Full Name: _____

EEL 3135 (Spring 2025) – Lab #03

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. Use `colormap(gray)` to display images in grayscale.

(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 `Fxs` 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 = 5` pixels [1] 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 with two different D values: $D = 10$ an $D = 15$. Use `subplot` again to show the side-by-side for these two different sampling rates.

(d) **Answer in your comments:** After applying sampling in some cases, specific regions of the images appear to have different frequencies. Describe the characteristics of those regions and explain why the samping would change them.

(e) **Answer in your comments:** Temporarily uncomment the `imread` line from the skeleton code to load a "natural" image. Run your code with this. How does aliasing manifest in this image differently? (Note: Submit your solution with the original image, not this one.)

[1] Technically, we are **resampling** the signal. Hence, if we originally sampled at $f_s = 100$ Hz and $D = 5$, then the new sampling rate is $f_s' = 100/D = 20$ Hz.

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 $z_{aa}[x, y]$ according to the 2D difference equation (i.e., a discrete-time system)

$$z_{aa}[x,y] = (1/9)(z[x-1,y-1] + z[x-1,y] + z[x-1,y+1] + z[x,y-1] \\ + z[x,y] + z[x,y+1] + z[x+1,y-1] + z[x+1,y] + z[x+1,y+1])$$

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

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

(c) Use a `for`-loop to apply the anti-aliasing filter to the high-resolution image `z` 64 times. This applies a "x64 anti-aliasing filter" to get `z64`. Use `subplot` to show side-by-side images with and without the x64 anti-aliasing filter.

(d) Use your `sample` function to sample every `D=15` pixels of the anti-aliased image `zaa` to obtain output `zaas` and to the 64x anti-aliased image `z64` to obtain output `z64s`. Use `subplot` to show side-by-side **sampled** images with 1x anti-aliasing and 64x anti-aliasing.

(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?

(f) **Answer in your comments:** Temporarily uncomment the `imread` line from the skeleton code to load a "natural" image. Run your code with this. How does the anti-aliasing affect this image differently? (Note: Submit your solution with the lines image, not this one.)

Question #4: *(Video)* Included is a video `wheel_video.mp4` [2]. The skeleton code plays the video at a rate of $f_s = 30$ Hz. The wheel in the video spins at a rate of approximately 0.375 rotations per second (Hz). However, since there are 5 spokes, we really observe the wheel repeating with a rate of 1.875 Hz (0.375 times 5). Use the skeleton code to complete the questions below.

(a) Create a function `z = video_sample(x, Dx, Dy, Dt)` that can sample a video `x`. Sample every `Dx` horizontal pixels, every `Dy` vertical pixels, and every `Dt` time values. The output of the function is the sampled video z.

(b) In space, sample the video to have 54 pixels in the horizontal direction and 90 pixels in the vertical direction. In time, sample the video so that the wheel does not move (note: it will move a little since the frequency is a little off). Submit a .mp4 output with your report.

(c) **Answer in your comments:** Using sampling theory to justify your choices for `Dt` in the last subproblem.

(d) In space, sample the video to have 54 pixels in the horizontal direction and 90 pixels in the vertical direction. In time, sample the video so that the wheel moves backwards. Submit a .mp4 output with your report.

(e) **Answer in your comments:** Using sampling theory to justify your choices for `Dt` in the last subproblem.

[2] The video is a modified version of the video from `https://www.videvo.net/video/spinning-wheel/793/`