ECE558 Project 02

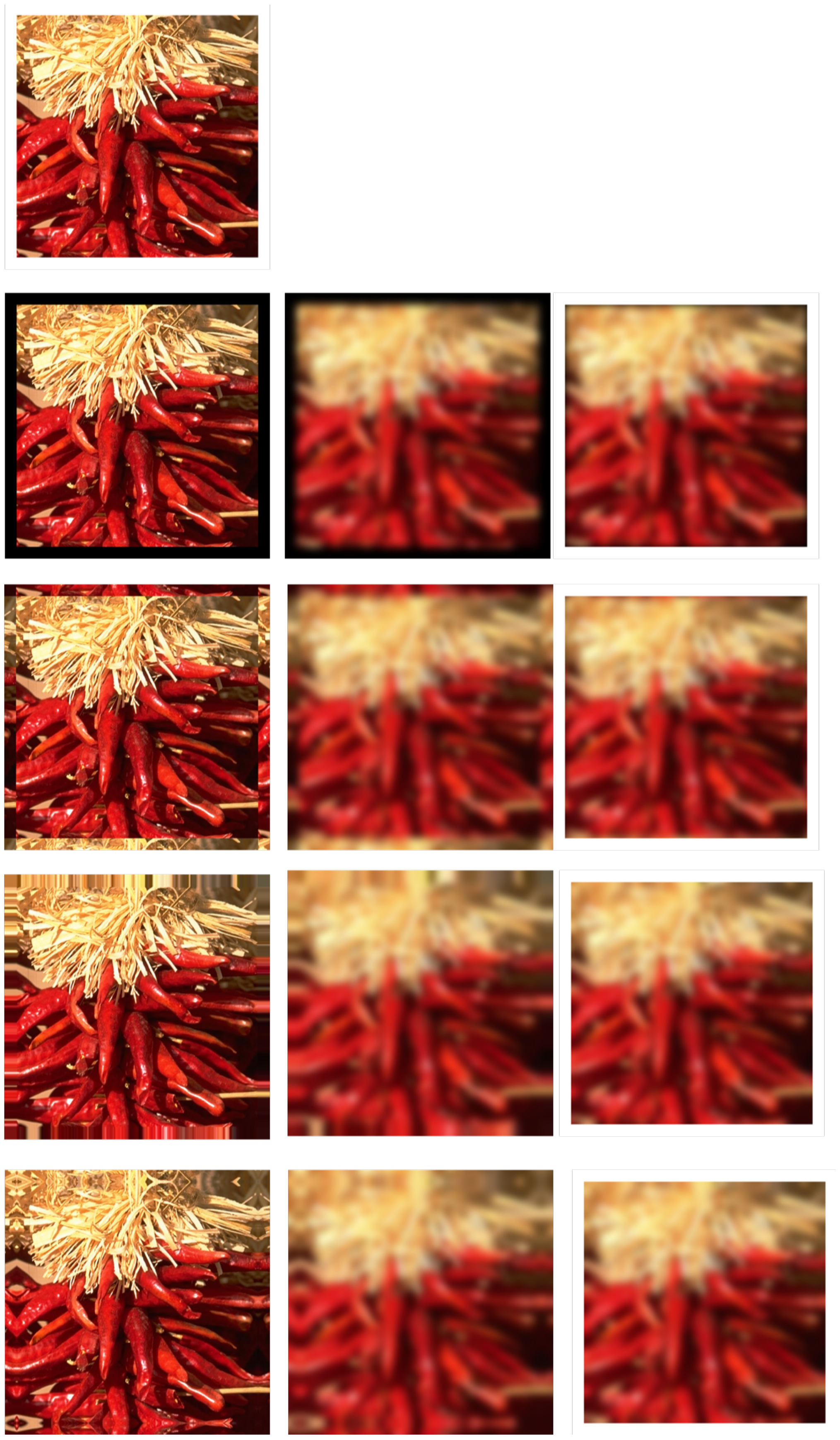
Due 10/30/2022

How to submit your solutions: put source code folder [your_unityid_code], your report (word or pdf) and results images (.png) if had in a folder named [your_unityid_project01] (e.g., twu19_project01), and then compress it as a zip file (e.g., twu19_project01.zip). Submit the zip file through **moodle**.

If you miss the deadline, please send your zip file to TAs and me.

Problem 1 (50 points): Two-dimensional convolution.

(a) [30 points] Write a function to implement $g = conv2(f, w, pad)$, where f is an input image (grey, or RGB), w is a 2-D kernel (e.g., 3×3 box filter), and pad represents the 4 padding type (see page 17 in the handout notes of lecture 8): clip/zero-padding, wrap around, copy edge, and reflect across edge, as illustrated in the following example (2nd to 5th row).



For RGB images, you should apply the filter kernel for each channel (R, G, B) independently, and the filtered image is also RGB color image.

You can use existing modules in Matlab or Python as reference. But, **you need to write the convolution function from scratch**.

Test your function with the provided lenna.png and wolves.png (you can use built-in color conversion functions to convert them to grey images). For the kernel w, test the following:

1) Box filter: $\frac{1}{9}\begin{array}{|c|c|c|}\hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline\end{array}$

2) The simple first order derivative filter: $\begin{array}{|c|c|}\hline -1 & 1 \\ \hline\end{array}$ and $\begin{array}{|c|}\hline -1 \\ \hline 1 \\ \hline\end{array}$ or $\begin{array}{|c|}\hline 1 \\ \hline -1 \\ \hline\end{array}$

3) And the six filters:

Prewitt: $M_x = \begin{array}{|c|c|c|}\hline -1 & 0 & 1 \\ \hline -1 & 0 & 1 \\ \hline -1 & 0 & 1 \\ \hline\end{array}$; $M_y = \begin{array}{|c|c|c|}\hline 1 & 1 & 1 \\ \hline 0 & 0 & 0 \\ \hline -1 & -1 & -1 \\ \hline\end{array}$

Sobel: $M_x = \begin{array}{|c|c|c|}\hline -1 & 0 & 1 \\ \hline -2 & 0 & 2 \\ \hline -1 & 0 & 1 \\ \hline\end{array}$; $M_y = \begin{array}{|c|c|c|}\hline 1 & 2 & 1 \\ \hline 0 & 0 & 0 \\ \hline -1 & -2 & -1 \\ \hline\end{array}$

Roberts: $M_x = \begin{array}{|c|c|}\hline 0 & 1 \\ \hline -1 & 0 \\ \hline\end{array}$; $M_y = \begin{array}{|c|c|}\hline 1 & 0 \\ \hline 0 & -1 \\ \hline\end{array}$

Show the filtering results. [optional] You can test more images.

(b) [20 points] Create a grey image of size 1024x1024 pixels that consists of a unit impulse at the center of the image (512, 512) and zeros elsewhere. Use this image and a kernel of your choice (e.g., selected one from (a)) to confirm that your function is indeed performing convolution. Show your filter result and explain why your function is indeed performing convolution.

Problem 2 (50 points): Implementing and testing the 2-D FFT and its inverse using a built-in 1-D FFT algorithm.

(a) [30 points] obtain a built-in routine that computes the 1-D FFT. For example, if you know how to integrate c/c++ functions to Matlab or Python, you can use open source implementations of 1-D FFT at www.fftw.org. Or you can use fft in Matlab itself. Use the built-in 1-D FFT to implement $F = DFT2(f)$ from scratch, where f is an input grey image.

Test your function with the provided lenna.png and wolves.png (you can use built-in color conversion functions to convert them to grey images). Before apply the DFT2, you need to scale the grey image to the range [0, 1] (e.g., implement Problem 4 in HW02 and integrate it in your DFT2 function).

Visualize the spectrum and phase angle image. When visualizing them, apply the transform $s = \log(1 + abs(F))$ or others as you see fit.

(b) [20 points] Using your DFT2 to implement the inverse FFT of an input transform F, $g = IDFT2(F)$ from scratch. (Hint, check item 12) in the top of page 12 in the handouts of lecture 13).

Test your function for the two results in (a). Given an input grey image f, you use (a) DFT2 to compute its F , and use your IDFT2 to compute g from F. Visualize f and g and they should look identical. To double-check it, visualize $d = f - g$, which should be a black image.