3,'color','g') ;

%plots the matches for the visualized points in each image
```

```
hold on
for i = 1:size(image1Points,2)
    Xvals = [image1Points(1,i) image2Points(1,i)+width];
    Yvals = [image1Points(2,i) image2Points(2,i)];
    plot(Xvals,Yvals);
end

%%

image1Matrix = [image1Points;ones(1,size(image1Points,2))];
image2Matrix = [image2Points;ones(1,size(image2Points,2))];
newImg1Mat = inv(K)*image1Matrix;
newImg2Mat = inv(K)*image2Matrix;

%%
%number of iterations of RANSAC
numIter=5;

X1 = newImg1Mat(1,:);
Y1 = newImg1Mat(2,:);
X2 = newImg2Mat(1,:);
Y2 = newImg2Mat(2,:);
%%

%gets an initial randomly sampled set
numInitPts = 8;
randIndices = randperm(size(X1,2));
eightPtsIndices = randIndices(1:numInitPts);
equMatrix = getAmatrix(X1,X2,Y1,Y2,eightPtsIndices);
vecs = null(equMatrix);
ff = vecs./norm(vecs);
fMatrix = reshape(ff,[3 3]);

%performs iterations of RANSAC
threshold = 0.0001;
for iter=1:numIter

    %figures out the inlier and outlier indices
    inlierIndices = [];
    outlierIndices = [];
    for index = 1:size(X1,2)
```

```
        img1Vec = [X1(index) Y1(index) 1];
        img2Vec = [X2(index);Y2(index);1];
        value = abs(img1Vec*fMatrix*img2Vec);
        if(value<threshold)
            inlierIndices = [inlierIndices index];
        else
            outlierIndices = [outlierIndices index];
        end
    end

    %recompute F based on Inlier Indices
    equMatrix = getAmatrix(X1,X2,Y1,Y2,inlierIndices);
    [U,S,V] = svd(equMatrix);
    calib = V(:,end);
    ff2 = calib./norm(calib);
    fMatrix = reshape(ff2,[3 3]);
end
%%

%computes rotation and translation after obtaining essential matrix
%       from previous code
essMatrix = fMatrix;
[U,S,V] = svd(essMatrix);
Wmatrix = [0 -1 0;1 0 0;0 0 1];
Zmatrix = transpose(Wmatrix);
tMatrix = V*Wmatrix*S*V';

%position vectors
tVecPos = U*[0;0;1];
tVecNeg = -tVecPos;

%rotation matrices
Rmatrix1 = U*Wmatrix*V';
Rmatrix2 = U*Wmatrix'*V';
```