Activity 7: TV Denoising

Primary Goal: Code the Total Variation denoising algorithm.
Secondary Goal: Gain familiarity with the idea of energy minimization and the calculus of variations.

The variational method is a 4-step process.

Step 1: Create an energy E that describes the quality of image u.
(Low energy = good image. High energy = bad image.)

$$E[u] = \int_{\Omega} g\left(u, u_x, u_y\right)\, d\vec{x}$$

Step 2: Compute the first variation of energy ∇E.

$$\nabla E = \frac{\partial g}{\partial u} - \frac{\partial}{\partial x}\frac{\partial g}{\partial u_x} - \frac{\partial}{\partial y}\frac{\partial g}{\partial u_y}$$

Step 3: Set up the PDE describing the steepest descent minimization:

$$\frac{\partial u}{\partial t} = -\nabla E$$

Step 4: Discretize the PDE and evolve the PDE towards the minimum of E.

$$u^{n+1} = u^n + \Delta t \left(-\nabla E[u^n]\right)$$

Note Steps 1-3 are all done by hand. We do not need a computer until Step 4.

Below is a rough outline of how you would code Step 4 in Matlab.

```
function [u] = MY_ALGORITHM (f)

% First set the values of the time step dt, stopping time T, and any other parameters.
u = double(f);          % Initialize as input image f.

for t = 0:dt:T
        % Calculate any finite differences and values necessary to compute ∇E.
        u = u + dt (− ∇E);
end

u = uint8(u);
```

1.) Computing the First Variation

As a toy example, define the energy

$$E[u] = \int_\Omega \left(u^3 + 2\, u_x^4 + 3u_y\right) d\vec{x}.$$

a.) Compute the quantities listed below.

$$g = \quad u^3 + 2u_x^4 + 3u_y$$

$$\frac{\partial g}{\partial u} = \quad 3u^2$$

$$\frac{\partial g}{\partial u_x} = \quad 8u_x^3$$

$$\frac{\partial}{\partial x}\left(\frac{\partial g}{\partial u_x}\right) = \quad 24u_{xx}u_x^2$$

$$\frac{\partial g}{\partial u_y} = \quad 3$$

$$\frac{\partial}{\partial y}\left(\frac{\partial g}{\partial u_y}\right) = \quad 0$$

b.) Now write the first variation of energy ∇E.

$$\nabla E = \frac{\partial g}{\partial u} - \frac{\partial}{\partial x}\frac{\partial g}{\partial u_x} - \frac{\partial g}{\partial y}\frac{\partial}{\partial u_y} = \quad (u^3 + 2u_x^4 + 3u_y) - 24u_{xx}u_x^2$$

c.) Write the PDE that you would evolve to minimize the energy E[u].

$$\frac{\partial u}{\partial t} = -\nabla E = -(u^3 + 2u_x^4 + 3u_y) + 24u_{xx}u_x^2$$

2.) The TV Energy

The Rudin-Osher-Fatemi Total Variation (TV) Energy for an input image f is defined as

$$E[u|f] = \int_{\Omega} ||\nabla u|| \quad + \lambda(u-f)^2 d\vec{x}$$

a.) Compute the following quantities.

$$g = \sqrt{u_x^2 + u_y^2} + \lambda(u-f)^2$$

$$\frac{\partial g}{\partial u} = 2\lambda(u-f)$$

$$\frac{\partial g}{\partial u_x} = \frac{u_x}{\sqrt{u_x^2 + u_y^2}}$$

$$\frac{\partial}{\partial x}\left(\frac{\partial g}{\partial u_x}\right) = \frac{u_{xx}u_y^2 - u_x u_y u_{xy}}{(u_x^2 + u_y^2)^{3/2}}$$

b.) Without actually doing the computation, use symmetry to find the quantity:

$$\frac{\partial}{\partial y}\left(\frac{\partial g}{\partial u_y}\right) = \frac{u_{yy}u_x^2 - u_x u_y u_{xy}}{(u_x^2 + u_y^2)^{3/2}}$$

c.) Now write the first variation of energy ∇E.

$$-\frac{u_{xx}u_y^2 - 2u_x u_y u_{xy} + u_{yy}u_x^2}{(u_x^2 + u_y^2)^{3/2}} + 2\lambda(u-f)$$

d.) Write the PDE that you would evolve to minimize the energy E[u].

$$\frac{\partial u}{\partial t} = \frac{u_{xx}u_y^2 - 2u_x u_y u_{xy} + u_{yy}u_x^2}{(u_x^2 + u_y^2)^{3/2}} - 2\lambda(u-f)$$

3.) TV Denoising

In lecture, we presented the Total Variation (TV) energy of an image u as

$$\min E[u|f] = \int_{\Omega} ||\nabla u|| \quad d\vec{x} + \lambda \int_{\Omega} (u-f)^2 d\vec{x}$$

where f is the original (possibly noisy) image and λ is a parameter that controls the relative importance of the terms. The first term is a regularization term that tries to remove all noise and "smooth" the image. The second term is a data fidelity term that tries to keep the resulting image u similar to the original image f. For an appropriate choice of the fidelity weight λ, we can produce a clean denoised image.

In #3 you showed the PDE for TV minimization by steepest descent is

$$\frac{\partial u}{\partial t} = \frac{u_{xx}{u_y}^2 - 2u_x u_y u_{xy} + u_{yy}{u_x}^2}{\left({u_x}^2 + {u_y}^2\right)^{3/2}} - 2\lambda(u-f)$$

To avoid division by zero, we can add a small "fudge factor" 0.1 to the denominator.

$$\frac{\partial u}{\partial t} = \frac{u_{xx}{u_y}^2 - 2u_x u_y u_{xy} + u_{yy}{u_x}^2}{0.1 + \left({u_x}^2 + {u_y}^2\right)^{3/2}} - 2\lambda(u-f)$$

Write a function that performs TV denoising on a grayscale image. The function should take the original image f and the parameter λ as input.

```
function [u] = tv (f, lambda)
```

Note you can copy and paste your curvature code from Lab 5 to compute the large fraction term. You will need to put this code inside a loop, so it is re-computed as the image u evolves.

Try a time step dt=0.1 and stopping time T=20.

Test this code on a noisy grayscale image, such as the "camerman.tif" or "pout.tif" image. Recall you can add Gaussian noise to an image using the `imnoise` command.

```
A_noisy = imnoise (A, 'gaussian', 0, 0.01);
```

Now denoise this image using the fidelity weight λ=0.1.

```
u = tv (A_noisy, 0.1);
```

Now try TV denoising with the values λ=0.01 and λ=1. Experiment with different values of the fidelity weight λ to find the optimal denoised image. (If λ is too large, the method may become unstable.)

a.) What happens when λ is small?

```
The image becomes blury
```

b.) What happens when λ is large?

```
The image becomes more like the noisy image
```