# Image Quantization and Sampling

Project 1

EECS 490

Shaun Howard

Due: September 16th, 2016

Submitted: September 16th, 2016

Image processing is very important to modern life as cameras have grown more mature along with computer hardware. Matlab has several different functions for handling the detailed images produced by these cameras and computers. In this paper, we examine some of the image quantization and sampling functionality Matlab R2016a has to offer. Functions for subsampling, sub-plotting, filtering, quantizing, and constructing image pyramids to reduce images with a Gaussian pyramid technique are compared and analyzed. Two scenarios are presented where such functionality is tested. The second scenario also implements a form of 3x3 average filtering between image subsamples. Results show that filtering between subsamples resulted in smoother images to both view and distinguish with Matlab subplot and imshow functions.

## 1. Technical Discussion

We were interested in the usefulness of Matlab image quantization and sampling functionality for our experiments. The first function utilized in Matlab was `imread`, which accepts an image file path and type as input, for reading in the 1024x1024 pixel test image as shown in figure 1 of section 3. One of the first techniques we utilized for subsampling images applied the Matlab `downsample` function to down-sample the image by factors of two. The `downsample` function, however, only down-samples one dimension of the image array given per call. Thus, the method was applied to both the rows and columns of the image in sequence, via transpose, to obtain the fully down-sampled image. Subsequently, we used the `subplot_tight` and `imshow` functions to display the subsampled images in a grid with equal cell size despite their varying resolutions as in figure 2 in section 3. The `subplot_tight` function was found on the Matlab File Exchange (FEX) and accepts the row and column counts of the plot, the current image position in the plot and an array of margins for spacing between images of the subplot. This function was necessary to display a grid of images without gaps in between as compared to the regular `subplot` function, which displays images with gaps. The `imshow` function simply takes the image array as input and displays the image in the current subplot cell.

Next, the image was quantized as gray levels were reduced in the original image using the Matlab `gray2ind` function, which takes the image and the number of gray levels to keep in the image as parameters. The gray levels were reduced in the test image from 256 to 2 by roots of two and displayed in a grid using the `subplot` and `subimage` functions as in figure 3 in section 3. The `subimage` function is equivalent to the `imshow` function for creating subplots, but it was easier to use `subimage` for displaying images with a color map than the `imshow` function.

Following, a Gaussian image pyramid was constructed using the `impyramid` function in Matlab. Each of the down-sampling results were provided as input to the function and outputs were displayed as discussed in lecture but according to the support of the Matlab `imshowTruesize` function found on the FEX. The external function `imshowTruesize` displays images in a true-sized format and can construct an image layout similar to that of the pyramids shown in class. The function takes as input an image cell array, which can include images of various sizes, an array of margins between the image borders, and the alignment of the pyramid. We chose a top alignment for the pyramid as it was closest to the desired image format. In this case, the images were displayed using the `imshowTruesize` and subplot attributes such as the axis labels of the subplots were modified with the `hSubplot` function of the `imshowTruesize` image handle as displayed in figure 4 of section 3.

Proceeding, we wanted to determine the usefulness of filtering in Matlab and image processing in general. Since we have not covered filtered thoroughly in class, a simple 3x3 average filter was utilized in between subsamples of images. The filter was created using the `fspecial` function in Matlab, which takes parameters of the filter type and an array for kernel dimensions. A 3x3 averaging kernel was utilized to filter the test image subsamples after each previous subsample. An equally-divided grid was created for the subsampling results using `subplot_tight` and `imshow` in figure 5 of section 3. As before, an image pyramid was constructed using `impyramid`, but now the pyramid was constructed with the filtered and subsampled images in a top-aligned fashion as show in figure 6.

**2. Discussion of Results**

Given the procedures outlined in section 1, Matlab was able to provide the functionality desired for image sampling and quantization. Matlab allowed sub-sampled images from 1024x1024 to 32x32 to be displayed at equal size within a 3x2 grid as discussed in lecture 2 using `subplot_tight` and `imshow` as in figure 2. These functions displayed the subsampled images at full resolution using their own form of pixel filling. As the image was resized by the subsampling process, the resolution was decreased. The decrease in resolution led to a degradation in the quality of the image. With this degradation, edges formed within the broader pixels that did not properly delineate the original edges in the image. Such edges made the image much less detailed and less pleasant to view.

Gray levels were reduced as desired from 256 to 2 in powers of 2 using `gray2ind` and displayed using `subplot_tight` and `subimage` in figure 3. Using `impyramid` and `imshowTruesize`, we were able to construct a Gaussian image pyramid like that of figure 7.47 in the project 1 assignment. The pyramid in figure 4 does not have smaller images left-aligned from top to bottom along the larger image, however, like that of figure 7.47 given the limitations of the `imshowTruesize` function, but it has a very similar representation with the largest on the left and smallest on the right in decreasing order along the top. With any reduction, loss of quality and detail are both side effects. Each time the grays levels of the image were reduced, the result degraded in quality and detail. The way the image degraded was a bit different than as with subsampling. Given that the original image is a geographical aerial view, it can be used as a map given there are enough colors to represent the delineations. However, reducing the gray colors in the already grayscale image reduced the clarity and distinguishability of the contents in the map. The reason that the overlap of once delineated edges is bad for such a picture is that false contours are created between mountains, houses, roads, bodies of water and land in general. These contours

make the image useless for any detailed purpose like trying to find a path in between two landmarks.

Continuing, a 3x3 averaging filter was applied on subsampled images, between subsample occurrences. The images were again tiled using `subplot_tight` and `imshow` with equal size as in figure 5. The effect of filtering was that the images did not become sharply pixelated after subsampling, but rather, they became smoothly pixelated as they decreased in resolution. These results are unlike the previous sub-sampling results, since the earlier subsampling resulted in a sharply pixelated image. An image pyramid was subsequently built to display the filtered and subsampled images in true-size as displayed in figure 6. Although generic 3x3 kernel size was used for the averaging filter, other-sized kernels could have been applied to make the results look different. For instance, a smaller kernel would have resulted in sharper results, whereas a larger kernel would have resulted in blurrier results given that the kernel size influences the spread of the filter across each image.

**3. Results**

The original image and the image processing results as discussed in section two are displayed visually below.

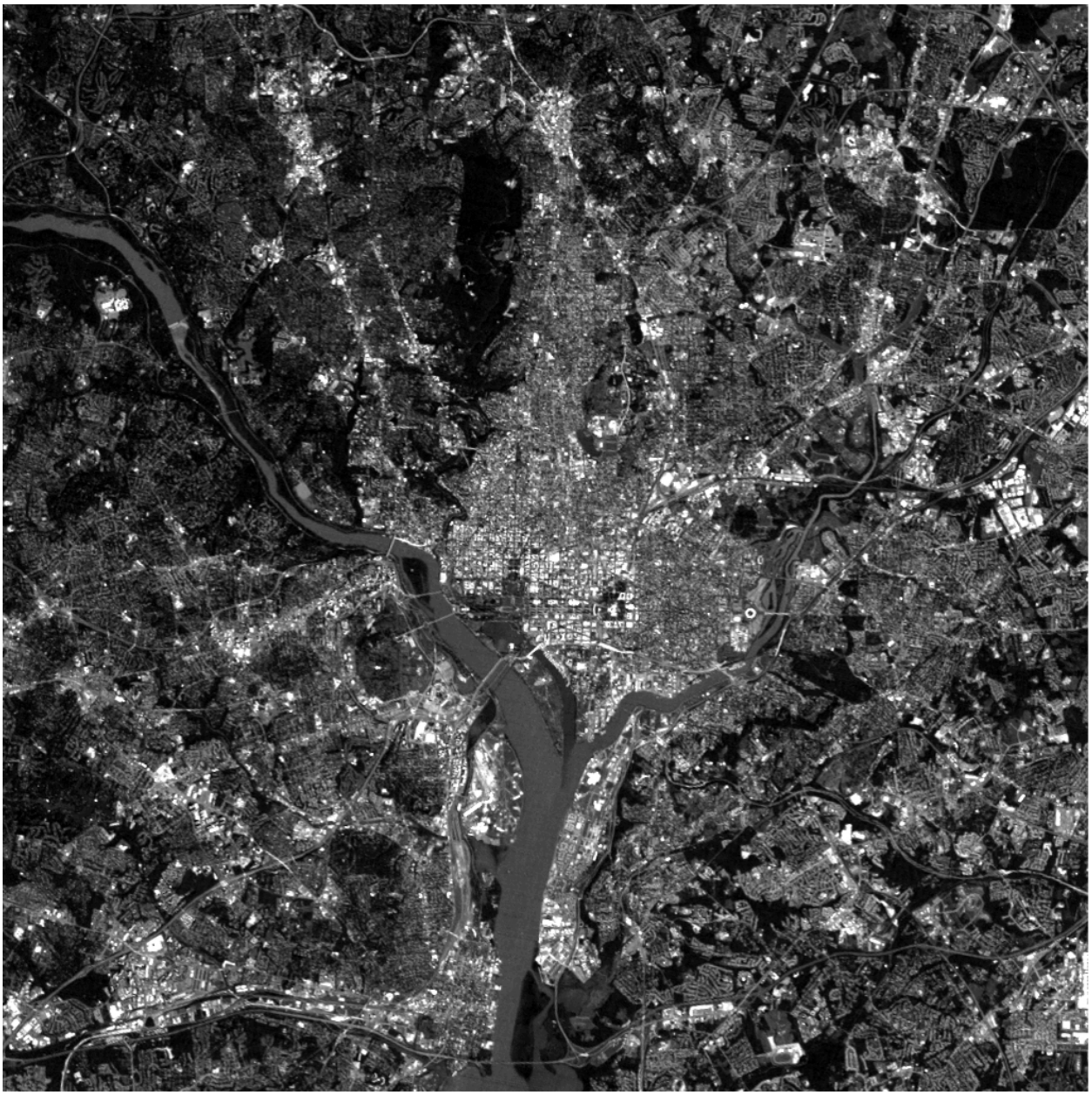

*Figure 1: The image used for sampling and quantization experiments with Matlab R2016a.*

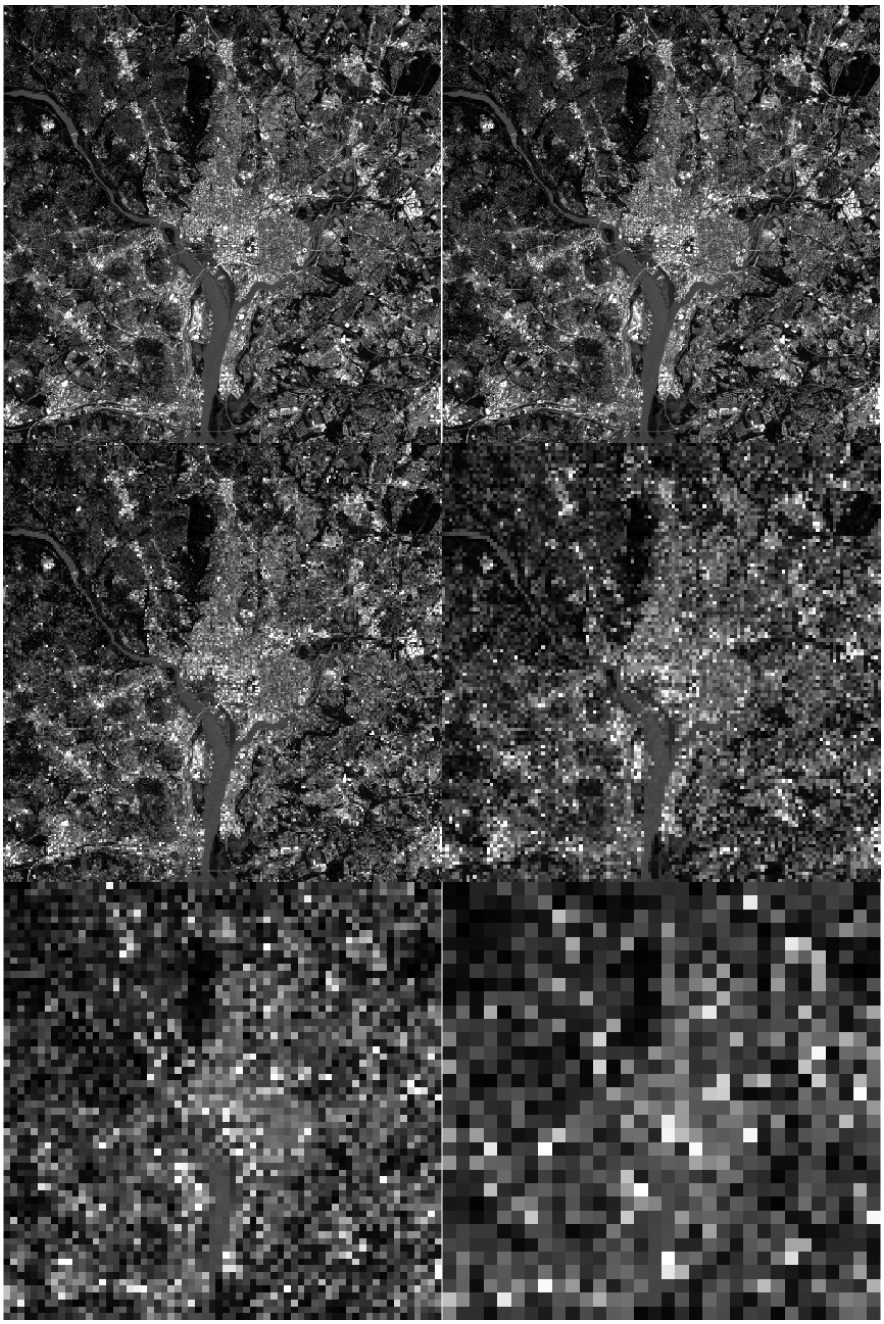

*Figure 2: Grid of down-sampled images tiled in decreasing resolution from left to right, top to bottom.*

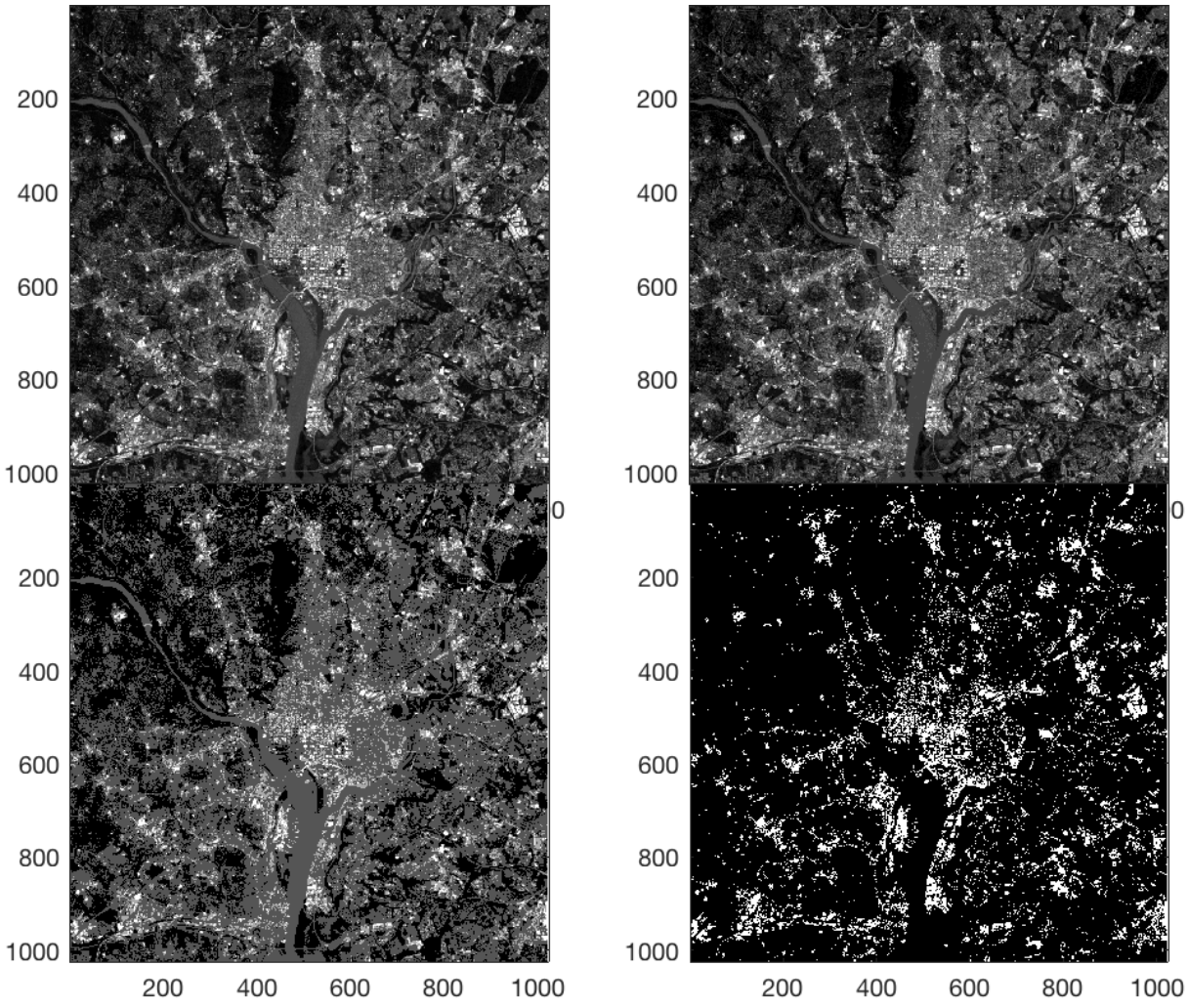


*Figure 3: Grid of images with decreasing gray levels from 256 to 2 over powers of two.*

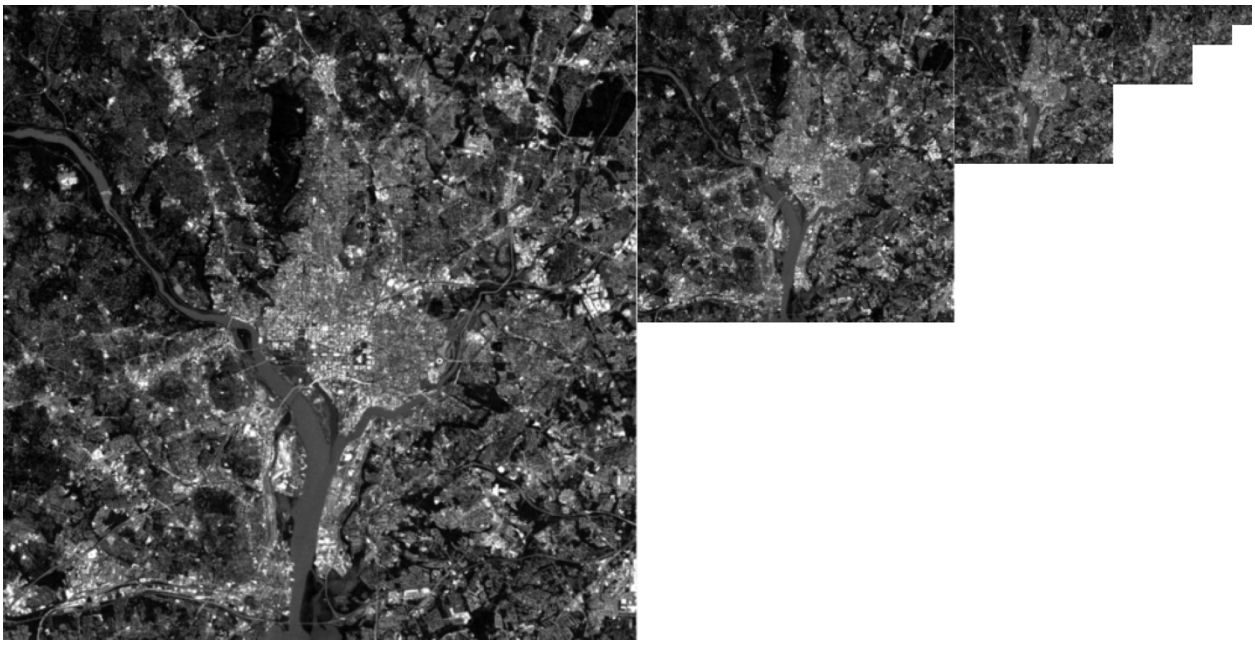

*Figure 4: Gaussian pyramid of images subsampled by factors of two as displayed in figure 2.*

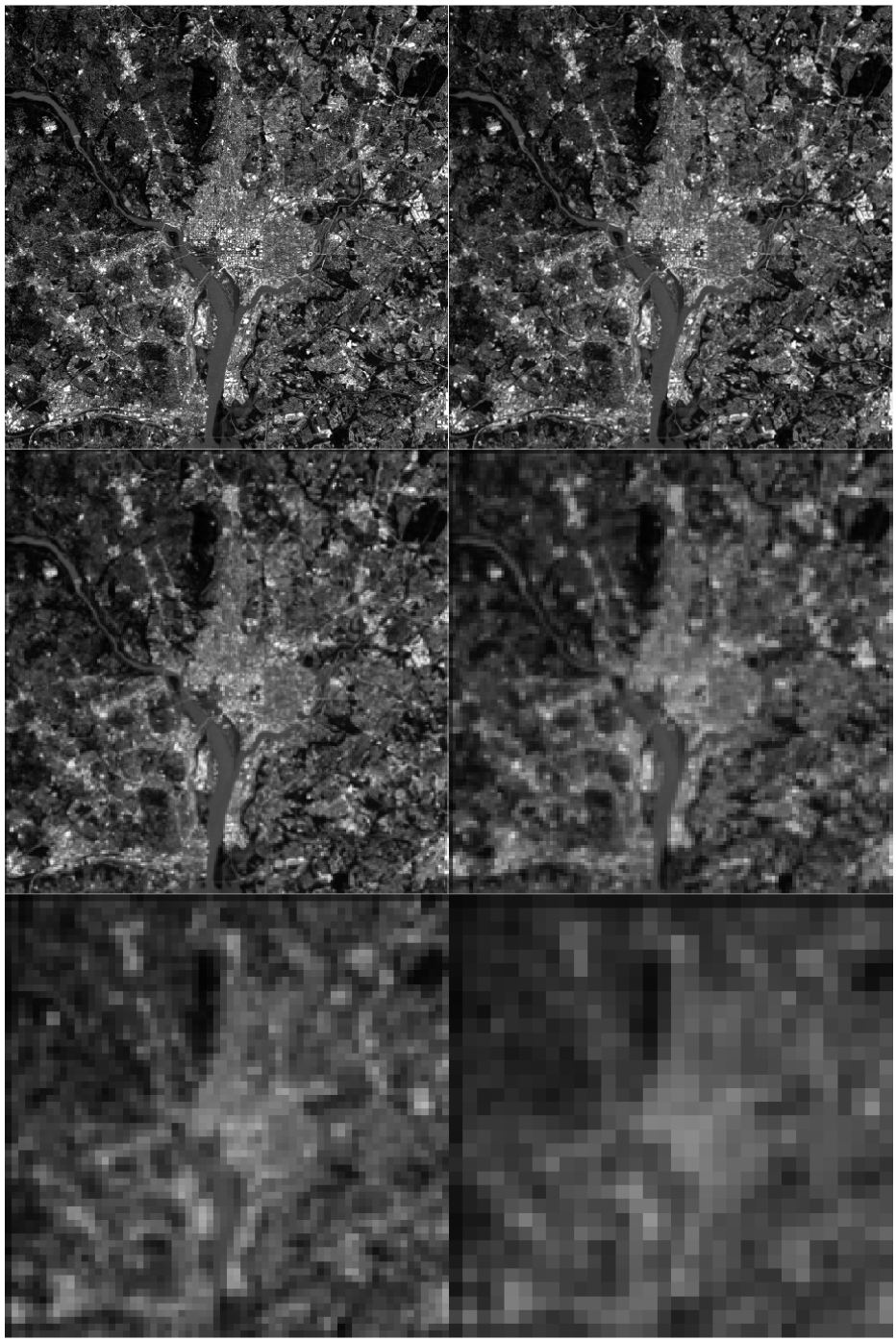

*Figure 5: Grid of images filtered with a 3x3 averaging filter and subsampled by factors of 2. Images are tiled in decreasing resolution from left to right, top to bottom.*

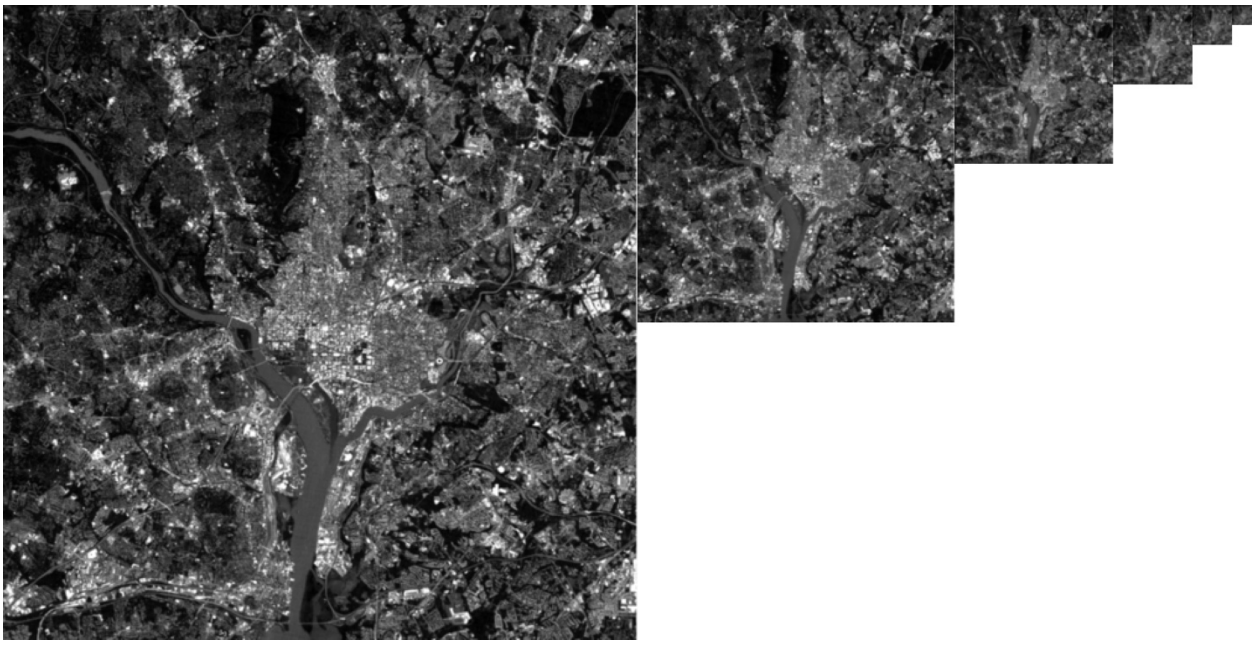

*Figure 6: Gaussian pyramid of images filtered and subsampled by factors of two as displayed in figure 5.*

## 4. Appendix

The program used to experiment with Matlab for image quantization and sampling is displayed below. This program also relies on two Matlab functions found on the File Exchange (FEX), `subplot_tight` and `imshowTruesize`, as previously discussed, but they were not included here for brevity.

```
% Shaun Howard
% EECS 490 Project 1: Image Quantization and Sampling

% read a 1024x1024 grayscale image
original_image=imread('proj1fig', 'jpg');
I=mat2gray(original_image);
% get image dimensions
[height,width] = size(I);
% store images and the total number of them
n_images=6;
figure('name', 'Subsampled by Factors of Two');
% display original image
margins=[0 0];
subplot_tight(3,2,1, margins);
imshow(I)
image_arr={I};
i=2;
while i <= n_images
    % downsample image rows, columns
```

```
    J=downsample(I,2);
    ds=downsample(J',2)';
    image_arr=[image_arr {ds}];
    % display downsampled image
    subplot_tight(3,2,i, margins);
    imshow(ds);
    I=ds;
    i=i+1;
end

i=1;
gray_levels=[256, 16, 4, 2];
quantized_images=[];
quantized_maps=[];
I=mat2gray(original_image);
figure('name', 'Gray Levels Reduced by Powers of Two')
% Quantize gray levels in original image from 256 to 2 in powers of 2
while i < 5
   [quantized_image,map32]=gray2ind(I, gray_levels(i));
   % display quantized images
   subplot_tight(3,2,i, margins)
   subimage(quantized_image,map32)
   i=i+1;
end

% construct an image pyramid from the downsampled images
i = 1;
pyramid_cells={};
while i <= n_images
    py=impyramid(image_arr{i}, 'reduce');
    pyramid_cells=[pyramid_cells {py}];
    i=i+1;
end

% plot pyramid with imshowTruesize - true aspect ratio is preserved
margins = [0 0];
Handles = imshowTruesize(pyramid_cells,margins,'top');
for iCol = 1:n_images
  axis(Handles.hSubplot(1,iCol),'off')
end

% construct 3x3 avging filter
avg_filter=fspecial('average',[3 3]);

figure('name', 'Images Filtered and Subsampled by Factors of Two');

% display original image
subplot_tight(3,2,1, margins)
imshow(original_image)
I = original_image;
image_arr={I};
i=2;
while i <= n_images
    % average filter image
    I=imfilter(I,avg_filter);
    % downsample image rows and columns
    J=downsample(I,2);
```

```
    ds=downsample(J',2)';
    image_arr=[image_arr ds];
    % display downsampled image
    subplot_tight(3,2,i, margins)
    imshow(ds)
    I=ds;
    i=i+1;
end

% construct an image pyramid from the filtered and downsampled images
i = 1;
pyramid_cells={};
while i <= n_images
    py=impyramid(image_arr{i}, 'reduce');
    pyramid_cells=[pyramid_cells {py}];
    i=i+1;
end

% plot with imshowTruesize - true aspect ratio is preserved
margins = [0 0];
Handles = imshowTruesize(pyramid_cells,margins,'top');
for iCol = 1:n_images
  axis(Handles.hSubplot(1,iCol),'off')
end
```