Activity 5: The Calculus of Images

Primary Goal: Learn how to calculate the gradient and Total Variation (TV) energy of an image.
Secondary Goal: Practice writing Matlab functions.

1.) The Gradient

The gradient ∇u is the a vector of the partial derivatives of the function u(x,y):

$$\nabla u = \langle u_x, u_y \rangle$$

We can approximate the derivatives using finite differences. The central differences D^0 are:

$$u_x \approx D_x^0 u = \frac{u(x+1,y) - u(x-1,y)}{2}$$
$$u_y \approx D_y^0 u = \frac{u(x,y+1) - u(x,y-1)}{2}$$

a.) Load the cameraman image.

```
u = imread('cameraman.tif');
u = double(u);
```

Compute the central difference matrices to approximate u_x and u_y. *(You might want to consult the lecture notes.)* Display these images side-by-side. You should notice that they detect different types of edges.

The derivative in the x-direction u_x detects ____vertical____ edges.

The derivative in the x-direction u_y detects ____horizontal____ edges.

b.) We can combine the two derivative matrices into one matrix by taking the norm of the gradient.

$$\|\nabla u\| = \sqrt{u_x^2 + u_y^2}$$

When you do the exponent in Matlab, make sure you use the .^ expression to do point-wise exponentiation. Compute the norm matrix N and display it. You should see that the image is brightest at the strong edges.

c.) To create an edge detector, we want to make a binary YES/NO decision about whether each pixel is an edge. We can do this by thresholding our norm matrix. Add a colorbar to the image from part (b) to help you decide on a good threshold value. Then threshold the image using a command like:

```
E = [N > t];
```

where t is some threshold value you decided upon. Try displaying edge detection images for different values of the threshold parameter t.

2.) Your Very Own TV

We want to write a function that computes the Total Variation (TV) energy of a grayscale image. Create a function in Matlab by selecting `New->Function`. Set up your function line as something like:

```
function [TV_value] = TV (A)
```

Save your file to your Desktop now as TV.m. Note the function name and return value must be different.

For good programming style, every function should start with documentation about the function and how it is used. You can create comments at the top by starting the line with the % sign. Add comments something like this directly below your function line.

```
% Calculate TV value of an image.
% Input:  Grayscale image A
% Output: Scalar TV_value specifying the TV energy of image A
```

The TV energy of an image u(x,y) defined on a rectangular domain Ω is

$$TV(u) = \iint_\Omega \|\nabla u\| \, dx\, dy.$$

First approximate the partial derivatives using a forward difference. Then create a matrix describing the norm of the gradient at every pixel. Then use the `sum` command to approximate the double integral. *(Consult this week's lecture notes for coding hints.)*

Test your function on a noise-free image and write down the TV value.

```
P = imread('pout.tif');
x = TV(P)
```

TV = 160322.3088

Now let's add 5% Salt & Pepper noise to the image and see how the TV value changes.

```
P_noisy = imnoise(P, 'salt & pepper', 0.05);
x = TV(P_noisy)
```

TV = 917146.6370

The Matlab code below will gradually add more noise to an image and display the TV value. Type this code into a script and watch what happens. *(Make sure no figures are open before you run it.)*

```
P = imread('pout.tif');
for i = 1:10
      P = imnoise(P, 'salt & pepper', 0.02);
      imshow(P);
      x = TV(P);
      xlabel( [ 'TV = ', num2str(x) ] );
      title( [ 'Iteration #', num2str(i) ] );
      pause(0.5);
end;
```