Computer Vision (600.461/600.661)
Homework 3: Line and Point Detection

Instructor: René Vidal

Due 10/02/2014, 11.59PM Eastern

1. **Line detection.**

   (a) Write a function `[line,inliers]=ransac(points,iter,thr,mininlier)` that implements the RANSAC algorithm for fitting a line with parameters `line` $\in \mathbb{R}^3$ to a collection of `points` $\in \mathbb{R}^{2\times P}$. Other input variables are the number of iterations (`iter`), the threshold (`thr`) on the distance from a point to the line used to determine if the point is an inlier, and the minimum number of inliers (`mininlier`) needed to declare a good fit. Other output variables are the set of `inliers` $\in \{0,1\}^P$, where $1 = inlier$ and $0 = outlier$. Write a script `hw3q1a.m` that does the following. Draw 100 points in $[-1,1] \times [-1,1]$ passing through the line $y = x$ and 100 points uniformly at random in $[-1,1] \times [-1,1]$. Add zero mean Gaussian noise to these points with $\sigma = 0.1$. Apply RANSAC to the noisy points. Plot each iteration of RANSAC with the estimated line and the inliers to the estimated line.

   ANSWER:

```
P = 200;
sigma = 0.1; %for Gaussian noise
points=zeros(2,P);

% 1) Add points belonging to x=y line
temp=random('unif',-1,1,[1,100]);
points(1,1:100)=temp;
points(2,1:100)=temp;

% 2) Add random points in [-1,1]x[-1,1]
points(:,101:200)=random('unif',-1,1,[2,100]);

% 3) Add Gaussian noises
temp=random('norm',0,sigma,[2,200]);
points = points + temp;

% 4) Call RANSAC
[line,inliers]=ransac(points,1000,0.1,50);
```

```
function [ransacline,inliers]=ransac(points,iter,thr,mininlier)

points = points';
[numberPoints,c]=size(points);
inliers=zeros([numberPoints,1],'uint8');
ransacline=[0,0,0];
largestInliers=0;
numGoodFit=0;

% Set if you want to plot all iterations
plotAllResults=false;

for i=1:iter
    randIndices=ceil(rand(2,1)*numberPoints);
    while(randIndices(1)==randIndices(2))
    randIndices=ceil(rand(2,1)*numberPoints);
    end
    point1=points(randIndices(1),:);
    point2=points(randIndices(2),:);

```

```
    % "n=[nx,ny]" is the normal vector to the line and "a" is the level of
    % point1 and point2.
    [nx,ny,a]=generateLineParameters(point1,point2);

    %Test again line
    dist=abs(points*[nx;ny]-a);
    IterationInliers=dist<thr;
    numInliers=sum(IterationInliers);
    if(numInliers>=mininlier)
        numGoodFit=numGoodFit+1;
    end
    if(numInliers>largestInliers)
        largestInliers=numInliers;
        ransacline=[nx,ny,a];
        inliers=IterationInliers;
    end

    if(plotAllResults || i<6)
         iterationName = ...
             sprintf('Ransac Iteration %d. Number of Inlieres %d / %d ', ...
             i,numInliers,numberPoints);
         fig=figure('Name',iterationName,'NumberTitle','off'),

         inliersIndices=find(IterationInliers);
         outliersIndices=find(¬IterationInliers);

         plot(points(outliersIndices,1), points(outliersIndices,2),'o',...
             'MarkerFaceColor','g','MarkerEdgeColor','g','MarkerSize',4);
         hold on,
         plot(points(inliersIndices,1),points(inliersIndices,2),'o',...
             'MarkerFaceColor','r','MarkerEdgeColor','r','MarkerSize',4);
         hold on,

         %Xcoord=[point1(1),point2(1)];
         %Ycoord=[point1(2),point2(2)]
         %line(Xcoord,Ycoord);
         [lX,lY]=findLineExtremaInBox(nx,ny,a);
         line(lX,lY);

         legend('Outliers','Inliers')
         title(iterationName)
         axis([-1 1 -1 1])
         set(gca,'XTick',-1:0.25:1)
         set(gca,'YTick',-1:0.25:1)

         outputName=sprintf('ransac%02d',i);
         print(fig,'-dpng',outputName);
    end
end

fprintf('Number of iterartion with a good fit : %d / %d \n',numGoodFit,iter);
fprintf('Largest number of inliers : %d / %d \n',largestInliers,numberPoints);

%Plot Best Result

plotName=sprintf('Ransac Best Result. Number of Inlieres %d / %d ',...
    largestInliers,numberPoints);
fig=figure('Name',plotName,'NumberTitle','off'),

inliersIndices=find(inliers);
outliersIndices=find(¬inliers);

plot(points(outliersIndices,1),points(outliersIndices,2),'o',...
    'MarkerFaceColor','g','MarkerEdgeColor','g','MarkerSize',4);
hold on,
plot(points(inliersIndices,1),points(inliersIndices,2),'o',...
    'MarkerFaceColor','r','MarkerEdgeColor','r','MarkerSize',4);
hold on,
```

```
89
90  [lX,lY]=findLineExtremaInBox(ransacline(1),ransacline(2),ransacline(3));
91  line(lX,lY);
92
93  legend('Outliers','Inliers')
94  title(plotName)
95  axis([-1 1 -1 1])
96  set(gca,'XTick',-1:0.25:1)
97  set(gca,'YTick',-1:0.25:1)
98
99  print(fig,'-dpng','ransacBest');
100
101 end
```

(b) Suppose you are given data drawn from m models and corrupted with outliers. How would you modify RANSAC to fit m models to the data? If w is the fraction of inliers, n is the number of points needed to fit the model, k is the number of samples, and m is the number of models, how many samples k do you need to guarantee that you recover the m models with probability p? How does this number grow with n and m?

ANSWER: Given N data points drawn from m models $M_1, \ldots, M_m$ and contaminated with outliers, we can identify one of the m models by running RANSAC on the entire dataset. However, in this case the points on the other $m-1$ models are considered as outliers to the model being identified. Once the first model has been identified, we can remove the inliers to the first model and apply RANSAC to the remaining data points to identify the second model and so on until all m models have been identified.

To compute the minimum number of trials needed to recover all m models with probability at least p, we assume for the sake of simplicity that all models generate the same number of points, i.e., the proportion of points generated by each model is w/m and the proportion of the outliers to all models is $1-w$.

To fit the first model, we sample n data points from the data. For each trial we have

$$\mathbb{P}\left(\text{All } n \text{ sampled data points are drawn from } M_\ell\right) = \left(\frac{w}{m}\right)^n, \quad \forall \ell = 1, \ldots, m.$$

Now, since the first fitted model is correct as long as all n sampled data points are from the same model, we have

$$\begin{aligned}\mathbb{P}\left(\text{All } n \text{ sampled data points are from the same model}\right) &= \sum_{\ell=1}^{m} \mathbb{P}\left(\text{All } n \text{ sampled data points are from } M_\ell\right) \\ &= m\left(\frac{w}{m}\right)^n.\end{aligned}$$

Now we calculate the probability that we can obtain a correct fitted model within k_1 trails. That is

$$\begin{aligned}p_1 &= \mathbb{P}\left(\text{The first fitted model is correct within } k_1 \text{ trails}\right) \\ &= 1 - \mathbb{P}\left(\text{The first fitted model is incorrect within } k_1 \text{ trails}\right) \\ &= 1 - \prod_{t=1}^{k_1} \mathbb{P}\left(\text{The model fitted in the } t^{\text{th}} \text{ tail is incorrect}\right) \\ &= 1 - \prod_{t=1}^{k_1} \left(1 - \mathbb{P}\left(\text{All } n \text{ sampled data points in the } t^{\text{th}} \text{ tail are from the same model}\right)\right) \\ &= 1 - \left[1 - m\left(\frac{w}{m}\right)^n\right]^{k_1}.\end{aligned}$$

To fit the second model, we condition on the following event,

$$A_1 = \left\{\text{The first fitted model is correct and all corresponding sample points have been removed}\right\}.$$

The event A_1 implies that we now have $N(1-w/m)$ data points and that the proportion of points generated by each one of the $m-1$ remaining models is

$$\frac{Nw/m}{N(1-w/m)} = \frac{w}{m-w}. \tag{1}$$

Following a similar argument to what we did for fitting the first model, we have

$$\mathbb{P}\left(\text{all sampled } n \text{ data points are from the same model} \mid A_1\right) = (m-1)\left(\frac{w}{m-w}\right)^n,$$

and

$$p_2 = \mathbb{P}\left(\text{The second fitted model is correct within } k_2 \text{ trails} \mid A_1\right) = 1 - \left[1-(m-1)\left(\frac{w}{m-w}\right)^n\right]^{k_2}.$$

To fit the ℓ^{th} model, we condition on the following event,

$$A_{\ell-1} = \left\{\text{The first } \ell-1 \text{ fitted models are correct and all corresponding sample points have been removed}\right\}.$$

The event $A_{\ell-1}$ implies that we now have $N(1-(\ell-1)w/m)$ data points and that the proportion of points generated by each one of the remaining $m-\ell+1$ models is

$$\frac{Nw/m}{N(1-(\ell-1)w/m)} = \frac{w}{m-(\ell-1)w}. \tag{2}$$

Following a similar argument to what we did for fitting the first and second models, we have

$$\mathbb{P}\left(\text{all sampled } n \text{ data points are from the same model}\right) = (m-\ell+1)\left(\frac{w}{m-(\ell-1)w}\right)^n.$$

and

$$\begin{aligned} p_\ell &= \mathbb{P}\left(\text{The second fitted model is correct within } k_\ell \text{ trails} \mid A_{\ell-1}\right) \\ &= 1 - \left[1-(m-\ell+1)\left(\frac{w}{m-(\ell-1)w}\right)^n\right]^{k_\ell}. \end{aligned}$$

Next, notice that by the definition of the conditional probability, we have

$$\begin{aligned} &\mathbb{P}\left(\text{All } m \text{ fitted models are correct within } k = \sum_{\ell=1}^m k_\ell \text{ trails}\right) \\ &= \mathbb{P}\left(\text{The first fitted models is correct within } k_1 \text{ trails}\right) \cdot \\ &\quad \mathbb{P}\left(\text{The second fitted models is correct within } k_2 \text{ trails} \mid A_1\right) \\ &\quad\quad \cdot \mathbb{P}\left(\text{The third fitted models is correct within } k_2 \text{ trails} \mid A_2\right) \cdots \\ &\quad\quad\quad \cdot \mathbb{P}\left(\text{The } m^{\text{th}} \text{ fitted models is correct within } k_m \text{ trails} \mid A_{m-1}\right) \\ &= \prod_{\ell=1}^m p_\ell = \prod_{\ell=1}^m \left(1 - \left[1-(m-\ell+1)\left(\frac{w}{m-(\ell-1)w}\right)^n\right]^{k_\ell}\right). \end{aligned}$$

Our goal is to find the smallest k such that

$$\mathbb{P}\left(\text{All } m \text{ fitted models are correct within } k = \sum_{\ell=1}^m k_\ell \text{ trails}\right) \geq p.$$

We can find the optimal k by solving the following optimization problem

$$(\hat{k}_1, ..., \hat{k}_m) = \arg\min_{k_\ell} \sum_{\ell=1}^{m} k_\ell \tag{3}$$
$$\text{subject to } \prod_{\ell=1}^{m} \left(1 - \left[1 - (m-\ell+1)\left(\frac{w}{m-(\ell-1)w}\right)^n\right]^{k_\ell}\right) \geq p.$$

To simplify our calculations, let us further assume that p_ℓ is greater than or equal to $p^{1/m}$ so that $\prod_\ell p_\ell \geq p$. Then (3) becomes the following optimization problem.

$$(\tilde{k}_1, ..., \tilde{k}_m) = \arg\min_{k_\ell} \sum_{\ell=1}^{m} k_\ell \tag{4}$$
$$\text{subject to } 1 - \left[1 - (m-\ell+1)\left(\frac{w}{m-(\ell-1)w}\right)^n\right]^{k_\ell} \geq p^{1/m} \ \ \forall\, \ell = 1, ..., m.$$

Obviously, (4) can be reduced to m subproblems, i.e., for any $\ell = 1, ..., m$, we have

$$\tilde{k}_\ell = \arg\min_{k_\ell} k_\ell \quad \text{subject to} \quad 1 - \left[1 - (m-\ell+1)\left(\frac{w}{m-(\ell-1)w}\right)^n\right]^{k_\ell} \geq p^{1/m}. \tag{5}$$

Then we can obtain the closed form solutions for $\tilde{k}_1, \ldots, \tilde{k}_m$ by

$$\tilde{k}_\ell = \frac{\log(1-p^{1/m})}{\log\left[1-(m-\ell+1)\left(\frac{w}{m-(\ell-1)w}\right)^n\right]} = \frac{\log\left(\frac{1}{1-p^{1/m}}\right)}{\log\left[\frac{1}{1-(m-\ell+1)\left(\frac{w}{m-(\ell-1)w}\right)^n}\right]}$$

Then, the smallest number of trials is

$$k = \sum_{\ell=1}^{m} \tilde{k}_\ell = \sum_{\ell=1}^{m} \frac{\log\left(\frac{1}{1-p^{1/m}}\right)}{\log\left[\frac{1}{1-(m-\ell+1)\left(\frac{w}{m-(\ell-1)w}\right)^n}\right]}.$$

Note that is hard to analyze whether this is exponential or polynomial in n and m, so we have eliminated that part of the question.

(c) Write a function `[line,inliers]=ransac(points,iter,thr,mininlier,nlines)` that extends your function in 1a) to multiple lines so that the ith entry of `inliers` $\in \{0, 1, \ldots,$ `nlines`$\}^P$ is 0 if the ith point is an outlier, or a number in $1, 2, \ldots,$ `nlines` indicating the line to which the ith point belongs. Write a script `hw3q1c.m` that does the following. Draw 100 points in $[-1, 1] \times [-1, 1]$ passing through the line $y = x$, 100 points passing through the line $y = -x$ and 100 points uniformly at random in $[-1, 1] \times [-1, 1]$. Add zero mean Gaussian noise to these points with $\sigma = 0.1$. Apply RANSAC to the noisy points. Plot each iteration of RANSAC with the estimated line and the inliers to the estimated line.

ANSWER:

```
P = 300;
points = zeros(2,P);
sigma = 0.1; %for Gaussian noise

% 1) Add points belonging to x=y line
temp=random('unif',-1,1,[1,100]);
points(1,1:100)=temp;
points(2,1:100)=temp;

% 2) Add points belonging to x=-y line
```

```
temp=random('unif',-1,1,[1,100]);
points(1,101:200)=temp;
points(2,101:200)=-temp;

% 2) Add random points in [-1,1]x[-1,1]
points(:,201:300)=random('unif',-1,1,[2,100]);;

%3) Add Gaussian noises
temp=random('norm',0,sigma,[2,300]);
points = points + temp;

%4) Call RANSAC 2lines
[line,inliers]=ransacn(points,1000,0.1,70,2);
```

```
function [ransacline,inliers]=ransacn(points,iter,thr,mininlier,nlines)

points = points';
[numberPoints,c]=size(points);
inliers=zeros([numberPoints,1],'uint8');
ransacline=zeros(nlines,3);
numInliersPerModel=zeros(nlines,1);
cummulativeNumberInliers=0;
pointSortedByModel=zeros(numberPoints,2);

lineExtremasModel=zeros(nlines,4);
modelColor=zeros(nlines,3);

currentModel=1;
pointsIndex=1:numberPoints;
currentPointSet=[points, pointsIndex'];
currentNumberOfPoints=numberPoints;
plotAllResults=false;

i=1;
while(i≤iter && currentModel< nlines+1)
   randIndices=ceil(rand(2,1)*currentNumberOfPoints);
   while(randIndices(1)==randIndices(2))
   randIndices=ceil(rand(2,1)*currentNumberOfPoints);
   end
   point1=currentPointSet(randIndices(1),:);
   point2=currentPointSet(randIndices(2),:);

   % "n=[nx,ny]" is the normal vector to the line and "a" is the level of
   % point1 and point2.
   [nx,ny,a]=generateLineParameters(point1,point2);

   %Test again line
   dist=abs(currentPointSet(:,1:2)*[nx;ny]-a);
   IterationInliers=dist<thr;
   numInliers=sum(IterationInliers);
   %Plot Results
   if(plotAllResults)
        iterationName = ...
            sprintf('Ransac Iteration %d. Number of Inlieres in this iteration ...
                %d / %d. Inliers in consolidated models %d / %d.',...
            i,numInliers,currentNumberOfPoints,cummulativeNumberInliers,...
            numberPoints);
        fig=figure('Name',iterationName,'NumberTitle','off'),

        inliersIndices=find(IterationInliers);
        outliersIndices=find(¬IterationInliers);

        plot(currentPointSet(outliersIndices,1),...
            currentPointSet(outliersIndices,2),'o','MarkerFaceColor',...
            'g','MarkerEdgeColor','g','MarkerSize',4);
        hold on,
```

```
        plot(currentPointSet(inliersIndices,1),...
            currentPointSet(inliersIndices,2),'o','MarkerFaceColor','r',...
            'MarkerEdgeColor','r','MarkerSize',4);
        hold on,
        [lX,lY]=findLineExtremaInBox(nx,ny,a);
        line(lX,lY);
        legend('Outliers','Inliers')

        initialPosition=1;
        for j=1:currentModel-1
        hold on,
        col=modelColor(j,:);
        lastPosition=initialPosition+numInliersPerModel(j)-1;
        plot(pointSortedByModel(initialPosition:lastPosition,1),...
            pointSortedByModel(initialPosition:lastPosition,2),'o',...
            'MarkerFaceColor',col,'MarkerEdgeColor',col,'MarkerSize',4);
        hold on,
        lmX=lineExtremasModel(j,1:2);
        lmY=lineExtremasModel(j,3:4);
        line(lmX,lmY);
        initialPosition=lastPosition+1;
        end

        title(iterationName)
        axis([-1 1 -1 1])
        set(gca,'XTick',-1:0.25:1)
        set(gca,'YTick',-1:0.25:1)

        outputName=sprintf('ransacn%02d',i);
        print(fig,'-dpng',outputName);
    end

    % Add a new model i above threshold

    if(numInliers>=mininlier)
    % Find model inliers indices
    currentPointSetInliersIndices=find(IterationInliers);
    globalPointSetInliersIndices=currentPointSet(currentPointSetInliersIndices,3);

    %Set model
    inliers(globalPointSetInliersIndices)=currentModel;
    ransacline(currentModel,:)=[nx,ny,a];
    numInliersPerModel(currentModel)=numInliers;

    %Set the line extremas in the box for plot purposes
    [lX,lY]=findLineExtremaInBox(nx,ny,a);
    lineExtremasModel(currentModel,1:2)=lX;
    lineExtremasModel(currentModel,3:4)=lY;
    modelColor(currentModel,:)=[0.5,0.5,0.5]+0.25*rand(1,3);

    %Update current point set by removing inlieers in this model
    currentPointSetNonInliersIndices=find(~IterationInliers);
    currentPointSet=currentPointSet(currentPointSetNonInliersIndices,:);

    %Update the global statistics
    pointSortedByModel(cummulativeNumberInliers+1:cummulativeNumberInliers...
        +numInliers,:)=points(globalPointSetInliersIndices,1:2);
    cummulativeNumberInliers=cummulativeNumberInliers+numInliers;
    currentModel=currentModel+1;
    currentNumberOfPoints=currentNumberOfPoints-numInliers;
    end

    i=i+1;
end

%Plot Final result
fprintf('Total Number Of Iterations %d \n', i);
plotName=sprintf('Final Result. Number of model %d. Inliers in consolidated ...
```

```
        models %d / %d.',...
120     currentModel-1,cummulativeNumberInliers,numberPoints);
121 fig=figure('Name',plotName,'NumberTitle','off'),
122
123 plot(currentPointSet(:,1),currentPointSet(:,2),'o','MarkerFaceColor','g',...
124     'MarkerEdgeColor','g','MarkerSize',4);
125 legend('Outliers');
126
127 initialPosition=1;
128 for j=1:currentModel-1
129 hold on,
130 col=modelColor(j,:);
131 lastPosition=initialPosition+numInliersPerModel(j)-1;
132 plot(pointSortedByModel(initialPosition:lastPosition,1),...
133     pointSortedByModel(initialPosition:lastPosition,2),'o',...
134     'MarkerFaceColor',col,'MarkerEdgeColor',col,'MarkerSize',4);
135 hold on,
136 lmX=lineExtremasModel(j,1:2);
137 lmY=lineExtremasModel(j,3:4);
138 line(lmX,lmY);
139 initialPosition=lastPosition+1;
140 %%modelName=sprintf('Model %d',j);
141 %%legend(modelName);
142 end
143
144 title(plotName)
145 axis([-1 1 -1 1])
146 set(gca,'XTick',-1:0.25:1)
147 set(gca,'YTick',-1:0.25:1)
148
149 print(fig,'-dpng','ransacnBest');
150
151 end
```

(d) Write a script `hw3q1d.m` that does the following. Apply the Canny edge detector to these images. Apply RANSAC to the detected edges. Plot the lines estimated by RANSAC superimposed on the image.
ANSWER:

```
1  ames2=imread('ames2.jpg');
2  ames2=cast(ames2,'double');
3  ames2=ames2/255;
4  ames2=rgb2gray(ames2);
5  ames2=imresize(ames2,[300,400]);
6
7  edgesAmes=edge(ames2,'canny');
8
9  figure('Name','Canny Edge Detection','NumberTitle','off'),
10 imshow(edgesAmes);
11 imwrite(edgesAmes,'cannyAmes.png');
12
13 %Setup point and put them in a [-1,1]x[-1,1] box
14
15 [r,c]=size(ames2);
16 aspectRatio=r/c;
17 height=2*r/c;
18 centeringVerticalShift=(2-height)/2;
19
20 numDetections=sum(sum(edgesAmes));
21 points=zeros(numDetections,2);
22 pointsAdded=0;
23 for i=1:r
24     for j=1:c
25       if(edgesAmes(i,j)==1)
26           posX=2*((j-1+0.5)/c)-1;
27           posY=height*(((r-i)+0.5)/r)-1+centeringVerticalShift;
28
```

```
            pointsAdded=pointsAdded+1;
            points(pointsAdded,1)=posX;
            points(pointsAdded,2)=posY;
        end
    end
end

fig=figure('Name','Detected Points Ames','NumberTitle','off'),

plot(points(:,1),points(:,2),'o','MarkerFaceColor','g','MarkerEdgeColor',...
    'g','MarkerSize',4);
title('Detected Points Ames')
axis([-1 1 -1 1])
set(gca,'XTick',-1:0.25:1)
set(gca,'YTick',-1:0.25:1)

print(fig,'-dpng','pointAmes');

[line,inliers]=ransacn(points,100000,0.003,80,100);
```

2. **Feature point detection and matching.**

(a) Write a function `keypoints=corners(I,w,kappa)` that implements the Harris corner detector. Your function should a) Apply a `w`×`w` Gaussian filter to the image `I` to remove noise. b) Compute the gradients I_x, I_y of the image in x and y directions. c) Compute the Harris operator at each pixel (x, y)

$$K(x,y) = \frac{\det(H(x,y))}{\text{trace}(H(x,y))} \quad \text{where} \quad H(x,y) = \sum_{(u,v)\in W(x,y)} w(u,v) \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix}_{(x+u,y+v)}, \tag{6}$$

where $w(u, v)$ is the Gaussian window function and $W(x, y)$ is a `w`×`w` neighborhood of (x, y). d) Find the set of pixels $\mathcal{H} = \{(x, y) : K(x, y) > \kappa\}$ such that the response of the Harris operator is above a threshold κ. e) Apply non-maximum-suppression in a `w`×`w` pixel neighborhood of each $(x, y) \in \mathcal{H}$. Store the pixel coordinates of the resulting `keypoints`. Write a script `hw3q2a.m` that tests your code on these images for different values of the window size w and the Harris corner threshold κ. For the best choice, plot the results of each step, as well as the Harris corners before and after non-maximum-suppression.

ANSWER:

```
clear;
%set parameters
w=9;
kappa=2000;
list=dir('HW2-images');
filenames={list([[list.bytes]>10000]).name};
for filest = filenames
    file = char(strcat('HW2-images/', filest));
    %test image with suppression
    rgb=imread(file);
    I=rgb2gray(rgb);
    keypoints=corners(I,w,kappa);
    h=figure;
    imshow(rgb);
    hold on;
    scatter(keypoints(2,:),keypoints(1,:));
    hold off;
    % without suppression
    keypoints=corners(I,w,kappa,0);
    h=figure;
    imshow(rgb);
    hold on;
    scatter(keypoints(2,:),keypoints(1,:));
    hold off;
end
```

```
function keypoints=corners(I,w,kappa,suppression)
%Set default parameters
if nargin ==3
    suppression = 1;
end
%Smooth the image
s=size(I);
W=fspecial('gaussian',[w w], 3);
IDx=filter2(W,I);
My = fspecial('sobel');
Mx = transpose(My);
%Computation of Ix, Iy, Ix2, Iy2, Ixy
Ix = filter2(Mx,IDx);
Iy = filter2(My,IDx);
Ix2 = Ix.^2;
Iy2 = Iy.^2;
Ixy = Ix.*Iy;
%Computation of smoothed versions of Ix2, Iy2, and Ixy
Ix2smooth = conv2(Ix2, W, 'same');
Iy2smooth = conv2(Iy2, W, 'same');
Ixysmooth = conv2(Ixy, W, 'same');
%Computation of cornerness measure M
det = (Ix2smooth.*Iy2smooth) - Ixysmooth.^2;
trace =Ix2smooth + Iy2smooth;
K = det./trace;
H2=K>kappa;
%non-maximum suppression if suppression==1
if suppression == 1
    p=0;
    lw=floor(w/2);
    uw=w-lw-1;
    for x = (lw+1):(s(1)-uw)
        for y= (lw+1):(s(2)-uw)
            if(H2(x,y)==1 && sum(sum(K(x,y)>K((x-lw):(x+uw),(y-lw):(y+uw)))) ≥ ...
                w^2-1)
                p=p+1;
                keypoints(:,p)=[x,y];
            end
        end
    end
else
    p=0;
    keypoints=zeros(2,sum(sum(H2)));
    for x = 1:s(1)
        for y= 1:s(2)
            if(H2(x,y)==1)
                p=p+1;
                keypoints(:,p)=[x,y];
            end
        end
    end
end

end
```

(b) Write a function `descriptors=features(I,keypoints)` that extracts a descriptor centered around each keypoint in `keypoints`$\in \mathbb{R}^{2\times P}$. Use a simple vectorized image patch of 9×9 pixels as your descriptor so that `descriptors`$\in \mathbb{R}^{81\times P}$. **Hint:** Study the function `reshape`.

ANSWER:

```
%find the 9*9 pixels for each keypoints in I
%descriptors is of dimension 81*npts
function descriptors=features(I,keypoints)
s=size(I);
npts=length(keypoints(1,:));
```

```
idmat=reshape(1:81,9,9);
%find the 9*9 pixels for each keypoints
descriptors = zeros(81,npts);
for i = 1:npts
    x=int16(keypoints(1,i));
    y=int16(keypoints(2,i));
    min1=max(x-4,1);
    min2=max(y-4,1);
    max1=min(x+4,s(1));
    max2=min(y+4,s(2));
    id=idmat((x-4):(x+4)>0 & (x-4):(x+4) ≤ s(1),(y-4):(y+4)>0 & (y-4):(y+4) ≤ s(2));
    tmpdat=I(min1:max1,min2:max2);
    descriptors(id(:)',i) = tmpdat(:);
end
end
```

(c) Write a function `[matches,ssdvalues]=matching(descriptors1,descriptors2,tau)` that matches two sets of feature descriptors using the sum of squared differences (SSD) as a matching score. Your function should return for each descriptor in image 1 the index of the best match in image 2 provided that the SSD score is above a threshold τ. If a descriptor in image 1 has no match, then the corresponding entry of `matches` should be set to 0. Write a script `hw3q2c.m` that tests your code on these images and plots the resulting matches. Plot also the ROC curve and use it to select the threshold.

ANSWER:

```
clear;
%set parameters
w=9;
kappa=2000;

filenames={'ames','barton','latrobe','library','mergenthaler','shaffer'};

for file = filenames
    %get keypoints and descriptors
    rgb1=imread(char(strcat('HW2-images/', file, '1.jpg')));
    I1=rgb2gray(rgb1);
    keypoints1=corners(I1,w,kappa);
    descriptors1=features(I1,keypoints1);
    rgb2=imread(char(strcat('HW2-images/', file, '2.jpg')));
    I2=rgb2gray(rgb2);
    keypoints2=corners(I2,w,kappa);
    descriptors2=features(I2,keypoints2);

    %matching
    [matches,ssdvalues] = matching(descriptors1,descriptors2,50000);

    %to plot in one image combine the two related images
    s=size(rgb1);
    rgb=rgb1;
    rgb(:,s(2)+1:2*s(2),:)=rgb2;

    %plotting
    h=figure;
    imshow(rgb);
    hold on;
    scatter(keypoints1(2,:),keypoints1(1,:));
    scatter(keypoints2(2,:)+s(2),keypoints2(1,:));

    for i = 1:length(keypoints1(1,:))
        if(matches(i)≠0)
            p1 = [keypoints1(1,i),keypoints1(2,i)];
            p2 = [keypoints2(1,matches(i)),keypoints2(2,matches(i))+s(2)];
            plot([p1(2),p2(2)],[p1(1),p2(1)],'m');
        end
    end
    hold off;
```

```
42
43 end
```

```
1  function [matches,ssdvalues] = matching(descriptors1,descriptors2,tau)
2  s1=size(descriptors1);
3  s2=size(descriptors2);
4  %for each one in descriptors1 find id with the smallest ssd in descriptors2
5  for i = 1:s1(2)
6      tmpmin=Inf;
7      for j = 1:s2(2)
8          tmpssdv=sum((descriptors1(:,i)-descriptors2(:,j)).^2);
9          if(tmpssdv < tmpmin)
10             tmpmin=tmpssdv;
11             tmpid=j;
12         end
13     end
14     if(tmpmin < tau)
15         matches(i) = tmpid;
16         ssdvalues(i) = tmpmin;
17     else
18         matches(i) = 0;
19         ssdvalues(i) = 0;
20     end
21
22 end
23 end
```

Submission instructions. Send email to vision14jhu@gmail.com with subject 600.461/600.661:HW3 and attachment `firstname-lastname-hw3-vision14.zip` or `firstname-lastname-hw3-vision14.tar.gz`. The attachment should have the following content:

1. A file called `hw3.pdf` containing your answers to each one of the analytical questions. If at all possible, you should generate this file using the latex template hw1-vision14.tex. If not possible, you may use another editor, or scan your handwritten solutions. But note that you must submit a single PDF file with all your answers.

2. For coding questions, submit a file called `README`, which contains instructions on how to run your code. Use separate directories for each coding problem. Each directory should contain all the functions and scripts you are asked to write in separate files. For example, for HW2 the structure of what you should submit could look like

   (a) `README`

   (b) `hw2.pdf`

   (c) `hw2q3`: `hw2q3c.m`, `hw2q3e.m`

   (d) `hw2q4`: `hw2q4b.m`, `hw2q4c.m`

   The TA will run your scripts to generate the results. Thus, your script should include all needed plotting commands so that figures pop up automatically. Please make sure that the figure numbers match those you describe in `hw2.pdf`. You do not need to submit input or output images. The output images should be automatically generated by your scripts so that the TA can see the results by just running the scripts. In writing your code, you should assume that the TA will place the input images in the directory that is relevant to the question solved by your script. Also, make sure to comment your code properly.