Lab 2 Part 2

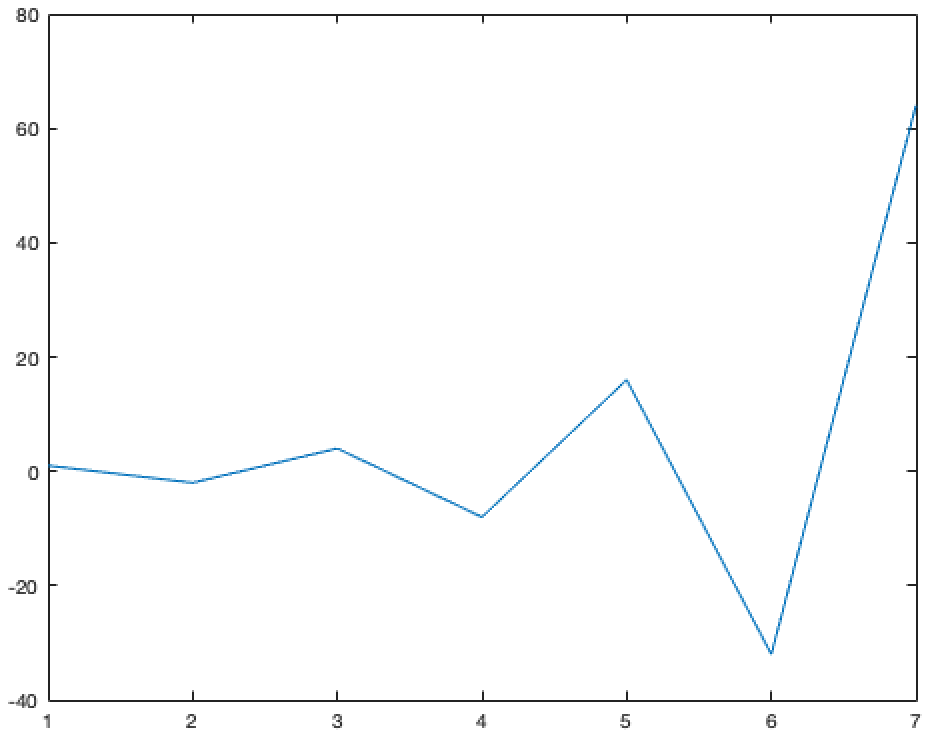
Table of Contents

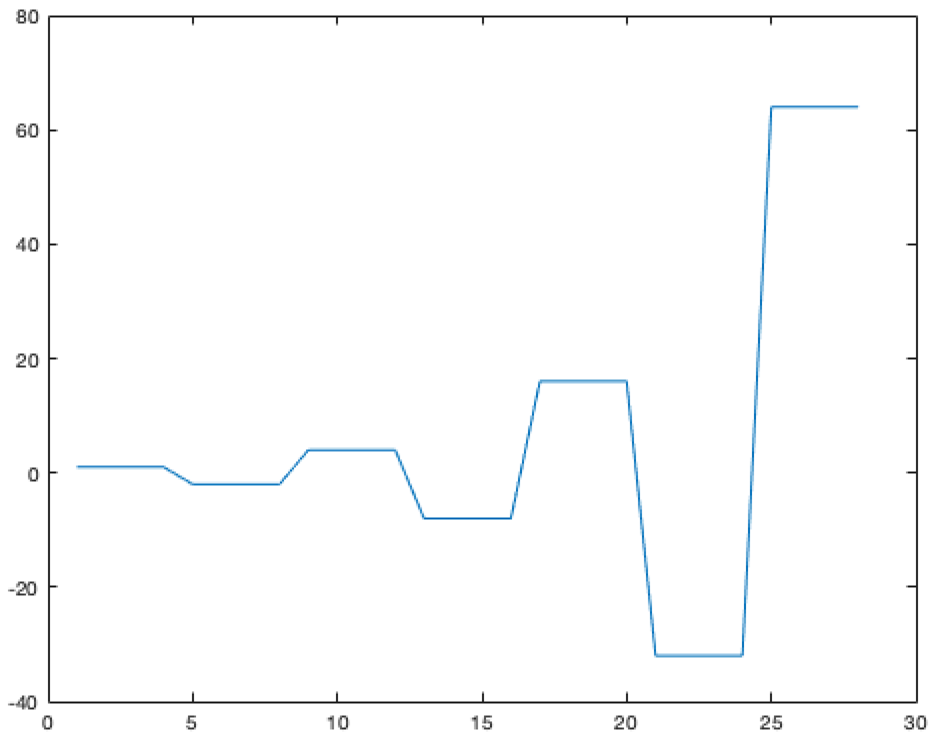
2.1 - Image Reconstruction from Downsampling

```
cl
```

1. Introduction to Interpolation

```
row = (-2).^(0:6);
L = length(row);
nn= ceil((0.999:1:4*L)/4); %<---Round up to the integer
hold1 = row(nn);
plot(row);
figure;
plot(hold1);
```





Question 1: What values are in the indexing vector nn , and why are they what they are?

nn contains the indexes of row that are used to generate the 0th order hold of the signal. Each indexes is repeated 4 times, and as a result, each value in row is repeated 4 times in hold1.

2. Interpolate a Lighthouse

a.

```
load lighthouse.mat
```

b.

```
x3 = xx(1:3:end,1:3:end);
show_img(x3);
```

```
Image being scaled so that min value is 0 and max value is 255
```

c.

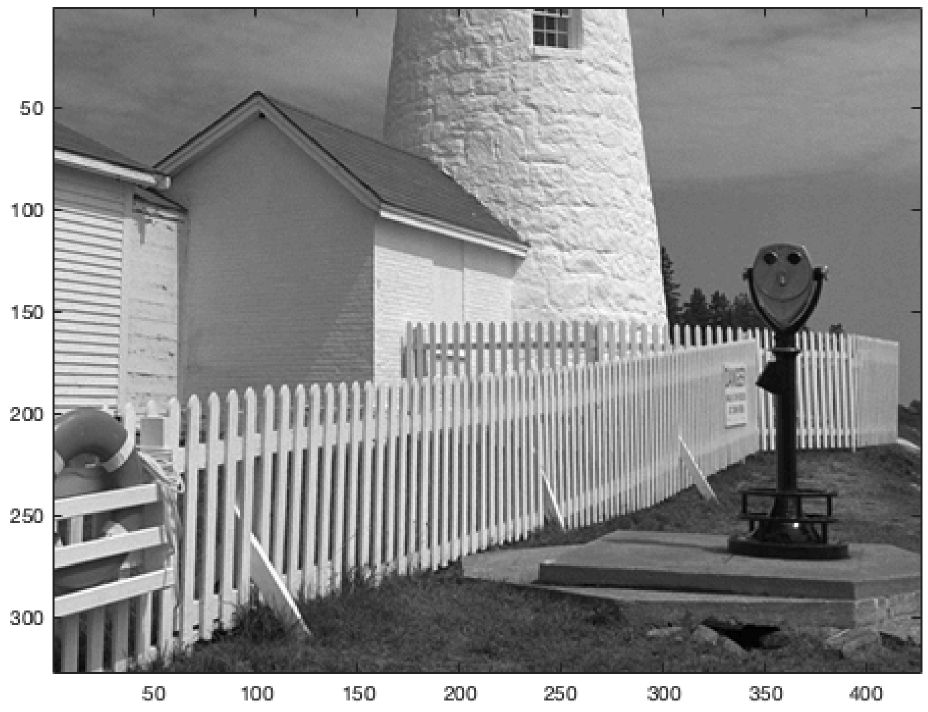
```
[rows, cols] = size(x3);
xrows = x3(ceil(1/3:1/3:rows), :);
```

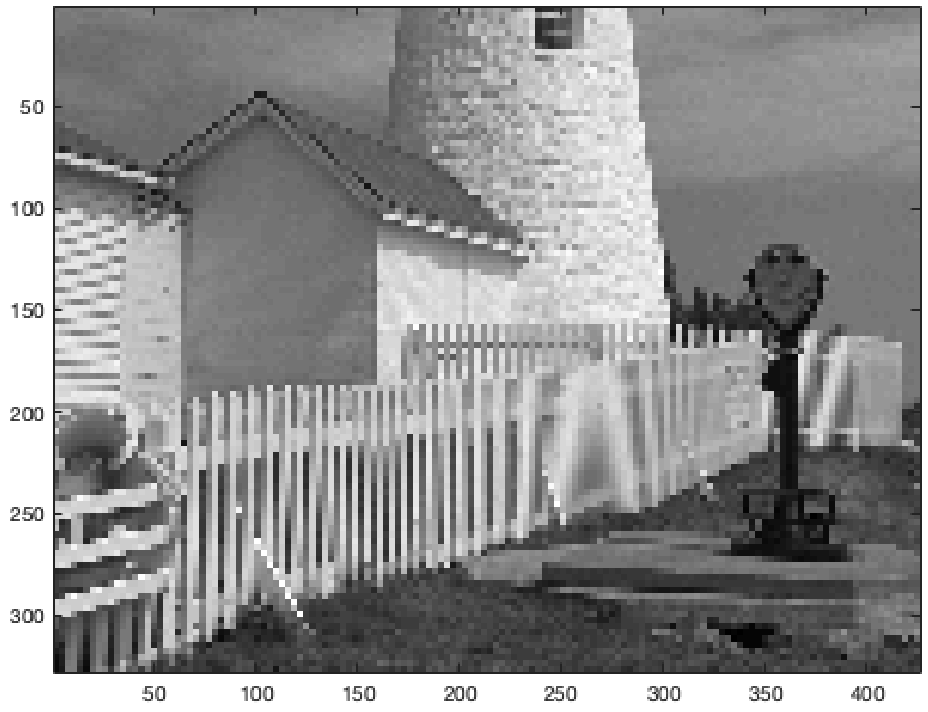
Question 2: What are the dimensions of the rows and columns of xrows?

```
size(xrows)


ans =

   327   142
```

d.

```
% downsample by a factor of 3 and compare the images
xhold = xrows(:, ceil(1/3:1/3:cols));
show_img(xx, 99);
show_img(xhold, 100);

Image being scaled so that min value is 0 and max value is 255
Image being scaled so that min value is 0 and max value is 255
```



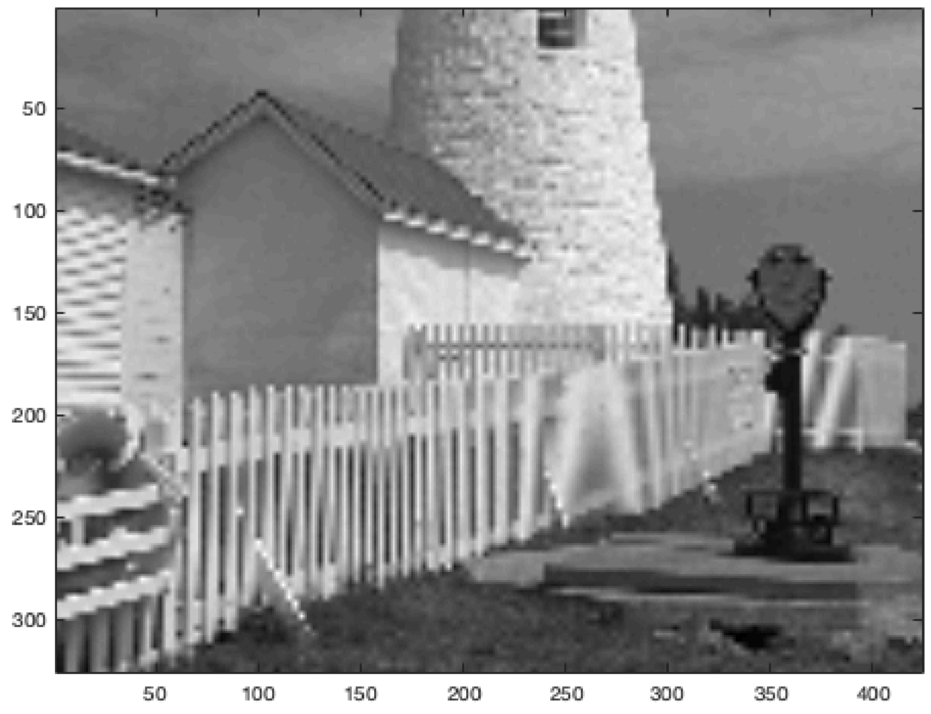


Question 3: Compare them and explain any differences that you can see.

The quality on this image is not very good, with little difference in quality between this image and the downsampled version (outside of restoring the original image size). This is expected from 0th order hold, as all we are doing is repeating data.

e.

```
[rows, cols] = size(x3);
new_rows = 3*rows-3+1;
new_cols = 3*cols-3+1;

% xrows is a temporary container for the row interpreted image, and
 xlin
% will contain the fully lineraly interpolated and restored image
xrows = zeros(rows, new_cols);
xlin = zeros(new_rows, new_cols);

% perform row interpolationn
for i = 1:rows
    xrows(i,:) = interp1(1:size(x3,2),x3(i,:),1:1/3:size(x3,2));
end
% perform col interpolation on the row interpolated image
for j = 1:new_cols
    xlin(:,j) =
 interp1(1:size(xrows,1),xrows(:,j),1:1/3:size(xrows,1));
end

show_img(xlin);

Image being scaled so that min value is 0 and max value is 255
```



f.

original image

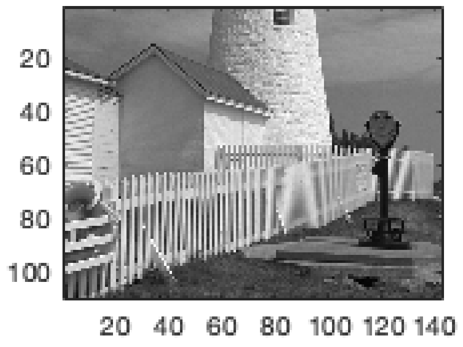
```
show_img(xx);
```

```
Image being scaled so that min value is 0 and max value is 255
```

down-sampled image

```
show_img(x3);
```

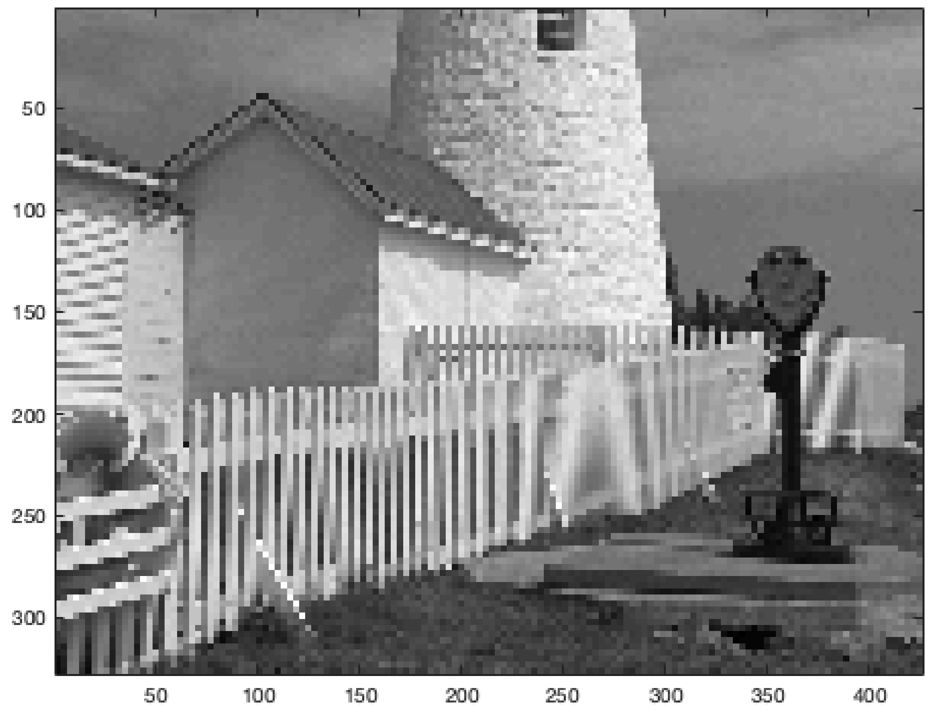
Image being scaled so that min value is 0 and max value is 255



zero-order-hold reconstructed image

```
show_img(xhold);
```

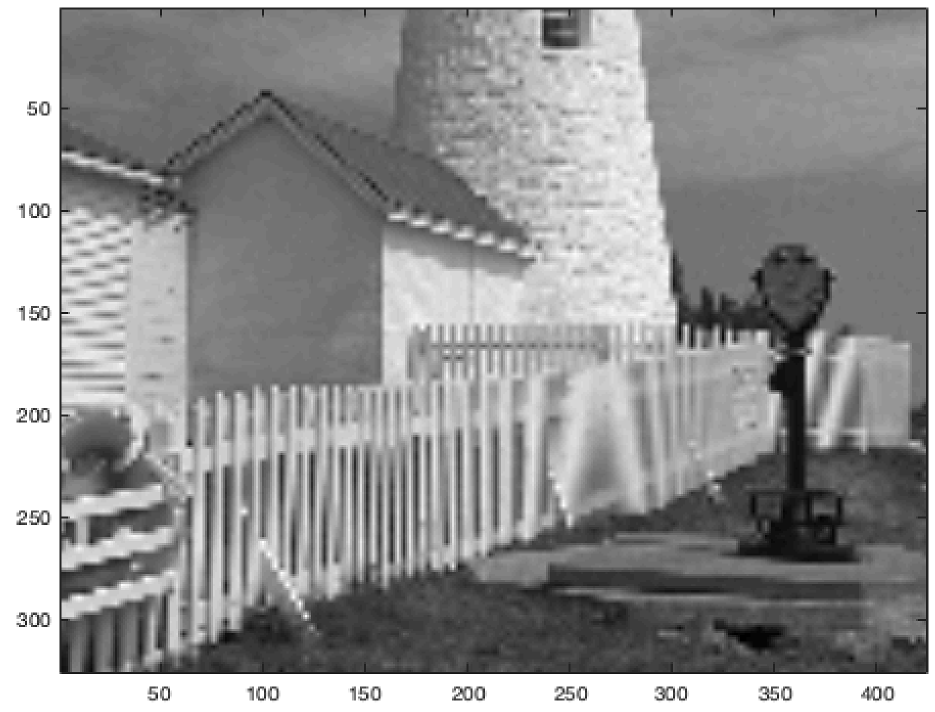
Image being scaled so that min value is 0 and max value is 255



linearly-interpolated reconstructed image

```
show_img(xlin);
```

```
Image being scaled so that min value is 0 and max value is 255
```



Point out their differences and similarities.

These images all depict the lighthouse with the fence and the viewing post. The first image is the original image, followed by the down sampled image, then the zero-order-hold reconstructed image and then the linearly interpolated reconstructed image. I will compare the last three images to the first, normal image. The down sampled image is much smaller than the original image, as a result of the downsampling process. It is much lower in quality as well, and appears grainy. While the 0th-order-hold image is the same size as the original image, it lacks the quality seen in the original image and retains the graininess seen in the down sampled image. In addition to the graininess, aliasing is apparent in the areas of the photo where there is a high color change frequency, such as the fence. The linearly interpolated photo produced a better reconstruction than the 0th order hold, but was still nowhere close to the quality that was found in the original photo. Aliasing is still apparent, but the graininess has been reduced.

Question 4: Can the linear interpolation reconstruction process remove the aliasing effects from the down-sampled lighthouse image?

No, the linear interpolation recostruction process is not able to remove the aliasing effects seen.

Question 5: Can the zero-order hold reconstruction process remove the aliasing effects from the down-sampled lighthouse image?

The zero-order hold reconstruction process is not able to remove the aliasing effects seen. It performed worse than the linear interpolation reconstruction process.

Question 6: Point out regions where the linear and zero-order reconstruction result images differ and try to justify this difference in terms of the frequency content in that area of the image. In other words, look for regions of "low-frequency" and "high-frequency" content in the image and explain how the interpolation quality is dependent on this factor.

The regions where the linear and zero-order hold recostruction result images differ are on the areas that have low frequency. The linear reconstruction performs better than the zero-order hold in such areas. In areas of high frequency, the content looks very similar in both the zero-order and the linear reconstruction.

Question 7: Are edges low-frequency or high-frequency features? Is the series of fence posts a low-frequency or high-frequency feature? Is the background a low-frequency or high-frequency feature?

Edges are a high-frequency feature, as the color changes very fast on the border of the edge. The series of fence posts are a high-frequency feature, as you can see aliasing clearly and the color changes rapidly. The background (the sky) is an example of a low-frequency feature, with the color staying relatively consistent and aliasing nowhere to be seen.

2.2 - Filtering Images

```
cl
```

1. 2-D Convolution

a.

```
load echart

% convolve the rows first
b = [1, -1];
echart_row_convolve = conv2(echart, b);
```

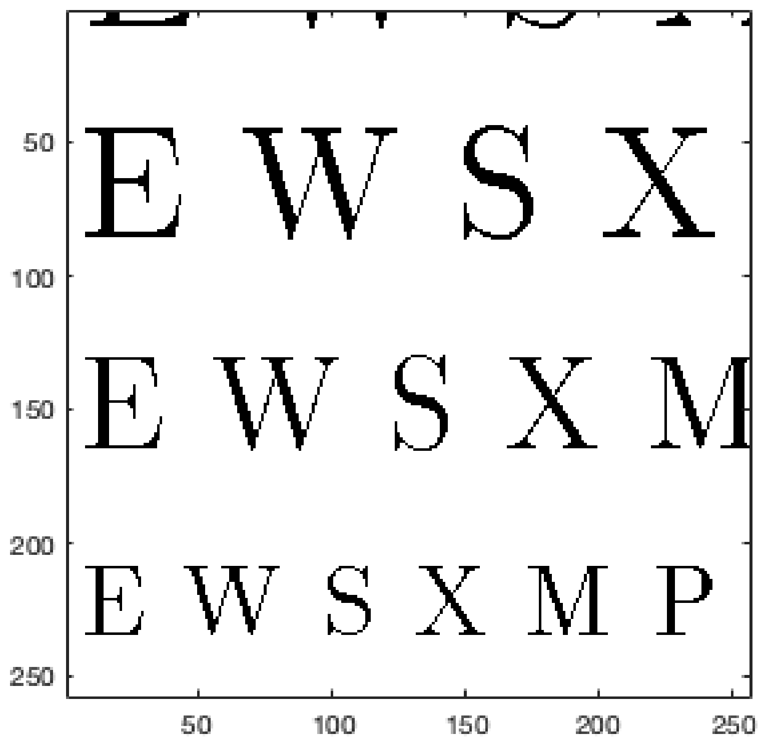
b.

```
% apply transposes to connvolve the columns and transpose back
y = conv2(echart_row_convolve',b')';
```

c.

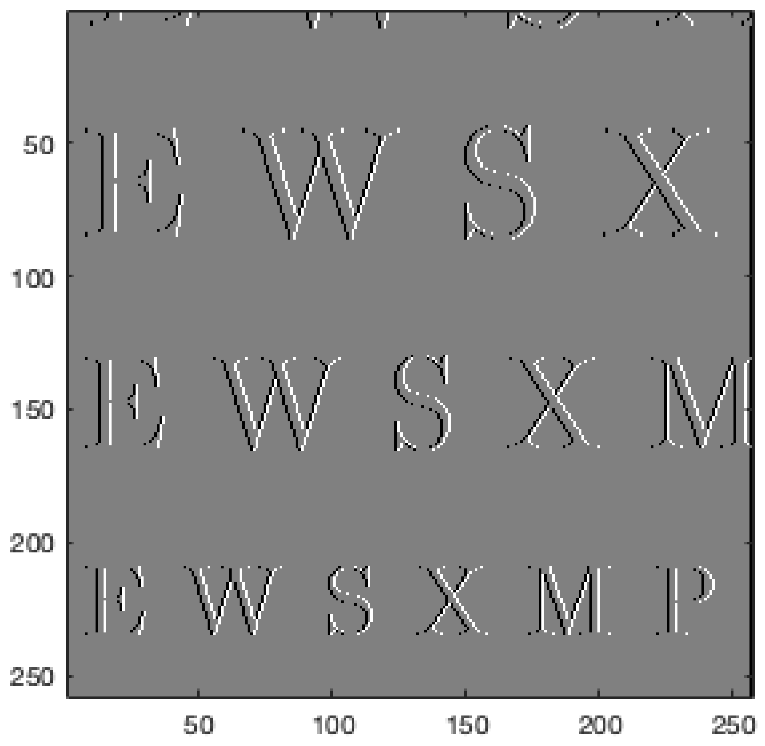
```
show_img(echart);
```

```
Image being scaled so that min value is 0 and max value is 255
```



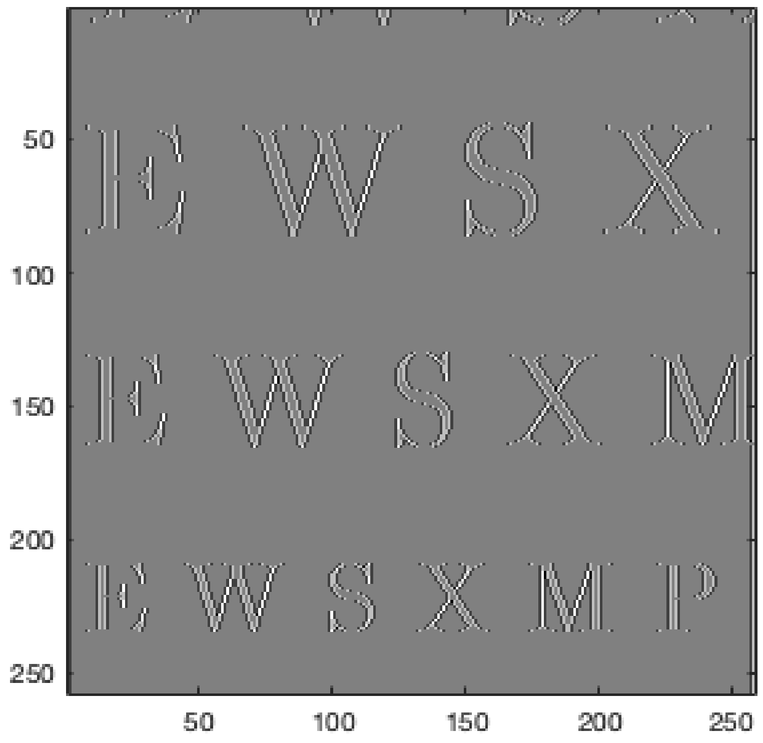
```
show_img(echart_row_convolve);
```

Image being scaled so that min value is 0 and max value is 255



```
show_img(y);
```

Image being scaled so that min value is 0 and max value is 255



Question 8: Compare the three images and give a qualitative description of what you see.

The first image is the unaltered echart. The second image is the row convolved version of the echart. The third image is the column convolved version of the row convolved echart. The second image has been dramatically altered from the convolving process, and now features a predominantly grey background. The letters appear almost embossed. The transformation to the third image is not quite as dramatic, but differences from the second image are still aparrent. More grey is present in this rendition.

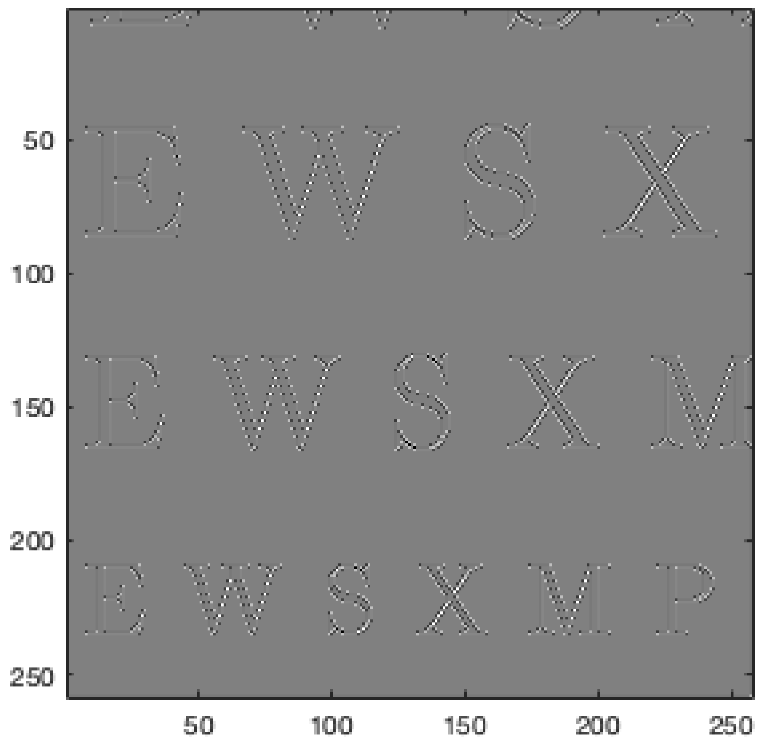
2. Distorting and Restoring Images

a.

```
load echart

% convolve the image
q = 0.9;
bk = [1, -q];
echo90 = conv2(conv2(echart,bk),bk');

show_img(echo90)
```

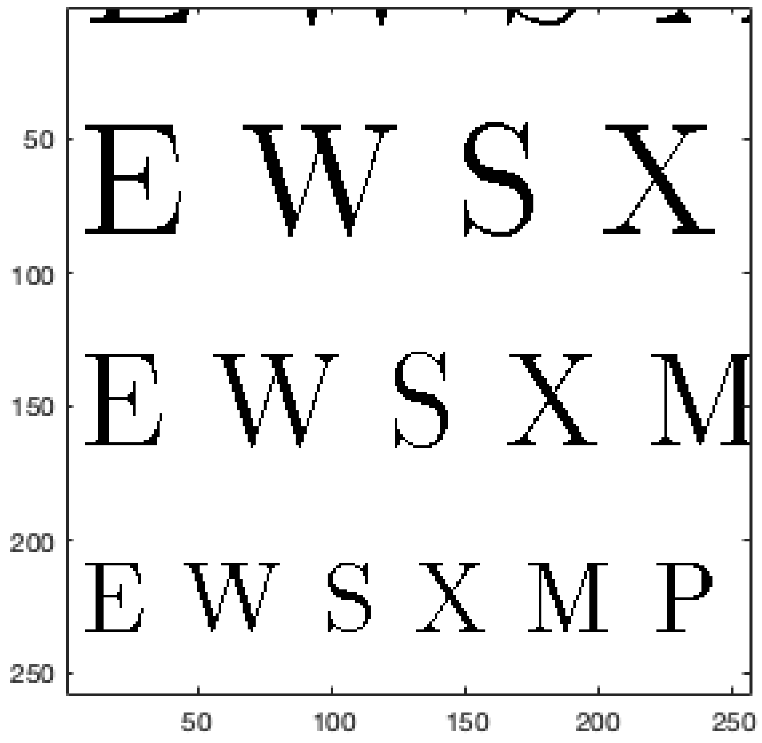
```
Image being scaled so that min value is 0 and max value is 255

ans =

  Axes with properties:
```

```
             XLim: [0.5000 257.5000]
             YLim: [0.5000 258.5000]
           XScale: 'linear'
           YScale: 'linear'
    GridLineStyle: '-'
         Position: [0.1305 0.1106 0.7741 0.8139]
            Units: 'normalized'

  Use GET to show all properties
```



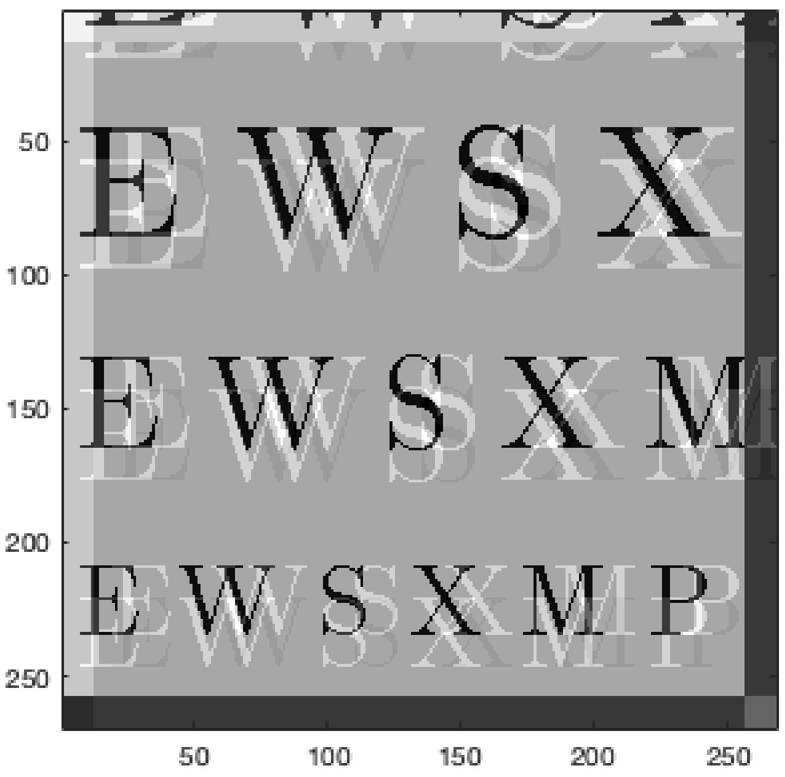
b.

```
% restore the image
M = 11;
r = .9;
bk = r.^(0:M);

restored = conv2(conv2(echo90,bk),bk');

show_img(echart);

Image being scaled so that min value is 0 and max value is 255
```



```
show_img(restored);
```

Image being scaled so that min value is 0 and max value is 255



Question 9: Describe the visual appearance of the output qualitatively, showing the image, and explain its features by invoking your mathematical understanding of the cascade filtering process and why you see ?ghosts? in the output image.

The output image appears very similar to the input image, but with the additional presence of "ghosts" around the letters. The color, position, and size of the actual letters that were present in the input image are exactly the same as the letters in the output image.

Question 10: Use some previous calculations to determine the size and location of the ?ghosts? relative to the original image.

The ghosts are the same size as the original letters. They are offset to the left, down, and diagonally (downn and left) by roughly 22 pixels for each direction. I believe that this is proportional to the size of the filter (M = 22 in this example).

c.

Question 11: Evaluate the worst-case error in order to say how big the ghosts are relative to "black-white" transitions which are 0 to 255. Make sure to show any code you used or plots to further your evaluation.

The ghosts are at a 1 to 1 scale with the original letters, meaning that they are the same size as the original letters.

2.3 Edge Detection and Reconstruction

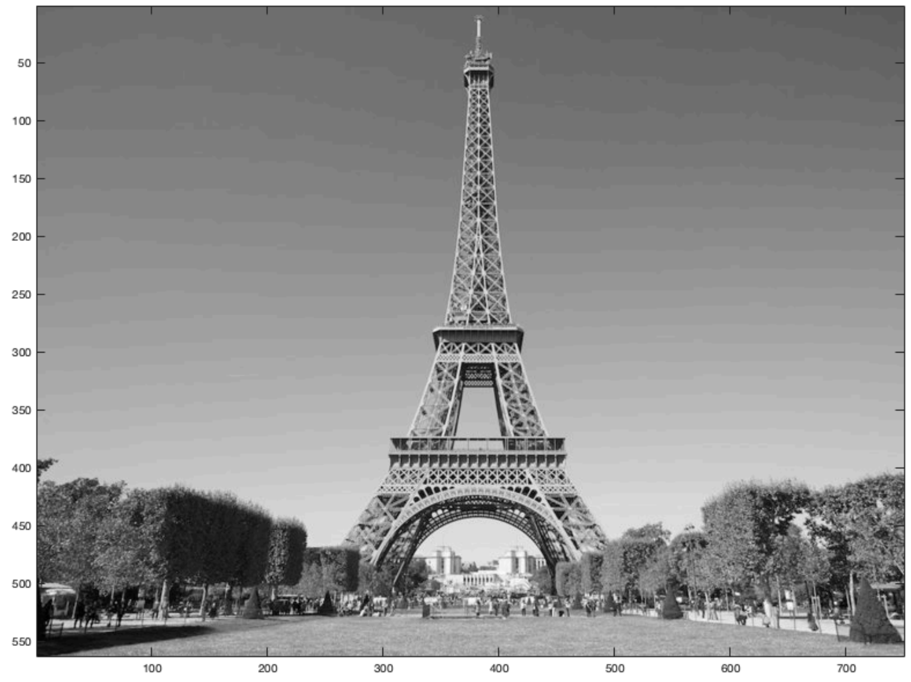
```
cl
```

1. Image Blurring

a.

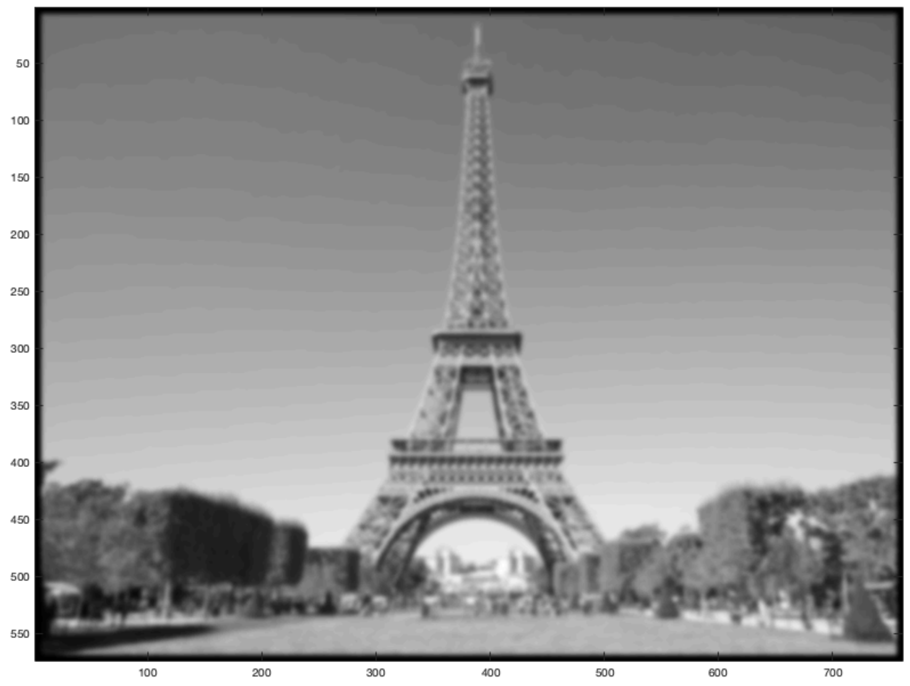
```
load tower
show_img(xx);
```

Image being scaled so that min value is 0 and max value is 255



```
% apply the kernel to apply the blur to the tower
blurred_tower = conv2(xx, kernel);
show_img(blurred_tower);
```

Image being scaled so that min value is 0 and max value is 255

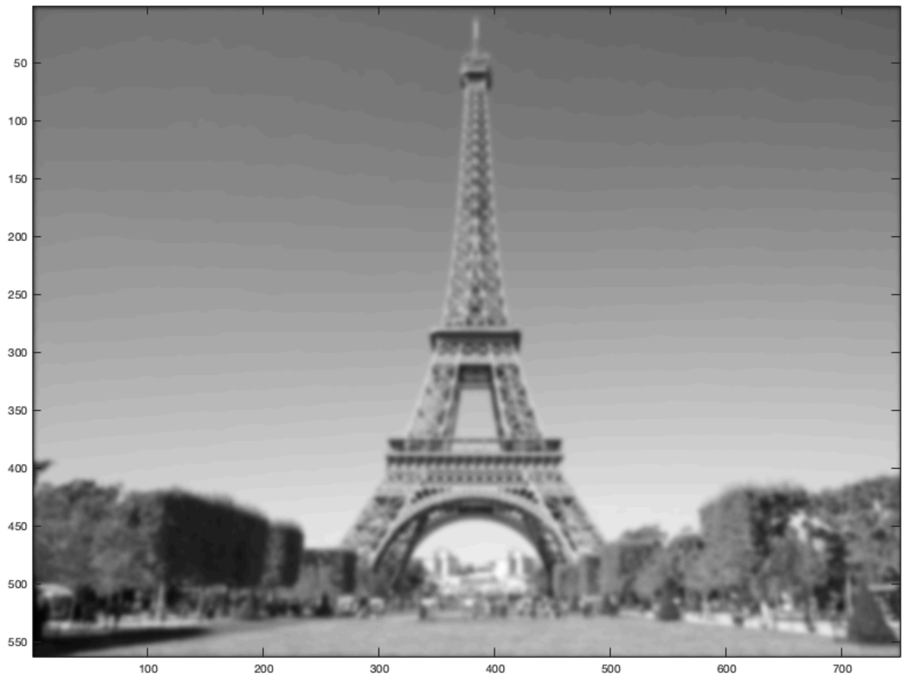


Show and describe the result.

The result of this blur is the same as the original image, but with a blur applied to the entirety of the image. There is also a black border on the edge of this image on all sides.

```
type remove_border
blurred_tower_no_border = remove_border(xx, blurred_tower);
show_img(blurred_tower_no_border);

% verify that the blurred tower with no border has the same dimensions
 as
% the original tower.
size(xx) == size(blurred_tower_no_border)


function [blurred_image_no_border] = remove_border(orig_image,
 blurred_image)
    % REMOVE_BORDER Removes the black border that was applied by a
 gaussian
    % kernel convolution operation and restores the image to the
 original size.
    %   Removes the half the difference between the old border size
 and the
    %   new border size along each border.

    % store the row and column sizes of the different images in
 appropriate
    % data containers
```

```
   [orig_row_size, orig_col_size] = size(orig_image);
   [blurred_row_size, blurred_col_size] = size(blurred_image);

   % calculate where the new image should start and end for each row
 and
   % column
   row_start = ceil( (blurred_row_size-orig_row_size)/2 );
   row_end = blurred_row_size - ceil( (blurred_row_size -
 orig_row_size)/2 );
   col_start = ceil( (blurred_col_size - orig_col_size)/2 );
   col_end = blurred_col_size - ceil( (blurred_col_size -
 orig_col_size)/2 );

   % generate and return the borderless image
   blurred_image_no_border = blurred_image(row_start:row_end,
 col_start:col_end);
end

Image being scaled so that min value is 0 and max value is 255

ans =

  1×2 logical array

   1   1
```



2. Edge Detection

```
cl
```

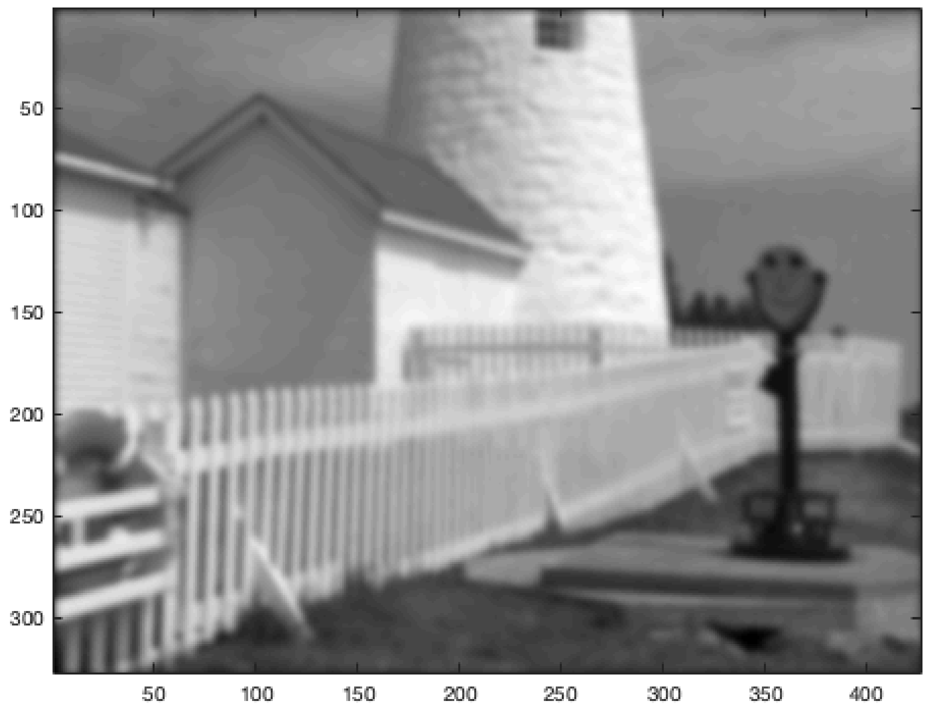
a. Create and show an image that only contains the high-frequency components of the image (think: the original image has both high and low-frequency components, and the blurred image only has low-frequency components).

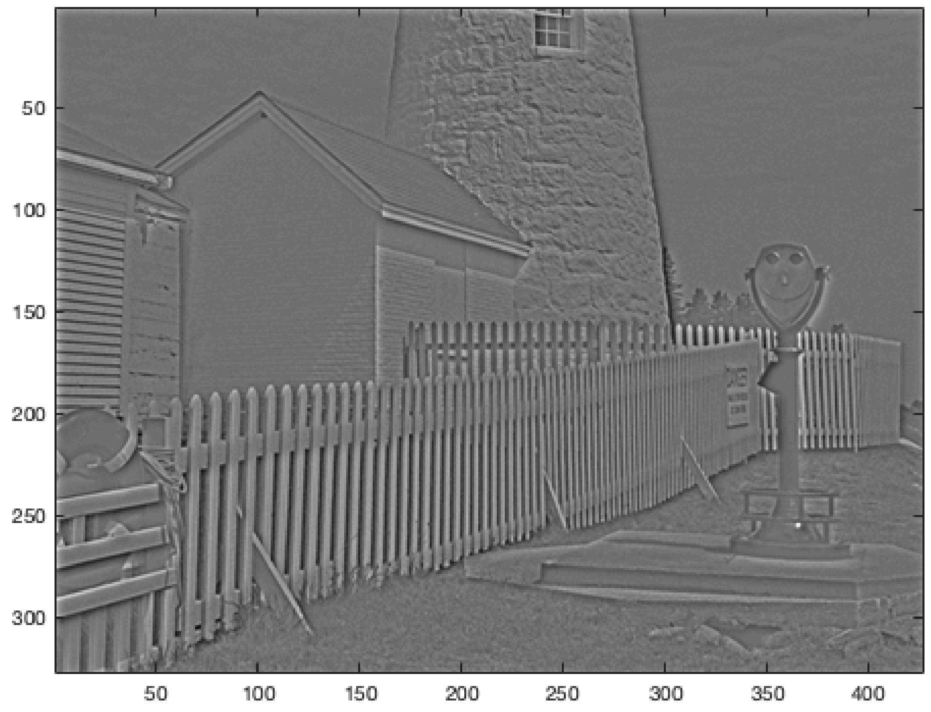
```
load tower
load lighthouse.mat
```

a.

```
% generate a blurred lighthouse with no border
lighthouse = xx;
blurred_lighthouse = conv2(lighthouse, kernel);
blurred_no_border = remove_border(lighthouse, blurred_lighthouse);
show_img(blurred_no_border);

% remove the low frequecy from the lighthouse by subtracting off the
% blurred lighthouse with no border
high_freq_lighthouse = lighthouse - blurred_no_border;

show_img(high_freq_lighthouse);
```

```
Image being scaled so that min value is 0 and max value is 255
Image being scaled so that min value is 0 and max value is 255
```

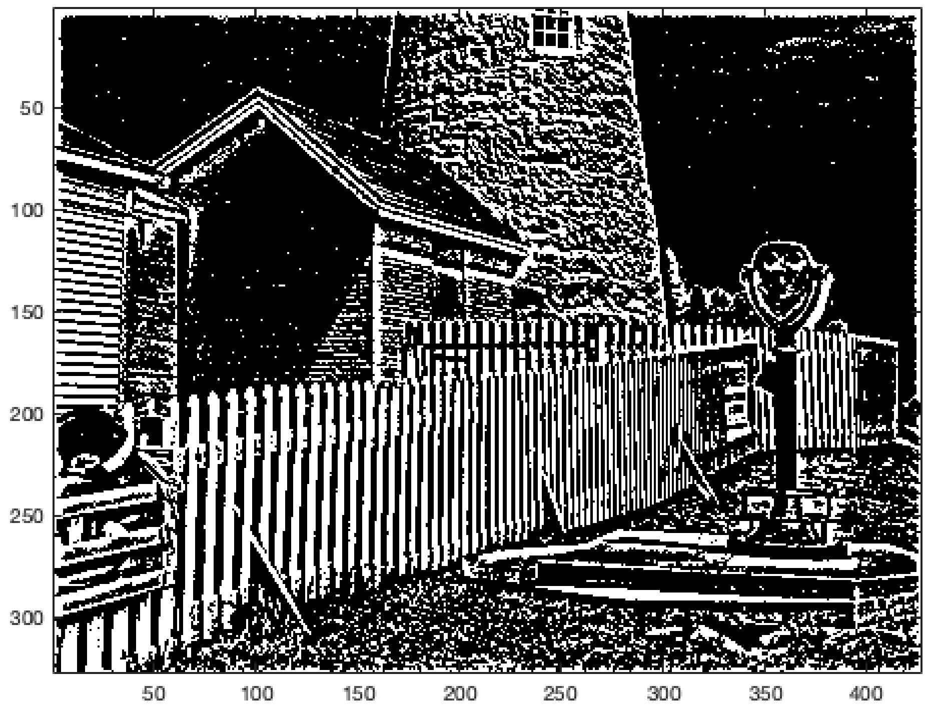




b.

```
thresholded_lighthouse = high_freq_lighthouse >= 0.025;
show_img(thresholded_lighthouse);
```

Image being scaled so that min value is 0 and max value is 255



```
show_img(lighthouse);
```

Image being scaled so that min value is 0 and max value is 255

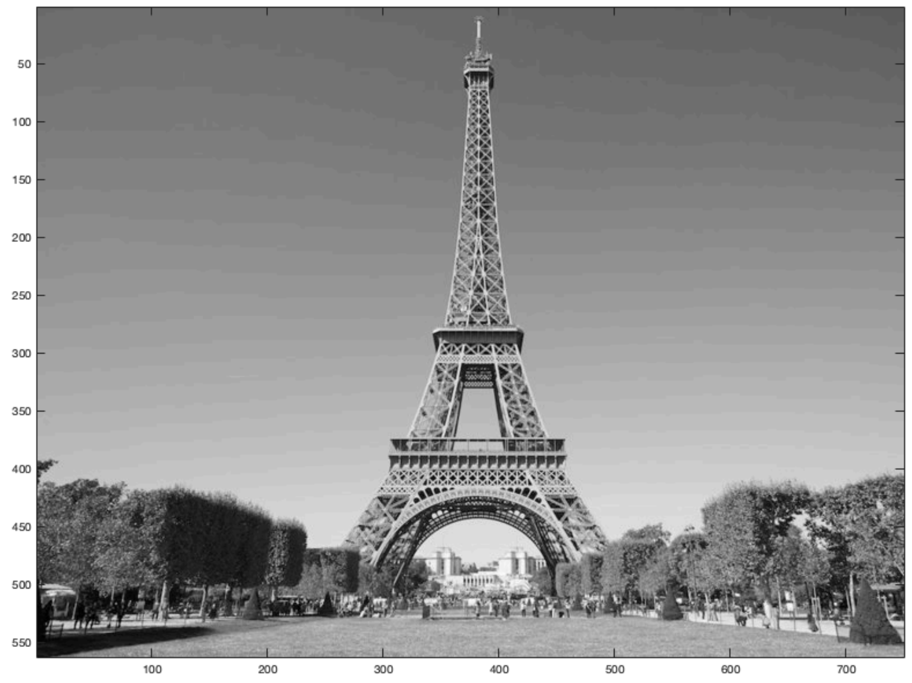
3. Image Restoration

```
cl
```

a.

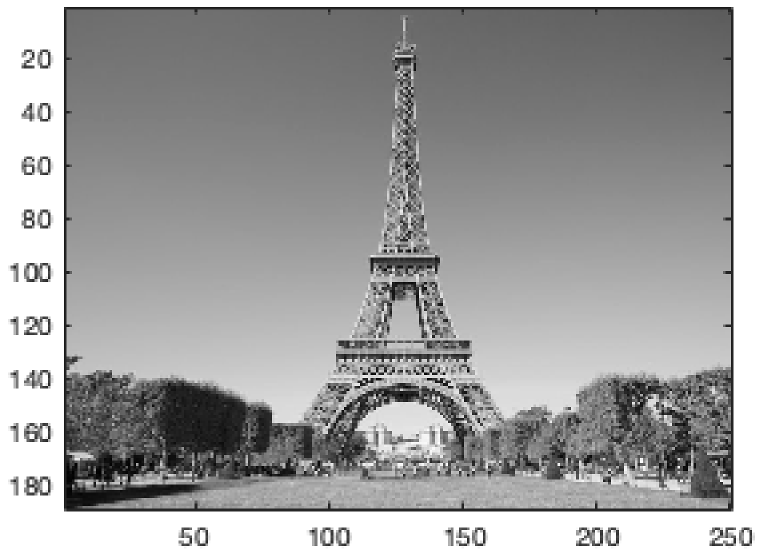
```
load tower
tower = xx;

show_img(tower);
```

```
Image being scaled so that min value is 0 and max value is 255
```



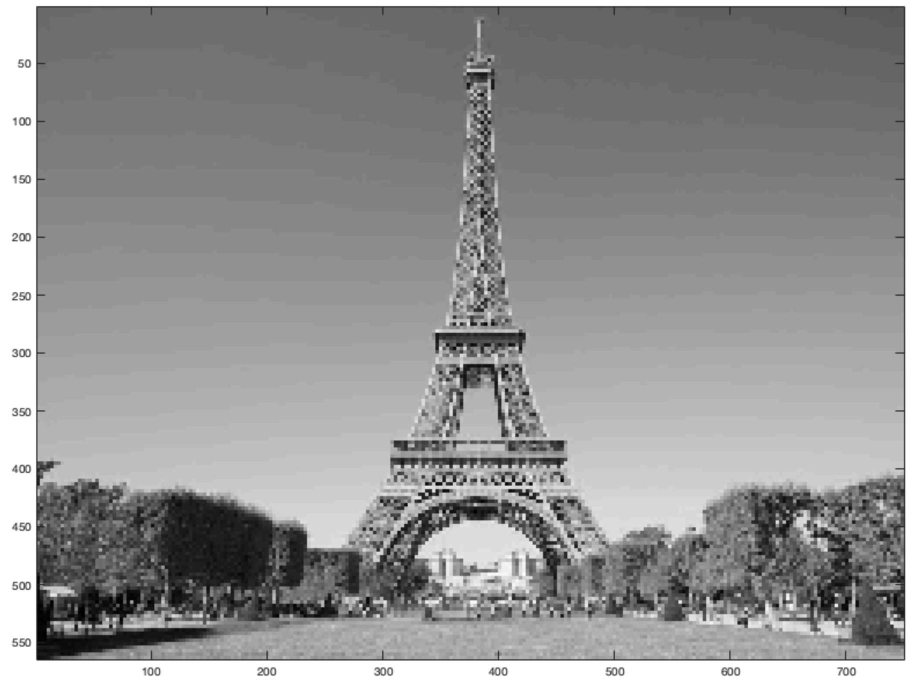
```
d = 3;

% downsample the tower and show it
tower_downsampled = tower(1:d:end,1:d:end);
show_img(tower_downsampled);
```

```
Image being scaled so that min value is 0 and max value is 255
```



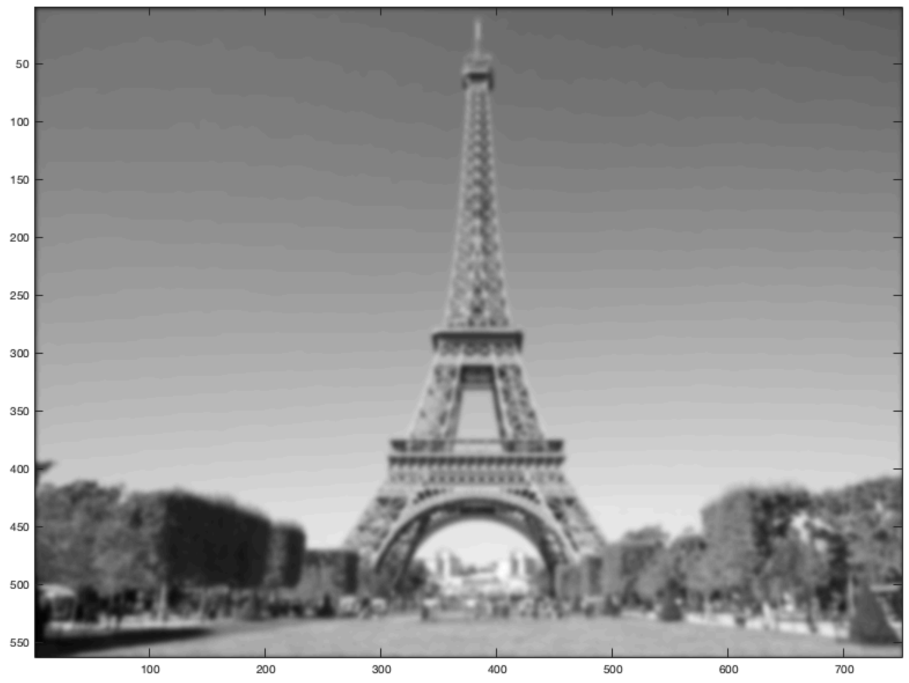
```
% interpolate by a factor of 3 and show it
[row_nums, col_nums] = size(tower_downsampled);
rows = tower_downsampled(ceil(1/d:1/d:row_nums), :);
tower_hold = rows(:, ceil(1/d:1/d:col_nums));
show_img(tower_hold);
```

```
Image being scaled so that min value is 0 and max value is 255
```



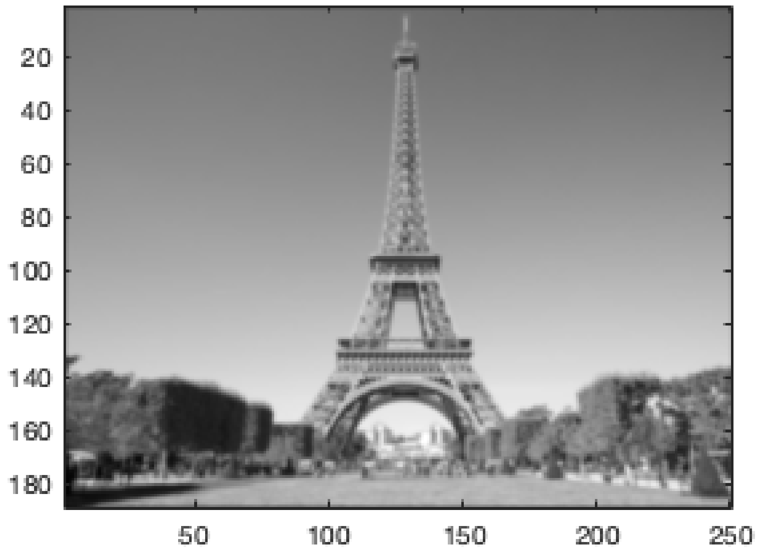
```
blurred_tower = conv2(tower, kernel);
blurred_no_border = remove_border(tower, blurred_tower);
show_img(blurred_no_border);
```

```
Image being scaled so that min value is 0 and max value is 255
```



```
blurred_no_border_downsampled = blurred_no_border(1:d:end,1:d:end);
show_img(blurred_no_border_downsampled);
```

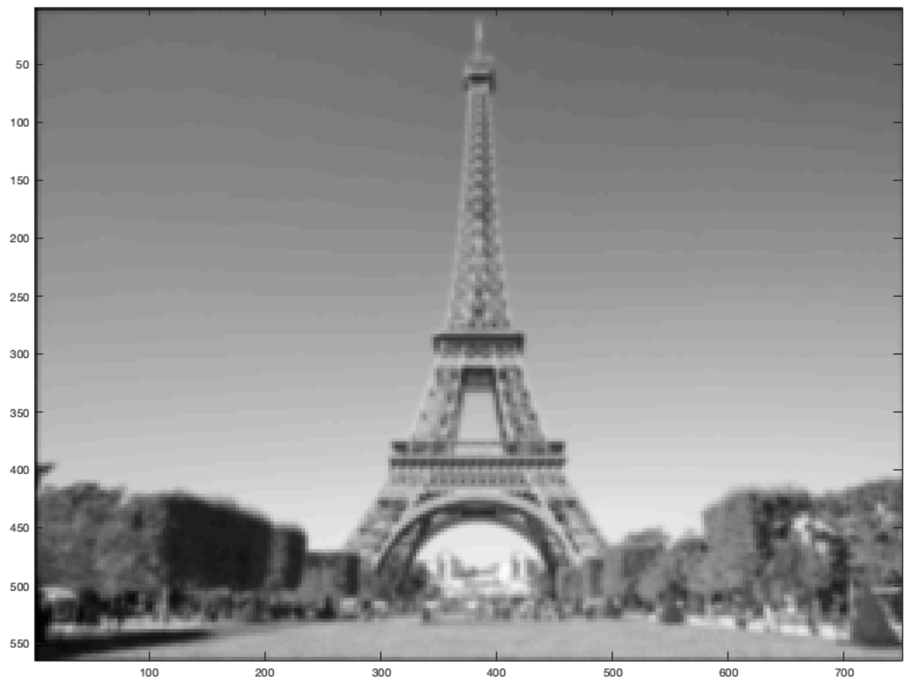
Image being scaled so that min value is 0 and max value is 255



```
[rows, cols] = size(blurred_no_border_downsampled);
xrows = blurred_no_border_downsampled(ceil(1/d:1/d:rows), :);
blurred_hold = xrows(:, ceil(1/d:1/d:cols));
```

```
show_img(blurred_hold);
```

```
Image being scaled so that min value is 0 and max value is 255
```



Question 12: Describe the visual differences between the interpolated tower and the interpolated blurry tower.

The following image numbers provide a mappinng to a description of the image:

1 -> unaltered

2 -> downsampled by a factor of 3 applied to #1

3 -> upsampled by a factor of 3 applied to #2

4 -> blurred, no border applied to #1

5 -> downsampled by a factor of 3 applied to #4

6 -> upsampled by a factor of 3 applied to #5

When we compare the differences between #3/#1 vs. #6/#4, we can see that the blurred reconstruction is much more true to the blurred image than the non-blurred reconnstruction is to the unaltered photo. The normal interpolated tower appears grainy and unsimilar to the origial tower photo. There are few differences between the blurred reconstruction and the blurred original photos. The two reconstructed photos are both of the tower, but there is better detail in the blurred reconstructed version when compared to the non-blurred reconstructed version.

Published with MATLAB® R2019a