**Full Name:** _____

**EEL 3135 (Spring 2021) – Lab #03** **Due: 11:59 PM ET (On Lab Day)**

## Submission Notes

- Your laboratory solutions should be submitted on Canvas as a single published MATLAB PDF.
- Use the provided skeleton code as the basis for your solutions (easier for you and the graders).

**Question #1:** *(Image Processing)* Download `eel3135_lab03_comment.m` from Canvas, replace each of the corresponding comments with the corresponding descriptions. This is designed to show you how to work with images in MATLAB.

**Note:** You should run the code to help you understand how it works and help you write your comments. You will use elements of this MATLAB code for the rest of the lab assignment.

**Question #2:** *(Sampling)* The following three questions handle three important processes in image processing: (1) sampling (converting a large image to a small image), (2) anti-aliasing (reducing distortions from aliasing), and (3) interpolation (converting a small image to a large image).

For these three questions, add your code into the pre-made skeleton `eel3135_lab03_skeleton.m` from Canvas. Include all code (and functions) in this one MATLAB m-file so that everything is published to a single PDF.

(a) Write a new function `[xs, ys, zs] = sample(z, D);` that inputs a high-resolution image `z` and samples every `D` pixels in both the horizontal and vertical direction. It outputs the sampled image `zs` and the new axes `xs` and `ys`. Use the function `image_system1` from Question #1 as a guide (hint: most of the information you need is in this function). Include this new function at the end of the skeleton .m file.

(b) Apply `sample` to the image `z` in the skeleton code. Sample every `D = 8` pixels in the horizontal and vertical directions. Use `subplot` to show side-by-side images before and after sampling.

(c) Apply `sample` to the image `z` two more times, first with `D = 16` and then `D = 24`. Use `subplot` again to show the side-by-side for these two different sampling rates.

(d) **Answer in your comments:** How does aliasing manifest in the sampled image (hint: aliasing distorts your signal)? Relate this to your understanding of aliasing from class.

**Question #3:** *(Anti-Aliasing)*

(a) Create a new function `zaa = antialias(z);` which inputs a high-resolution image `z` and outputs high-resolution anti-aliased image `zaa`. Use the function `image_system2` from Question #1 as a guide. Design the anti-aliasing system to compute each point of $z_{aa}[x, y]$ (i.e., the mathematical notation for `zaa`) according to the two-dimensional difference equation (i.e., a discrete-time system)

$$z_{aa}[x, y] = (1/2)z[x, y] + (1/8)(z[x-1, y] + z[x+1, y] + z[x, y-1] + z[x, y+1])$$

Include this new function at the end of the skeleton .m file.

(b) Apply `antialias` to your high-resolution image `z`. Use `subplot` to show side-by-side images before and after applying the anti-aliasing filter.

(c) Use your `sample` function to sample every `D=8` pixels of the anti-aliased image `zaa` to obtain output `zaas`. Use `subplot` to show side-by-side **sampled** images with and without the anti-aliasing filter.

(d) Use a `for`-loop to apply the anti-aliasing filter to the high-resolution image `z` 64 times and then apply `sample` to obtain a new `zaas`. This applies a "x64 anti-aliasing filter." Use `subplot` to show side-by-side **sampled** images with and without the x64 anti-aliasing filter.

(e) **Answer in your comments:** What does the anti-aliasing filter do to the image? How does this reduce aliasing? Why is this useful in real-world applications?

**Question #4:** *(Interpolation)*

(a) Create a new function `[xz, yz, zz] = addzeros(zaas, U);` which inputs an anti-aliased and sampled image `zaas`. The function outputs the image `zz`, which contains `U` zero-valued pixels inserted between each pixel from `zaas`, both horizontally and vertically. It also outputs the new axes `xz` and `yz`. Use the function `image_system1` from Question #1 as a guide (hint: most of the information you need is in this function). Include this new function at the end of the skeleton .m file.

(b) Apply `addzeros` to your x64 anti-aliased and sampled image `zaas` in Question #3 to obtain `zz`. Add `U=8` zeros between each pixel. Use `subplot` to show side-by-side images before and after applying the function.

(c) Your function `antialias` can be an anti-aliasing filter or an interpolation filter! Therefore, apply `antialias` to your image-with-zeros `zz` to obtain `zzaa`. Use `subplot` to show side-by-side images before and after applying the interpolation filter.

(d) Use a `for`-loop to apply the filter to `zz` 64 times to obtain a new `zzaa`. This results in a "x64 interpolation filter." Use `subplot` to show side-by-side the original high-resolution image `z` and the new interpolated image `zzaa`.

(e) **Answer in your comments:** What does the interpolation filter do to the image? Why is this useful in real-world applications?