ამოცანა 1

ა) კონვოლუციისათვის f დავატრიალოთ ორივე x და y ღერძის გარშემო მივიღებთ $F' = \begin{bmatrix} -1 & 1 \\ -1 & 1 \end{bmatrix}$

შემდეგ უბრალო გადაწყვეტის გამოყენება

$$I * F = \begin{bmatrix} -4 & +4 & -3 \\ -2 & +3 & -2 \end{bmatrix} = \begin{bmatrix} -4 & 4 & -3 \\ -2 & 3 & -2 \end{bmatrix}$$

თითოეული ელემენტის დასათვლელად [illegible] [illegible] [illegible] ზედა [illegible] ელემენტებს I-ში f'-ის [illegible] [illegible] ვამრავლებთ და ვკრებთ. თუ f' [illegible] [illegible] [illegible] 0-ად ვთვლით I-ის გარეთ ელემენტებს.

ბ) $f_1' = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$ $f_2' = \begin{bmatrix} -1 & 1 \end{bmatrix}$

$$I * f_1' = \begin{bmatrix} 3 & -1 & 3 \\ 1 & -1 & 2 \end{bmatrix}$$

$$(I * f_2') * f_2' = \begin{bmatrix} -4 & 4 & -3 \\ -2 & 3 & -2 \end{bmatrix}$$

ანუ დაემთხვა პირველი გადაწყვეტის.

ამოცანა 1

ბ) განვიხილოთ ინდექსირებული [i,j] ელემენტი თუ $f_1 f_2$-ის შევცვლით იგივეა რაც F-ის შედეგი ე.ი. ეკვივალენტობას აქვს ადგილი.

$$((I * f_1) * f_2)[i,j] = \left(\sum_k I[i-k,j] \cdot f_1[k]\right) * f_2$$

ახლა f_2-ით მოვქმედოთ და გავშალოთ

$$\sum_l \left(\sum_k I[i-k,j-l] \cdot f_1[k]\right) \cdot f_2[l] =$$

$$= \sum_{k,l} I[i-k,j-l] \cdot f_1[k] f_2[l] \quad (A)$$

ანუ [illegible] აქვს $\sum_{k,l} f_1[k] f_2[l] = F[k,l]$

რადგან $F = f_1 \cdot f_2$, ანუ

$$(A) = \sum_{k,l} I[i-k,j-l] \cdot F[k,l] = I * F$$

რ.დ.გ.

ამოცანა 1

ა) (i) I-ს flip და shift ვაკეთებს ცალ-ცალ. თითო ელემენტისთვის [illegible] $2\times2=4$ გამრავლება სულ, ანუ ჯამში $2\times3\times4=24$

(ii) აბსოლუტურად იგივე, 24

(iii) ჯამში ასე $2\times3\times2+2\times3\times2=24$

(iv) რაოდენობა იგივეა

ბ) (i) თითოეული ელემენტისთვის I-დან $M_1\times N_1$ ანუ ჯამში $M_2\times M_2\times N_2\times N_2$

(ii) ერთის ვექტორისთვის $M_1\times N_1\times M_2$
მეორისთვის $M_2\times N_2\times N_2$
ჯამში $M_2\times N_1(M_1+N_2)$

(iii) ნაკლები ვიდრე მაქსიმუმი $O(M_2N_2(M_1+M_1)K$
$O(M_1N_1M_1N_2)$, რადგან [illegible] [illegible]
ეფექტი [illegible] [illegible] [illegible] ნელი.
ანუ [illegible] 1D [illegible] [illegible] 2D-ს.

ამოცანა 2

ა) ესაა კოდი:

```
% Read in the main image and the template image.
photo = rgb2gray(imread('u2cuba.jpg'));
template = rgb2gray(imread('trailer.png'));

% Display the image and the template at the same magnification, to make
% clear that the template is an extremely close match for the relevant
% section of the image.
figure('Name','Original photo');
imshow(photo, [], 'InitialMagnification', 50);

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

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                                                                    %
%                         YOUR CODE HERE: Part (a)                      %
%     Use normxcorr2 to cross-correlate the template image with the photo.     %
%                     (Your code should be very simple.)                    %
%                                                                    %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
correlationImg = normxcorr2(template(:,:,1),photo(:,:,1));

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                                                                    %
%                              END OF YOUR CODE                             %
%                                                                    %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Displays correlationImg and draw a rectangle around the highest peak.
figure('Name','Correlation, near-perfect template');
hold on % Allows drawing on top of the displayed image
imshow(correlationImg, [], 'InitialMagnification', 50);

% Find the maximum of correlationImg.
% The max function requires a vector, so we unroll the array with (:).
% The max function returns both the value of the maximum value and its
% linear index within the array. We don't care about the value of the
% maximum, so we discard the first output value with using the ~ notation.
[~, maxValLinearIndex] = max(correlationImg(:));

% We use the ind2sub function to convert a linear back into matrix
% coordinates.
[y, x] = ind2sub(size(correlationImg), maxValLinearIndex);

% Display a rectangle around the peak we just found.
rectangle('Position', [x-15, y-15, 30, 30], 'EdgeColor', 'r')
hold off;

% Load a template which is sized slightly larger than the object appears in
% the image.
```

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

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                                                        %
%                   YOUR CODE HERE: Part (b)                    %
%          Repeat the process from (a) with the larger template.        %
%                                                        %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
mismatchedCorrelationImg = normxcorr2(largerTemplate(:,:,1),photo(:,:,1));

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                                                        %
%                      END OF YOUR CODE                          %
%                                                        %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

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

% Repeat the same process as above: find the maximum of correlationImg and
% draw a box around it.
[~, maxValLinearIndex] = max(mismatchedCorrelationImg(:));
[y, x] = ind2sub(size(mismatchedCorrelationImg), maxValLinearIndex)
rectangle('Position', [x-15, y-15, 30, 30], 'EdgeColor', 'r')
hold off;
```

კროს-კორელაციისას რადგან საძებნი სურათი თეთრი იყო შავ ფონზე, თეთრი ხაზები უფრო გამოკვეთილად ჩანს დამუშავებულ სურათშიც, რაც არტეფაქტების მიზეზია. რადგან ზომა ზუსტი იყო ზუსტად იპოვა.
ბ) უკვე ვეღარ იპოვა, რადგან ზომა განსხვავებულია და სადღაც უფრო ზემოთ მონიშნა წითლად. გამოსადეგია თუ ზუსტად ვიცით რას ვეძებთ, ან მცირე ფარგლებში მაინც.
გ)ყველა ვარიანტი შესამოწმებლად   - O(Nr * Ns) თითოეული ვარიანტის შემოწმება კი მეოთხე ხარისხის დიდი ო ნოტაციაა, O(n*n*m*m), რადგან ვთვლით რომ პირდაპირ ვაკეთებთ კროს-კორელაციას და კონსტანტებს არ ვითვლით, ასევე არ ვითვალისწინებთ იმ ოპტიმიზაციებს რაც ცხადია მატლაბის კოდში იქნება. ანუ ჯამური ოპერაციათა რაოდენობის შეფასება – O(Nr * Ns * n * n * m * m)

ამოცანა 3

ა) Canny edge detector-ში მნიშვნელობა აქვს მხოლოდ გრადიენტს. გრადიენტის მიხედვით, $G = \sqrt{G_x^2 + G_y^2}$ სადაც G_x არის ჰორიზონტალური წარმოებული, G_y კი ვერტიკალური.

მოცემულ შემთხვევაში $G = \sqrt{x^2 + 0^2} = \sqrt{x^2}$, ($y = 0$ სადაც x ღერძზეა)

მოცემულის შემთხვევაში $G = \sqrt{x^2\cos^2\theta + x^2\sin^2\theta} = \sqrt{x^2}$

ანუ შენარჩუნდება G, ანუ იგივე იქნება.

ბ) იმისთვის რომ გამოჩნდეს edge-ები შევცვალოთ, რამოდენიმე ბევრი წერტილი მაინც სწორ ან გადაიხრებოდეს. ამის შემდეგ ავიღოთ ძალიან ბევრი წერტილი მაინც, სანამ noise არ გაქრება. სადაც გადაიხრება შევცვალოთ რომ ვასწორ არსებობს, ამ ვიპოვით.

ამოცანა 4

ა) fn = @(x) (1/sqrt(2*pi)*exp(-x*x/2)*(x*x-1));
fplot(fn,[-8,8]);
ეს მატლაბის კოდი დახატავს გაუსის მეორე წარმოებულს როცა სიგმა=1
ბ)k=1;

```
for c = 1:5
    k=k+0.2;
    gaussK = @(x) (1/(sqrt(2*pi)*k) *exp(-x*x/(2*k*k)));
    KK=1;
    gaussKK = @(x) (1/(sqrt(2*pi)*KK) *exp(-x*x/(2*KK*KK)));
    sxvaoba = @(x) ((gaussK(x) - gaussKK(x))/(k-1));
    figure('Name',strcat('k=',num2str(k))),fplot(sxvaoba,[-8;8]);
end
```

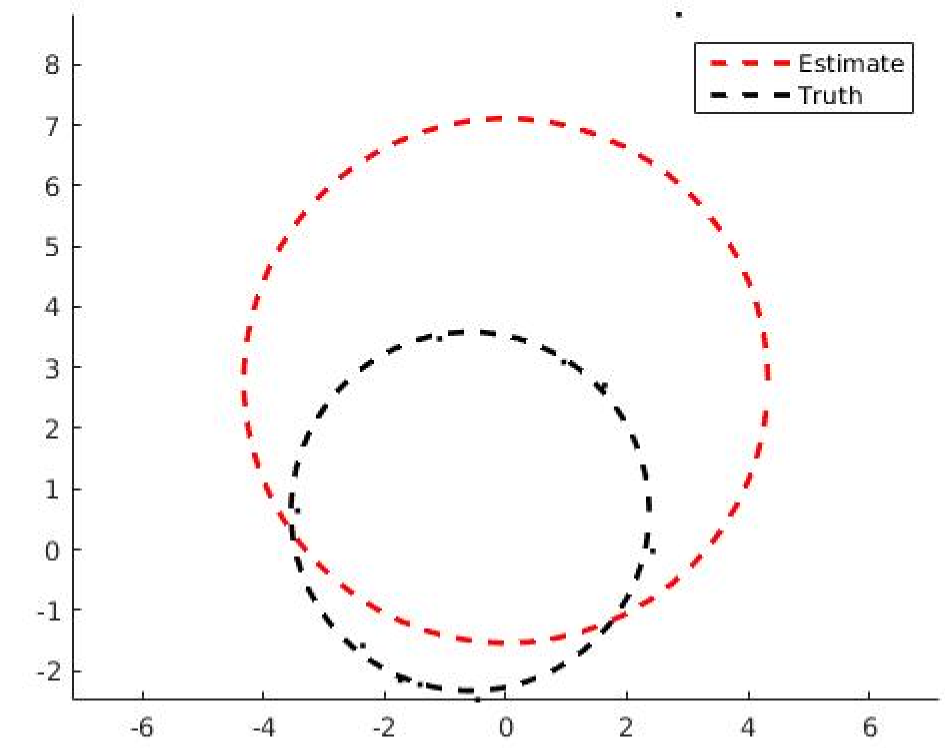
ეს კოდი დახატავს k=1.2-დან k=2-ის ჩათვლით, სიგმა=1.
ყველაზე დიდი დამთხვევა არის k=1.2-ის დროს. ეს კარგად ჩანს შემდეგი კოდის გაშვებისას

```
fn = @(x) (1/sqrt(2*pi)*exp(-x*x/2)*(x*x-1));
k=1;
for c = 1:5
    k=k+0.2;
    gaussK = @(x) (1/(sqrt(2*pi)*k) *exp(-x*x/(2*k*k)));
    KK=1;
    gaussKK = @(x) (1/(sqrt(2*pi)*KK) *exp(-x*x/(2*KK*KK)));
    sxvaoba = @(x) ((gaussK(x) - gaussKK(x))/(k-1));
    figure('Name',strcat('k=',num2str(k))),fplot(sxvaoba,[-8;8]);
    hold on
    fplot(fn,[-8,8]);
    hold off
end
```

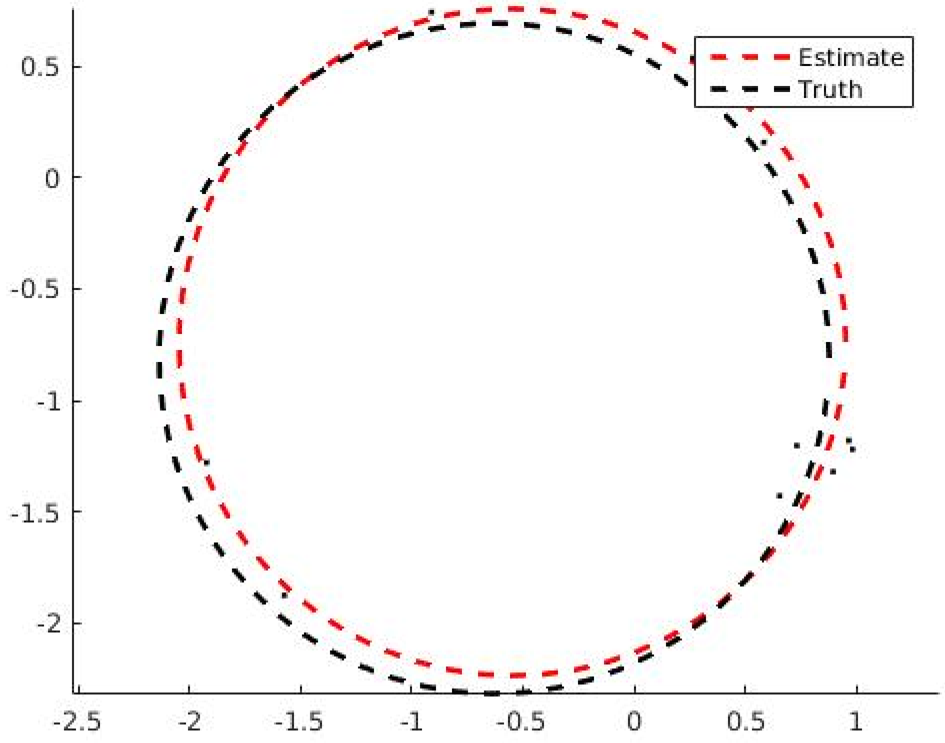
გ) გაუსიანი კარგად ამოიცნობს ბუბლიკს, ანუ ვთქვათ დიდი შავი წრე რომ იყოს ცენტრში და გარშემო თეთრი რგოლი და იმ თეთრი რგოლის გარეთ სიშავე.

ამოცანა 5)

```
ა)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;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                                                              %
%                          YOUR CODE HERE:                        %
%       Find the value of the above variables as discussed in part (a).       %
%                                                              %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


A = [];
B = [];
for i = 1:n
    A(i,1) = 2*D(i,1);
    A(i,2) = 2*D(i,2);
    A(i,3) = 1;
    B(i,1) = D(i,1)*D(i,1)+D(i,2)*D(i,2);
end
res = A\B;
x=res(1);
y=res(2);
R = sqrt(res(3)+res(1)*res(1)+res(2)*res(2));

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                                                              %
%                          END YOUR CODE                        %
%                                                              %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
end
```



დაგენერირებული სურათები

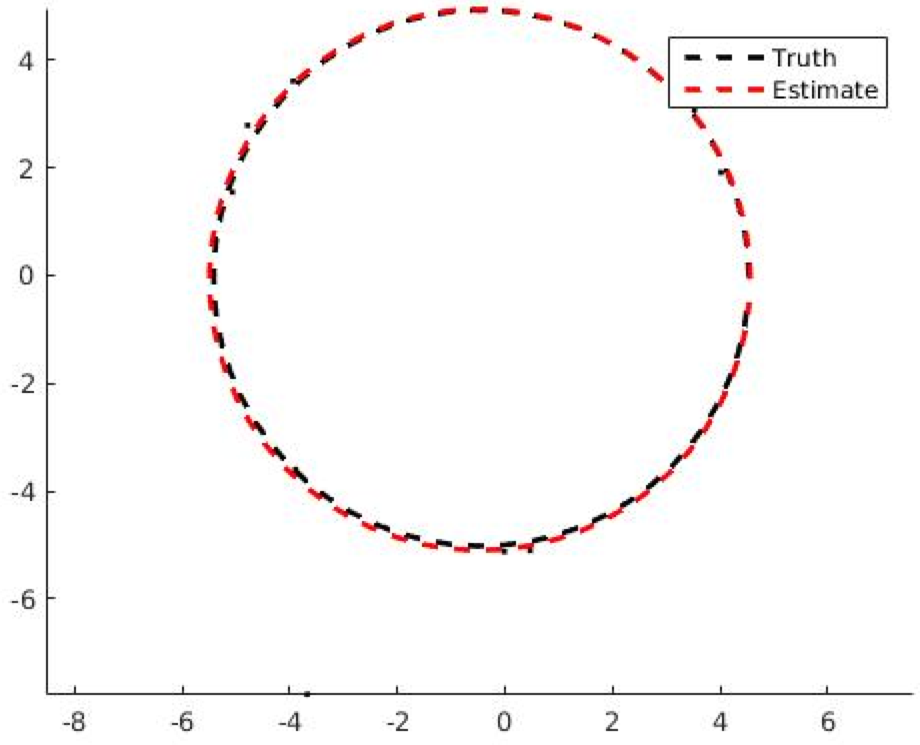


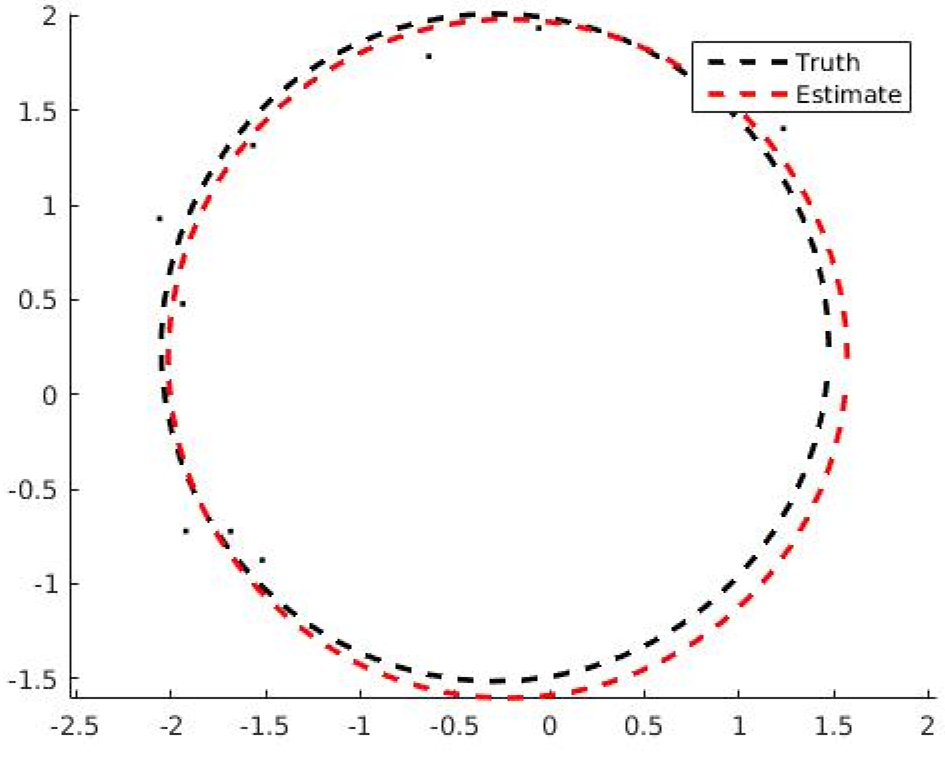
ბ)ransac-ის კოდი:

```
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 abs(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 assicated 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                     %
%                                                          %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

        % Determine which of the points in the non-seed set agree with
        % this model. The nonSeedIsInlier vector should have a 1 (true)
        % when the corresponding point is an inlier, and a 0 when it is not
        % an inlier.
        nonSeedIsInlier = false(nonSeedSetSize, 1);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                                                          %
%               YOUR CODE HERE. Fill in the above variable.          %
%                                                          %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
        for j = 1: size(nonSeedErrors,1)
           nonSeedIsInlier(j) = nonSeedErrors(j)<maxInlierError;
        end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                                                          %
%                        END YOUR CODE                     %
%                                                          %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

        % If at least goodFitThresh points are inliers
        % then the model is good so fit a new model using the seed set and
        % the non-seed inliers.
        if sum(nonSeedIsInlier(:)) >= goodFitThresh
            % Combine the seed set and the non-seed inliers into a single
            % set of inliers.
            % Hint: in MATLAB, you can use an array
            % of true/false values as an "index", and the result will be
            % only the entries where the "index" array = 1. For example,
            % inliers([1 0 1],:) would return an array containing the
            % 1st and 3rd inlier.
            inliers = seedSet;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                                                          %
%                 YOUR CODE HERE. Fill in the above variable.                 %
%                                                          %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
            inliers = [seedSet; nonSeedSet(nonSeedIsInlier,:)];
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                                                          %
%                              END YOUR CODE                              %
%                                                          %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

            % Fit a new model using the inliers.
            xx = 0;
            yy = 0;
            RR = 0;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                                                          %
%                YOUR CODE HERE. Fill in the above 3 variables.               %
%                                                          %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
             [xx, yy, RR] = FitCircle(inliers);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                                                          %
%                              END YOUR CODE                              %
%                                                          %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

            % Compute the total error of the new model on the inliers.
            % There are several ways to define this, but for our purposes,
            % just add up the error at each inlier to produce a total
            % error.
            error = 0;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                                                          %
%                 YOUR CODE HERE. Fill in the above variable.                 %
%                                                                    %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
          error = sum(ComputeErrors(xx,yy,RR,inliers));
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                                                                    %
%                              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

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
```
შესაბამისად მიღებული სურათები:





გ)N=1000-სთვის უკვე ვეღარ პოულობს ხოლმე, ცუდად ეძებს და არც იტერაციების გაზრდა შველის. ამისი წამალია goodFitThresh-ის გაზრდა N/2-მდე, ან ცოტა მეტად. N=600-ზეც მშვენივრად მუშაობს. გამოწვეული იყო იმითი, რომ ამდენი წერტილიდან 5-ეულების ამორჩევა რომ წრეზე იყვნენ ადვილია და ბევრი false positive იყო. N=500-ზე კი აღარა.