

1)

$$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 2 & 0 & 1 & 0 \\ 0 & 1 & -1 & 2 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \end{bmatrix} * \begin{bmatrix} 1 & -1 \\ 1 & -1 \end{bmatrix} = \begin{bmatrix} -1 & 1 \\ -1 & 1 \end{bmatrix}$$

$$0 \cdot (-1) + 0 \cdot 1 + 0 \cdot (-1) + 0 \cdot 2 \cdot (-1) = 2$$

$$0 \cdot (-1) + 0 \cdot 1 + 2 \cdot (-1) + 0 \cdot 1 = -2$$

$$0 + 0 + 0 + 1 = 1$$

$$0 + 0 + 0 + 1 = 1$$

$$0 + 2 + 0 + 1 = 3$$

$$-2 + 0 - 1 - 1 = -4$$

$$0 + 1 + 1 + 2 = 4$$

$$-1 + 0 - 2 + 0 = -3$$

$$0 + 0 + 0 + 1 = 1$$

$$0 - 1 - 1 + 0 = -2$$

$$1 + 2 + 0 + 0 = 3$$

$$-2 + 0 + 0 + 0 = -2$$

$$\begin{bmatrix} 2 & -2 & 1 & -1 \\ 3 & -4 & 4 & -3 \\ 1 & -2 & 3 & -2 \end{bmatrix}$$

b)
$$1 * F_1 = \begin{bmatrix} 2 & 0 & 1 \\ 3 & -1 & 3 \\ 1 & -1 & 2 \end{bmatrix}$$

$$1 * F_1 + F_2 = \begin{bmatrix} 2 & 0 & 1 \\ 3 & -1 & 3 \\ 1 & -1 & 2 \end{bmatrix} * \begin{bmatrix} 1 & -1 \end{bmatrix} = \begin{bmatrix} 2 & -2 & 1 & -1 \\ 3 & -4 & 4 & -3 \\ 1 & -2 & 3 & -2 \end{bmatrix}$$

(zero padding ამ კერნელს თითქოს მარცხ მარცხნივ და გვერდით უნდა და, მაგრამ ყველა ამბობდა ოთხივე მხარეს უნდაო და იმედია არ შეეშალა)

C. რანკი უნდა იყოს ჩვენს შემთხვევაში 1, იმიტომ, რომ ის ერთი ვექტორი რის გამოც რანკი არის და მეორე მაგის წრფივი კომბინაციები გამოვიღეს. მაგალითად $[1 \ 2; 7 \ 14; 3 \ 6]$ როცა გვაქვს ღამისლევა $[1; 7; 3]$ და ამ სვეტის გამრავლებები $[1 \ 2]$ ამ შემთხვევაში.

- D. 1. 48
- 2. yes
- 3. 42
- 4. b

E. 1. $m^2 * n^2 * m^1 * n^1$

2. $m^2 * n^1 * m^1 + n^2 * m^1 * n^1$

3. $O(n^2)$ ოპერაცია იმიჯისთვის და თითო ოპერაცია მოიცავს $O(n^2)$ a სთვის და $O(n)$ b სთვის ფილტრის ოპერაციას.

2.

```
photo = rgb2gray(imread('u2cuba.jpg'));  
template = rgb2gray(imread('trailer.png'));
```

```
figure('Name','Original photo');  
imshow(photo, [], 'InitialMagnification', 50);  
figure('Name','template');  
imshow(template, [], 'InitialMagnification', 50);
```

% a)

```
% peak is where correlation was maximal. this means that peak is where it's  
% most likely for photo and template to match. the straight-lines were homogeneous  
% and they caused straight-line artifacts.  
correlationImg = normxcorr2(template,photo);
```

```
figure('Name','Correlation, near-perfect template');  
hold on  
imshow(correlationImg, [], 'InitialMagnification', 50);
```

```
[~, maxValLinearIndex] = max(correlationImg(:));
```

```
[y, x] = ind2sub(size(correlationImg), maxValLinearIndex);
```

```
rectangle('Position', [x-15, y-15, 30, 30], 'EdgeColor', 'r')  
hold off;
```

```
largerTemplate = rgb2gray(imread('trailerSlightlyBigger.png'));
```

% b)

```
% this method is not scale-invariant. this means that if what we are  
% searching is scaled, cross-correlation can't find it. In real world it can  
% only be used in factory or where distance between camera and object  
% always stays the same.  
mismatchedCorrelationImg = normxcorr2(largerTemplate,photo);
```

```
figure('Name', 'Correlation, larger-sized template');  
hold on;  
imshow(mismatchedCorrelationImg, [], 'InitialMagnification', 50);
```

```
[~, maxValLinearIndex] = max(mismatchedCorrelationImg(:));  
[y, x] = ind2sub(size(mismatchedCorrelationImg), maxValLinearIndex);  
rectangle('Position', [x-15, y-15, 30, 30], 'EdgeColor', 'r')
```

hold off;

c) $O(n^2 m^2)$ იყო და ახლა $O(n^2 m^2 Nr Ns)$ გახდება

3.

a)

$$\begin{aligned}
 G_y &= 0 \Rightarrow Mg = G_x \\
 \theta &= \tan^{-1} \left(\frac{G'_y}{G'_x} \right) & G'_x &= G_x \cos \theta \\
 G'_y &= G'_x \tan \theta = G_x \cos \theta \tan \theta = G_x \sin \theta \\
 Mg' &= \sqrt{(G_x \sin \theta)^2 + (G_x \cos \theta)^2} = G_x = Mg
 \end{aligned}$$

b)

სეგმენტაციის მოსაგვარებლად ზედა ზღვარს ღვადაბლებდი, რომ გადაგდებულ სეგმენტებში კანგი მეზობლები გაზრდილიყო და აღარ გადავარდნილიყო. ნოისის მოსაშორებლად ქვედა ზღვარს ავამაღლებდი, რის გამოც ნოისის ნაწილს საერთოდ არ განვიხილავდი და მხოლოდ მაღალ გრადიენტიანი ნოისი დამრჩებოდა, რაც იშვიათია, რადგან ნოისი რენდომია და მე უკვე გაუსიანი მაქვს გადატარებული, რამაც ფოტო გაბლარა და ნოისის გრადიენტი დაადაბლა.

4.
a/b)

The image shows a handwritten derivation of the second derivative of a Gaussian function on a grid-lined notebook page. The derivation is as follows:

$$\begin{aligned} \left(\frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{x^2}{2\sigma^2}} \right)'' &= \frac{1}{\sqrt{2\pi}\sigma} \left(-\frac{x}{\sigma^2} e^{-\frac{x^2}{2\sigma^2}} \right)' = \\ &= -\frac{1}{\sqrt{2\pi}\sigma^3} \left(x e^{-\frac{x^2}{2\sigma^2}} \right)' = \\ &= -\frac{1}{\sqrt{2\pi}\sigma^3} \left(e^{-\frac{x^2}{2\sigma^2}} + \frac{x^2}{\sigma^2} e^{-\frac{x^2}{2\sigma^2}} \right) \end{aligned}$$

```
function g=gaussianSecDer(x,sigma)
g= -(1/(sqrt(2*pi)*sigma)).*(1-x.^2./sigma^2).*exp(-x.^2./(2*sigma^2));
end
```

```
function d = DifferanceOfGaussian( x, k, sigma )
d = (gaussian(x,k*sigma) - gaussian(x,sigma))/(k*sigma-sigma);
end
```

```
function g = gaussian(x, sigma)
g = (1/(sqrt(2*pi)*sigma))*exp(1).^(-x.^2/(2*sigma^2));
end
```

```
x=-5:0.001:5;
sigma=1;
g=gaussianSecDer(x,sigma);
p2=DifferanceOfGaussian(x,1.2,sigma);
p4=DifferanceOfGaussian(x,1.4,sigma);
```

```
p6=DifferenceOfGaussian(x,1.6,sigma);  
p8=DifferenceOfGaussian(x,1.8,sigma);  
two=DifferenceOfGaussian(x,2.0,sigma);  
plot(x,g,x,p2,x,p4,x,p6,x,p8,x,two);  
legend('gaussian','k=1.2','k=1.4','k=1.6','k=1.8','k=2.0');
```

c) რგოლი

5.

a) function [x, y, R] = FitCircle(D)

% FitCircleLeastSquares Fit a circle using at least 3 points.

% Input:

% D: An N x 2 matrix, where each row is a point in 2D space.

% Output:

% x, y, R: (x, y) is the center of the fitted circle, R is the radius of

% the fitted circle

```
n = size(D, 1);
```

```
if n < 3,
```

```
    error('You need at least three points to fit a circle.');
```

```
end
```

```
x = 0;
```

```
y = 0;
```

```
R = 1;
```

```
a=D(:,1);
```

```
b=D(:,2);
```

```
c=ones(n,1);
```

```
c=repmat(1,n,1);
```

```
A=[2*a,2*b,c];
```

```
Y=[a.^2+b.^2];
```

```
xyR=A\Y;
```

```
x=xyR(1,:);
```

```
y=xyR(2,:);
```

```
R=xyR(3,:)+x^2+y^2;
```

```
R=sqrt(R);
```

```
end
```

b) function [x, y, R] = RANSAC(D, maxIter, maxInlierError, goodFitThresh)

% RANSAC Use RANSAC to fit circles to a set of points.

% Input:

% D: The data to fit. An N x 2 matrix, where each row is a 2d point.

% maxIter: the number of iterations RANSAC will run

% maxInlierError: A point not in the seed set is considered an inlier if

% its error is less than maxInlierError. Error is

% measured as $\text{abs}(\text{distance}^2 - R^2)$, using the provided

```

%          ComputeErrors() function
% goodFitThresh: The threshold for deciding whether or not a model is
%          good; for a model to be good, at least goodFitThresh
%          non-seed points must be declared inliers.
%
% Output:
% x, y, R: (x, y) is the center of the fitted circle, R is the radius of
% the fitted circle
%
%

% The number of randomly-chosen seed points that we'll use to fit our
% initial circle
seedSetSize = 3;
% The number of data points that weren't in the seed set.
nonSeedSetSize = size(D, 1) - seedSetSize;

% x, y, and R are the best parameters that we have seen so far.
x = 0;
y = 0;
R = 0;

% bestError is the error associated with the best set of parameters we
% have seen so far.
bestError = Inf;

for i = 1:maxIter
    % Randomly partition the data into seed and non-seed sets.
    % The RandomlySplitData() function below may be helpful
    seedSet = zeros(seedSetSize, 2);
    nonSeedSet = zeros(nonSeedSetSize, 2);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                                     %
%          YOUR CODE HERE: Fill in the above two variables.          %
%                                     %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    [seedSet,nonSeedSet]=RandomlySplitData(D,seedSetSize);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                                     %
%          END YOUR CODE          %

```



```
%  
%  
%% Use the seed set to fit a model using least squares. The  
% function you made in part (a) may be useful.  
xx = 0;  
yy = 0;  
RR = 0;  
  
%  
%  
% YOUR CODE HERE: Fill in the above 3 variables. %  
%  
%  
%  
%  
%  
[xx,yy,RR]=FitCircle(seedSet);  
%  
%  
%  
%  
% END YOUR CODE %  
%  
%  
%  
% Compute the error of this model on the non-seed set, using the  
% ComputeErrors function below.  
nonSeedErrors = zeros(nonSeedSetSize, 1);  
%  
%  
% YOUR CODE HERE. Fill in the above variable. %  
%  
%  
%  
%  
nonSeedErrors=ComputeErrors(xx,yy,RR,nonSeedSet);  
%  
%  
% END YOUR CODE %  
%  
%
```



```

%
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

```

```

    % If this model is better than any we've seen before then
    % record its parameters.
    if error < bestError
        bestError = error;
        x = xx;
        y = yy;
        R = RR;
    end
end
end
if R == 0
    disp('No RANSAC fit was found.')
end
end
end

```

```

function [D1, D2] = RandomlySplitData(D, splitSize)
% Randomly split the rows of a matrix into two sets.
%
% Input:
% D: n x m matrix of data to split
% splitSize: The desired number of elements in the first set.
%
% Output:
% D1: splitSize x m matrix containing splitSize rows of D chosen at random
% D2: (n - splitSize) x m matrix containing the rest of the rows of D.
    idx = randperm(size(D, 1));
    D1 = D(idx(1:splitSize), :);
    D2 = D(idx(splitSize+1:end), :);
end

```

```

function error = ComputeErrors(x, y, R, D)
% Compute the error associated with fitting the circle (x, y, R) to the
% data in D.
%
% Input:
% x, y, R: Center and radius of the circle
% D - n x 2 matrix where each row is a data point [x_i, y_i]
%
% Output:

```

```
% error: An n x 1 vector where error(i) is the error of fitting the data
% point D(i, :) to the circle (x, y, R).
% Error is measured as abs(dist^2-R^2). This error measure isn't
% very intuitive but it's fast.
```

```
error = abs((x-D(:,1)).^2 + (y-D(:,2)).^2 - R^2);
end
```

c) N როცა იზრდება ველარ პოულობ ისე მარტივად ამ მონაცემებით. ამიტომ უნდა გაეზარდოს იტერაციების რაოდენობა და მინიმალური ინლაიერების რაოდენობა (ეს თავიდანვე მთლიან რაოდენობაზე დამოკიდებული რომ ყოფილიყო აჯობებდა ალბათ)