```
%==========================
%  Name:                hw1_1.m
%
%  Author:              Kairi Kozuma
%
%==========================
A = [-7,6,-2,-9;-14,-5,-4,0;-12,14,8,-2;10,-14,9,4];
[UU, SS, VV] = svd(A);
fprintf('UU\n');
disp(UU);
fprintf('SS\n');
disp(SS);
fprintf('VV\n');
disp(VV);

fprintf('Singular Values of SS\n');
disp([SS(1,1),SS(2,2),SS(3,3),SS(4,4)]);
fprintf('Last column of VV\n');
disp([VV(1,4),VV(2,4),VV(3,4),VV(4,4)]');
```

```
UU
   -0.4004   -0.0663    0.1975    0.8923
   -0.2428   -0.8664   -0.4293   -0.0783
   -0.6129    0.4856   -0.6148   -0.1029
    0.6365    0.0955   -0.6315    0.4325

SS
   28.4064         0         0         0
         0   14.7480         0         0
         0         0   11.8593         0
         0         0         0    6.5524

VV
    0.7013    0.5235    0.4799    0.0623
   -0.6576    0.6371    0.3006   -0.2670
    0.0914    0.5657   -0.7824    0.2438
    0.2596    0.0005   -0.2592   -0.9303

Singular Values of SS
   28.4064   14.7480   11.8593    6.5524

Last column of VV
    0.0623
   -0.2670
    0.2438
   -0.9303
```

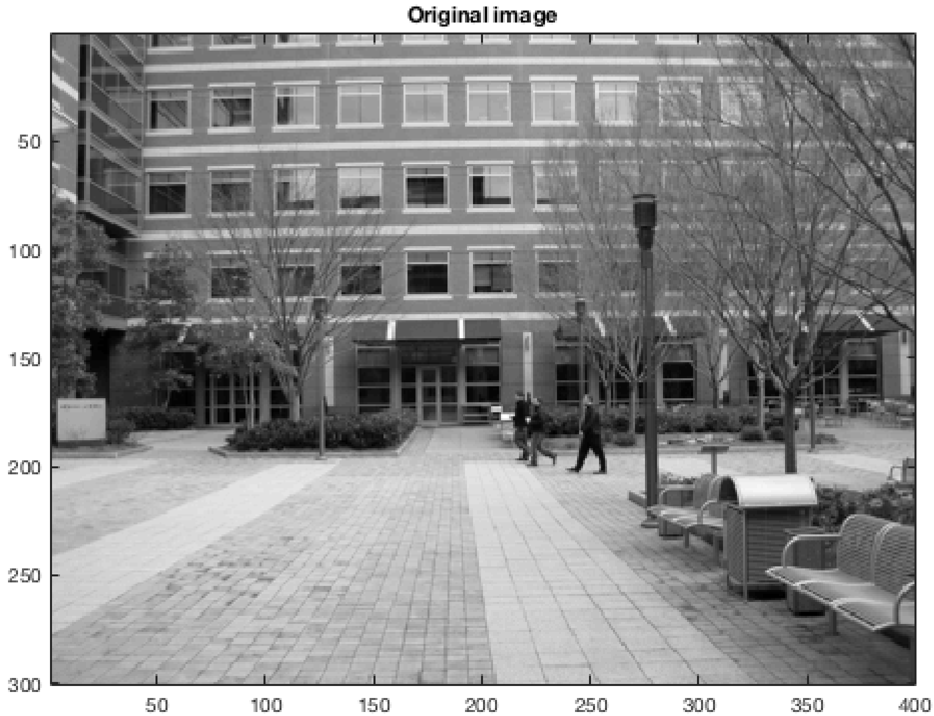
Published with MATLAB® R2016b

```
%==========================
%  Name:                 hw1_2.m
%
%  Author:               Kairi Kozuma
%==========================
q1 = [-36.6; -25.7;  66.0];
q2 = [ 23.1;   0.1;  77.0];
q3 = [-45.0;  49.7; 150.0];
q4 = [-81.0;  89.5; 270.0];

p1 = pcamera(q1);
p2 = pcamera(q2);
p3 = pcamera(q3);
p4 = pcamera(q4);

fprintf('Q1 projects to:\n');
disp(p1);
fprintf('Q2 projects to:\n');
disp(p2);
fprintf('Q3 projects to:\n');
disp(p3);
fprintf('Q4 projects to:\n');
disp(p4);

fprintf('All points project onto image plane if the range is [-0.5,0.5]\n');
fprintf('There are four unique projections. Q3 and Q4 project almost onto the same point.\n');
```

```
Q1 projects to:
   -0.3327   -0.2336

Q2 projects to:
    0.1800    0.0008

Q3 projects to:
   -0.1800    0.1988

Q4 projects to:
   -0.1800    0.1989

All points project onto image plane if the range is [-0.5,0.5]
There are four unique projections. Q3 and Q4 project almost onto the same point.
```

```matlab
%================================ plotEdges ================================
%
%  script plotEdges.m
%
%
%  Loads the edgethresh.mat Matlab file (make sure to have it in your
%  path or your current directory) and then thresholds the edge scores
%  to identify which parts of the image reflect edge-like structures.
%
%================================ plotEdges ================================

%
%  Name:                plotEdges.m
%
%  Author:              Patricio A. Vela,                      pvela@gatech.edu
%
%  Created:             2014/01/13
%  Modified:    2014/01/13
%
%================================ plotEdges ================================

%--[1] Load the edgethresh Matlab file.
load('edgethresh.mat');

%--[2] Apply a threshold to the edge scores to get binary images.
thresh1 = 105.0;
thresh2 = 1.0;

fprintf('Threshold for edge 1: %f\n', thresh1);
fprintf('Threshold for edge 2: %f\n', thresh2);

edge1new = edge1 > thresh1;
edge2new = edge2 > thresh2;
detect1 = edge1new ;
detect2 = edge2new ;

%--[3] Up to you to run or not.  Thin out thick edge zones to give slim line.
detect1 = bwmorph(detect1, 'thin');
detect2 = bwmorph(detect2, 'thin');

%--[4] Plot the image and also visualize the detected edge locations.
figure(1);
  imagesc(I);
  colormap('gray');
  axis image;
  title('Original image');

figure(2);
  imagesc(detect1);
  colormap('gray');
  title('Edge 1');

figure(3);
  imagesc(detect2);
  colormap('gray');
  title('Edge 2');
```

```
Threshold for edge 1: 105.000000
Threshold for edge 2: 1.000000
```

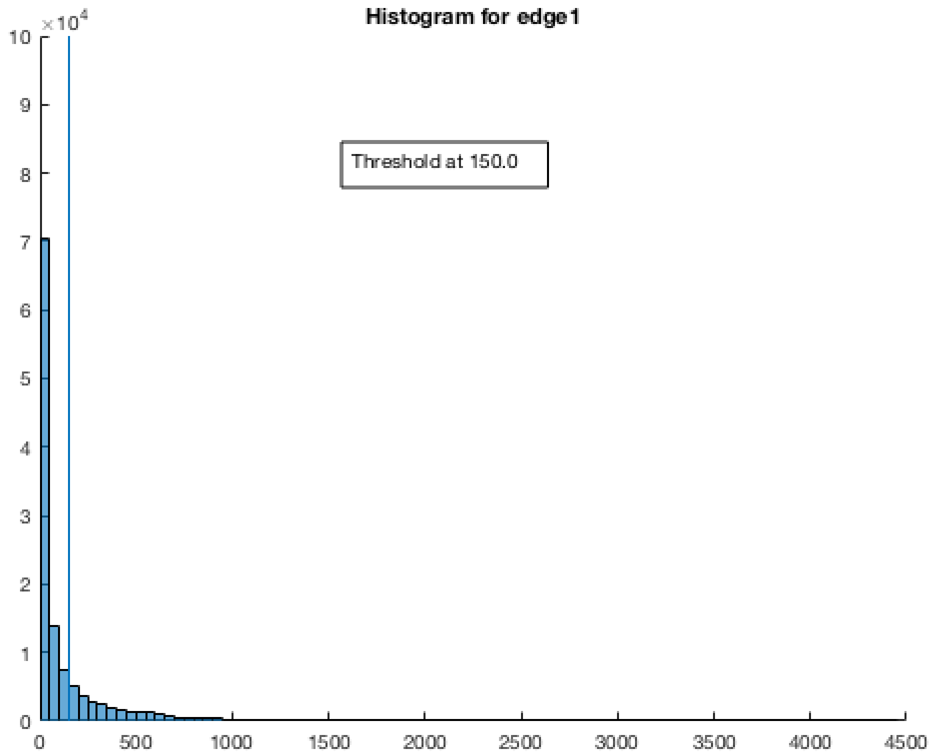


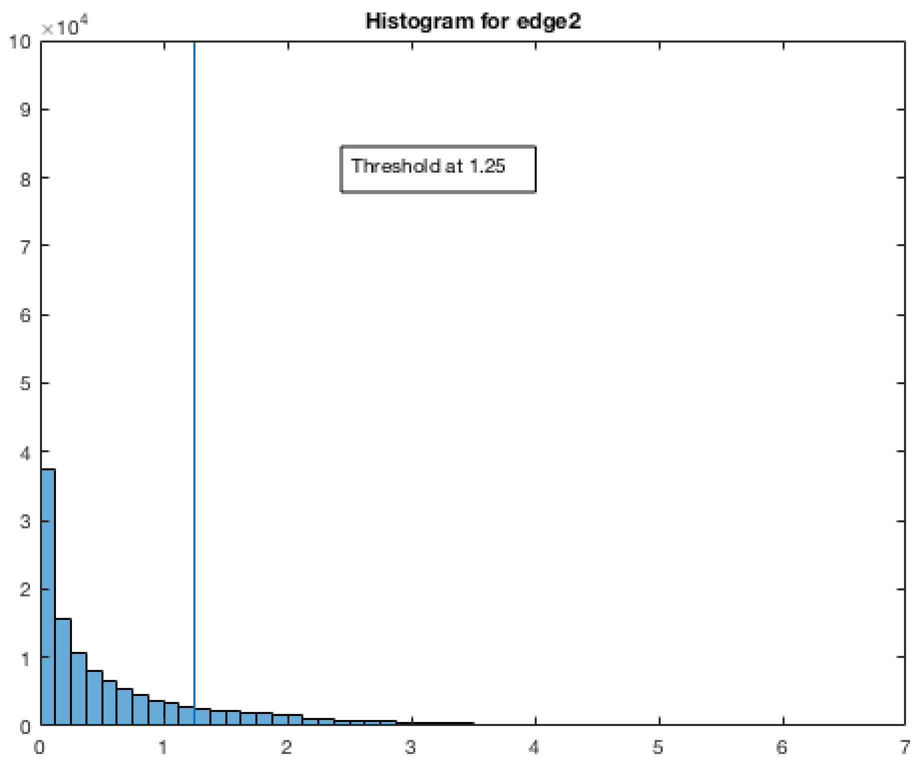
Published with MATLAB® R2016b

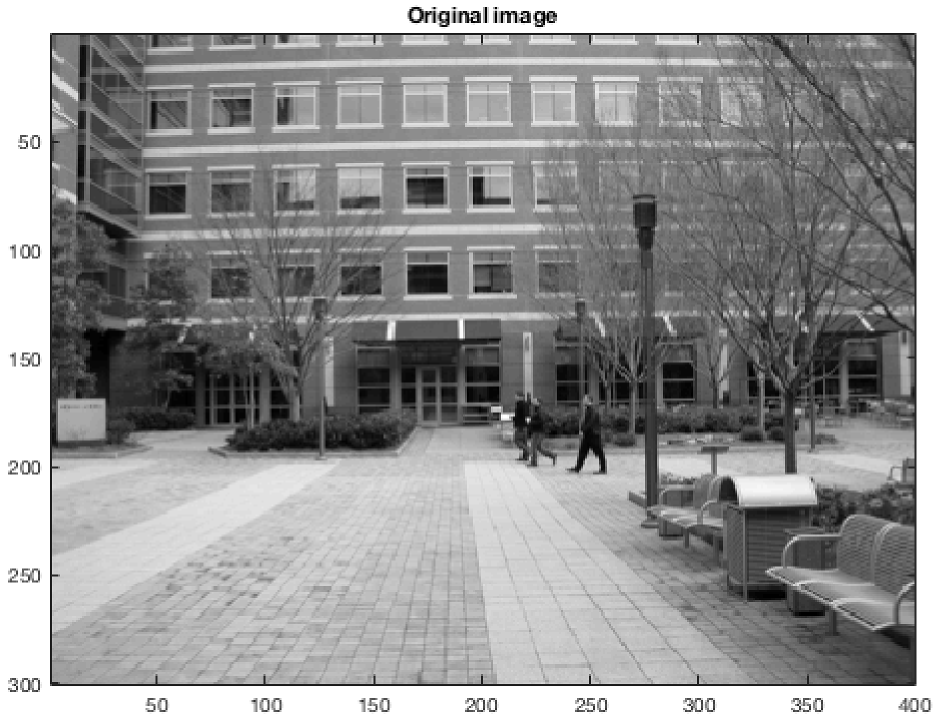
```
%  script histedges.m

%--[1] Load the edgethresh Matlab file.
load('edgethresh.mat');

max1 = max(reshape(edge1, 1, []));
max2 = max(reshape(edge2, 1, []));

figure(1);
hold on
hist1 = histogram(reshape(edge1, 1, []), [0:50:max1]);
line([150,150],[0,100000])
dim = [.4 .5 .3 .3];
annotation('textbox',dim,'String','Threshold at 150.0','FitBoxToText','on');
title('Histogram for edge1');

figure(2);
hist2 = histogram(reshape(edge2, 1, []), [0:0.125:max2]);
line([1.25,1.25],[0,100000])
dim = [.4 .5 .3 .3];
annotation('textbox',dim,'String','Threshold at 1.25','FitBoxToText','on');
title('Histogram for edge2');

%--[2] Apply a threshold to the edge scores to get binary images.
thresh1 = 150.0;
thresh2 = 1.25;

fprintf('Threshold for edge 1: %f\n', thresh1);
fprintf('Threshold for edge 2: %f\n', thresh2);

edge1new = edge1 > thresh1;
edge2new = edge2 > thresh2;
detect1 = edge1new ;
detect2 = edge2new ;

%--[3] Up to you to run or not.  Thin out thick edge zones to give slim line.
detect1 = bwmorph(detect1, 'thin');
detect2 = bwmorph(detect2, 'thin');

%--[4] Plot the image and also visualize the detected edge locations.
figure(3);
  imagesc(I);
  colormap('gray');
  axis image;
  title('Original image');

figure(4);
  imagesc(detect1);
  colormap('gray');
  title('Edge 1');

figure(5);
  imagesc(detect2);
  colormap('gray');
  title('Edge 2');

% The threshold for edge1 produced an image that was less satisfactory than
% that produced by edge2. Although objects such as signs and benches appear
% with more detail in the binary image, less visible edges on the ground
```

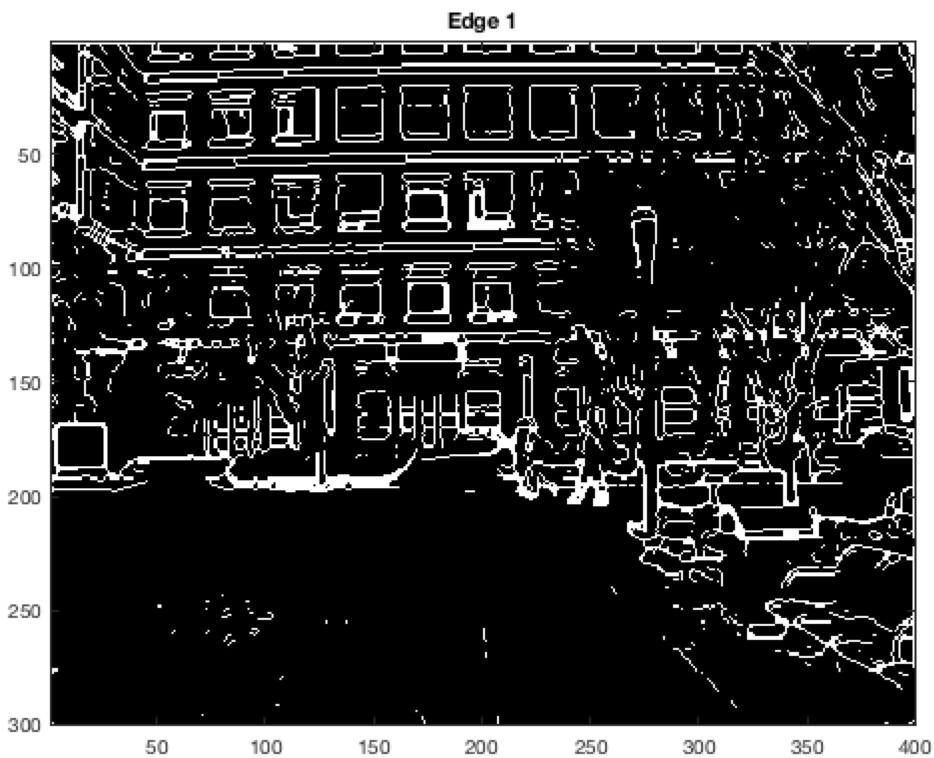
```
% are completely missing.

% The threshold for edge2 aligned more with what I envisioned, with most of
% the windows outlined and objects such as people, signs, and benches
% recognizable in the binary image.
```

```
Threshold for edge 1: 150.000000
Threshold for edge 2: 1.250000
```


Histogram for edge1
×10^4
Threshold at 150.0
0
500
1000
1500
2000
2500
3000
3500
4000
4500


Histogram for edge2
×10^4
Threshold at 1.25


Original image
50
100
150
200
250
300
50
100
150
200
250
300
350
400


Edge 1
50
100
150
200
250
300
50
100
150
200
250
300
350
400

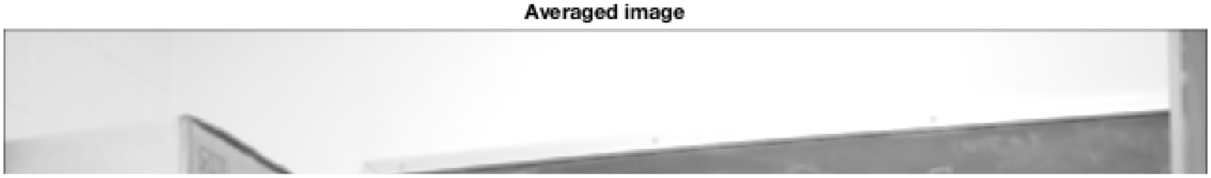

Edge 2
50
100
150
200
250
300
50
100
150
200
250
300
350
400

```matlab
%===========================
%  Name:                 hw1_4.m
%
%  Author:               Kairi Kozuma
%
%===========================

% Average filter
n = 5;
aFilt = ones(n)/(n^2);

% Filter the image
campus3filt = uint8(imfilter(double(campus3), aFilt));

% Show image before and after filtering
figure(1);
imshow(campus3);
title('Original image');

figure(2);
imshow(campus3filt);
title('Averaged image');
dim = [.4201, .55729, .090056, .014478];
annotation('rectangle',dim,'Color','red')
dim = [.4321, .62429, .090056, .014478];
str = 'Illegible';
annotation('textbox',dim,'String',str,'FitBoxToText','on');

dim = [.276, .20061, .043167, .14105];
annotation('rectangle',dim,'Color','red')
dim = [.306, .28061, .043167, .14105];
str = 'Still legible';
annotation('textbox',dim,'String',str,'FitBoxToText','on');


% a) The small office depot label on the box becomes unreadable. The large
% DELL sign on the box closest to the camera is still visible. The area
% passed through OK because the letters were large enough so that smoothing
% the edges did not render the letters illegible. A greater neighborhood
% size would make all letters unreadable.

% b) A sharpening filter is another type of convolution kernel. The
% sharpening filter uses the matrix:
%               [0  -1  0]
%               [-1  5 -1]
%               [0  -1  0]
% This has the effect of emphasizing disparity between adjacent pixel
% values, so that small differences in the image become more apparent.
```

```
Warning: Image is too big to fit on screen; displaying at 67%
Warning: Image is too big to fit on screen; displaying at 67%
```

Original image

Averaged image




Illegible
Still legible