Total Variation Denoising

Imaging Lab 7 - May, 2017

Kamyar Nazeri
Student ID: 100633486

Total Variation Denoising: Color Images

The Rudin-Osher-Fatemi Total Variation (TV) Energy is the most famous and widely used image restoration model and for an input image f is defined as:

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

Where f is the original noisy image and λ is parameter (fidelity weight) that controls the relative importance of the terms. According to *Lagrange-Euler* equation, the first variation of energy of this function is:

$$\nabla E = -\nabla \cdot \left(\frac{\nabla u}{\|\nabla u\|} \right) + 2\lambda(u-f)$$

Where

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

The energy is convex, so we can use steepest descent to evolve the PDE:

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

To apply TV denoising on a color image, we calculate the equation above on each of the 3 RGB channels and storing it in the corresponding channel of a new image. As the first step inside the for loop, we calculate the central differences to approximate the first derivatives u_x and u_y:

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

And to calculate u_{xx} and u_{yy} we perform central differences twice on the input image; to calculate u_{xy} we use diagonal derivative:

$$u_{xx} \approx D_x^0(D_x^0 u) = u(x+1,y) - 2u(x,y) + u(x-1,y)$$
$$u_{yy} \approx D_y^0(D_y^0 u) = u(x,y+1) - 2u(x,y) + u(x,y-1)$$
$$u_{xy} \approx D_x^0(D_y^0 u) = u(x+1,y+1) + u(x-1,y-1) - u(x-1,y+1) - u(x+1,y-1)$$

Finally the PDE becomes:

$$u^{n+1} = u^n + \Delta t \left[\frac{u_{xx}u_y^2 - 2u_x u_y u_{xy} + u_{yy}u_x^2}{(u_x^2 + u_y^2)^{3/2}} - 2\lambda(u-f) \right]$$

We use Forward Euler Method with Neumann boundary conditions to solve the above equation.

Coding Total Variation on Color Images

Listing 1 shows the Total Variation denoising applied on a grayscale image in Matlab; *Listing 2* uses the same function to apply TV on color images, and calculates SNR/RMSE of the output:

```
function [u] = tv(f, lambda)
    % Computes the total variation of an input grayscale image

    dt = 0.1;                   % time step
    T = 20;                     % stopping time
    a = 0.1;                    % fudge factor
    [m,n] = size(f);            % image size
    f = double(f);              % convert to double
    u = f;                      % initialization

    for t = 0:dt:T
        % u_x = (u(x+1,y) - u(x-1,y)) / 2
        u_x = (u(:,[2:n,n]) - u(:,[1,1:n-1])) / 2;

        % u_y = (u(x,y+1) - u(x,y+1)) / 2
        u_y = (u([2:m,m],:) - u([1,1:m-1],:)) / 2;

        % u_xx = u(x+1,y) - 2u(x,y) + u(x-1,y)
        u_xx = u(:,[2:n,n]) - 2 * u + u(:,[1,1:n-1]);

        % u_yy = u(x,y+1) - 2u(x,y) + u(x,y-1)
        u_yy = u([2:m,m],:) - 2 * u + u([1,1:m-1],:);

        % u_xy = (u(x+1,y+1) + u(x-1,y-1) - u(x-1,y+1) - u(x+1,y-1)) / 4
        u_xy = (u([2:m,m],[2:n,n]) + u([1,1:m-1],[1,1:n-1]) - u([2:m,m],[1,1:n-1]) - u([1,1:m-1],[2:n,n])) / 4;

        k_num = (u_xx.*u_y.^2)-2*(u_x.*u_y.*u_xy)+(u_yy.*u_x.^2);
        k_denom = (u_x.^2 + u_y.^2).^(3/2) + a;
        pde = k_num ./ k_denom - 2 * lambda * (u - f);

        u = u + dt * pde;
    end

    u = uint8(u);
end
```

Listing 1: Total Variation function on grayscale images in Matlab

```
function [u, snr, rmse] = colortv(f, lambda)
    u = f;
    for i=1:3; u(:,:,i) = tv(f(:,:,i), lambda); end;
    snr = SNR(rgb2gray(f), rgb2gray(u));
    rmse = RMSE(rgb2gray(f), rgb2gray(u));
end
```

Listing 2: Total Variation + RMSE/SNR on color images in Matlab

Figure 1 shows a test image and three TV donoised images for different values of fidelity weight parameter. As shown below, when fidelity weight is too small the output image becomes blurry; on contrast when fidelity weight is too large, the output image resembles the original noisy input image:

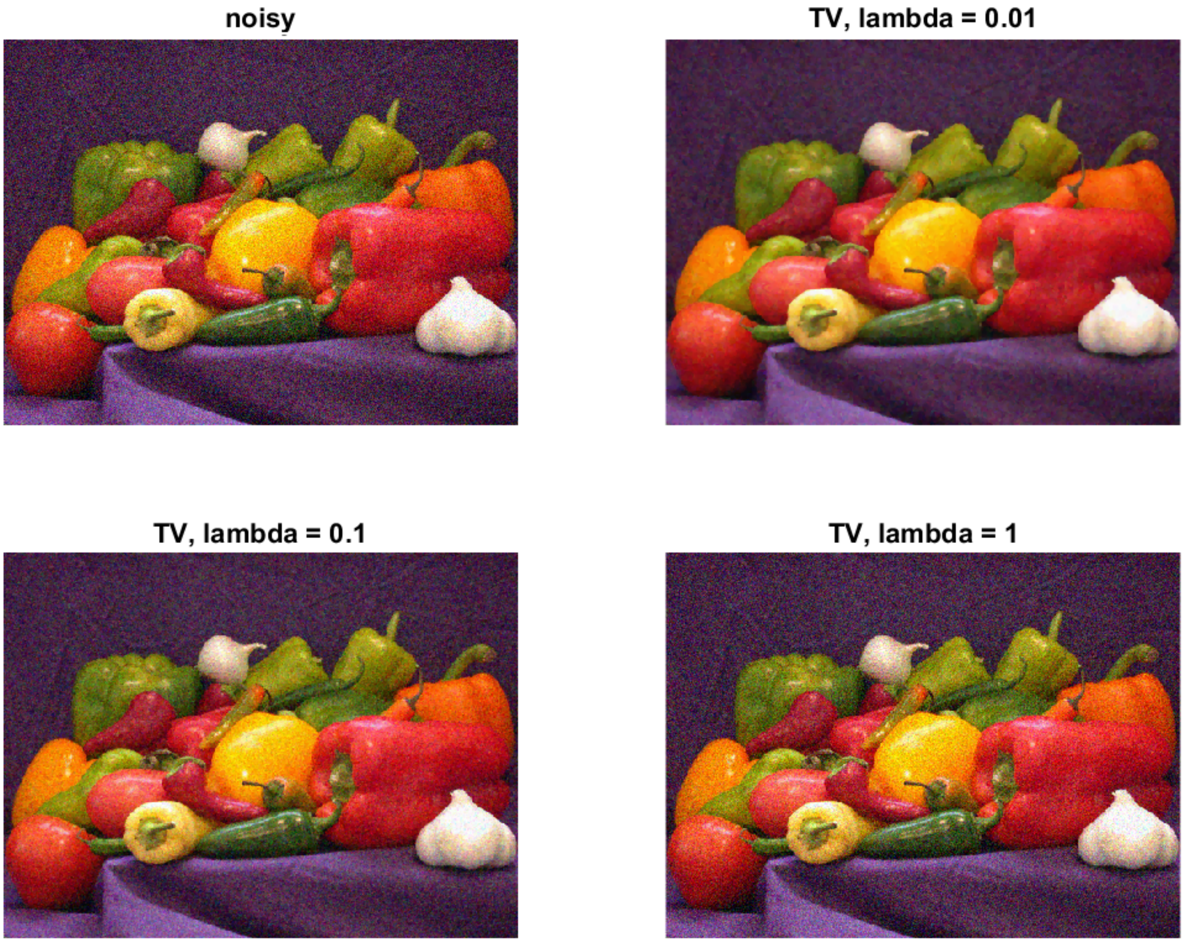

Figure 1: Total Variation denoising applied on a color image with different fidelity weight values.