Eric Murphy

BDA432-Prof. Yuesheng Xu

Project #1 – Image Denoising in MATLAB

A Survey of Two Elementary Image Denoising Techniques

Abstract:

The main challenge when conducting digital image processing is to remove noise from the original source image, if possible. Depending on the level and type of noise, discretion can be used to determine the optimal use case for any given image. For our purposes, I will be conducting a simple exercise in taking an original noiseless image, adding randomized gaussian noise to the image, and performing an A/B experiment based on how optimal each denoising algorithm performs.

We will be looking at two separate images, each with their own unique level of detail, shadowing, and contrast. MATLAB will be used for all executables.

Upon importing the Noiseless first image, we add random noise to the image as follows:

```
%Here we are importing the original noiseless image, adding Gaussian Noise
%at a level of 20, and exporting the image from MATLAB
NoiselessIm1 = imread('C:\Users\emurp010\Documents\BDA432\Images\Cameraman.png');
NoiseLevel1 = 20;
sigma1 = (NoiseLevel1*NoiseLevel1)/(255*255);
NoisyIm1 = imnoise(NoiselessIm1, 'gaussian',0,sigma1);
imshow(NoisyIm1,[]);
```

Which produces the following afterimage. Comparing the noiseless and noisy images :

Our first algorithm we use to denoise is called the ROF algorithm, also known as Total Variation Algorithm. This denoising method is based on total-variation, originally proposed by Rudin, Osher and Fatemi (ROF). In this particular case fixed point iteration is utilized. It is based on the principle that signals with excessive and possibly spurious detail have high total variation, that is, the integral of the absolute gradient of the signal is high. According to this principle, reducing the total variation of the signal subject to it being a close match to the original signal, removes unwanted detail whilst preserving important details such as edges. The regularization parameter lambda plays a critical role in the denoising process. When lambda = 0, there is no smoothing and the result is the same as minimizing the sum of squares. As lambda tends to infinity, however, the total variation term plays an increasingly strong role, which forces the result to have smaller total variation, at the expense of being less like the input (noisy) signal. Thus, the choice of regularization parameter is critical to achieving just the right amount of noise removal.

Matlab code is shown below for the Total-Variation Algorithm. This function gets called in the main script later.

```
function A = ROFdenoise(Im, Theta)
[Image1_h, Image1_w] = size(Im);
g = 1; dt = 1/4; nbrOfIterations = 5;
Im = double(Im);
p = zeros(Image1_h,Image1_w,2);
d = zeros(Image1_h,Image1_w,2);
div_p = zeros(Image1_h,Image1_w);
for i = 1:nbrOfIterations
    for x = 1:Image1_w
        for y = 2:Image1_h-1
            div_p(y,x) = p(y,x,1) - p(y-1,x,1);
        end
    end
    for x = 2:Image1_w-1
        for y = 1:Image1_h
            div_p(y,x) = div_p(y,x) + p(y,x,2) - p(y,x-1,2);
        end
    end
    % Handle boundaries
    div_p(:,1) = p(:,1,2);
    div_p(:,Image1_w) = -p(:,Image1_w-1,2);
    div_p(1,:) = p(1,:,1);
    div_p(Image1_h,:) = -p(Image1_h-1,:,1);
    % Update u
    u = Im-Theta*div_p;
    % Calculate forward derivatives
    du(:,:,2) = u(:,[2:Image1_w, Image1_w])-u;
    du(:,:,1) = u([2:Image1_h, Image1_h],:)-u;
    % Iterate
    d(:,:,1) = (1+(dt/Theta/g).*abs(sqrt(du(:,:,1).^2+du(:,:,2).^2)));
    d(:,:,2) = (1+(dt/Theta/g).*abs(sqrt(du(:,:,1).^2+du(:,:,2).^2)));
    p = (p-(dt/Theta).*du)./d;
end
```

Upon calling this function as follows:

```
ROFDenoisedIm1 = ROFdenoise(NoisyIm1, 16);
imshow(ROFDenoisedIm1,[]);
```

We achieve the following denoised image (left), comparing to noisy original (center), and noiseless original (right):

Denoised with TV(ROF) Noised Image Original Noiseless

Visually, at the expense of removing more and more noise, we will see a slight decline in the sharpness of the original image. For the denoised image, a fairly good result is obtained by using a theta value around 12-16. A possible addition would be to analyze the residual with an entropy function and add back areas that have a lower entropy, i.e. there are some correlation between the surrounding pixels.

PRIMAL DUAL METHOD

In contrast to the 1D case, solving this denoising yields a non-trivial result. A recent algorithm that solves this is known as the primal dual method, or TVL1 algorithm:

```
function newim = TVL1denoise(NoisyIm1, lambda, niter)
  if(nargin<3) || isempty(niter)
    niter=100;
  end
  L2=8.0;
  tau=0.02;
  sigma=1.0/(L2*tau);
  theta=1.0;
  lt=lambda*tau;
  [height width]=size(NoisyIm1);
  unew=zeros(height, width);
  p=zeros(height, width, 2);
  d=zeros(height, width);
  ux=zeros(height, width);
  uy=zeros(height, width);
  mx=max(NoisyIm1(:));
  if(mx>1.0)
    nim=double(NoisyIm1)/double(mx); % normalize
  else
    nim=double(NoisyIm1); % leave intact
  end
  u=nim;
  %[p(:, :, 1), p(:, :, 2)]=imgradientxy(u, 'IntermediateDifference');
  p(:, :, 1)=u(:, [2:width, width]) - u;
  p(:, :, 2)=u([2:height, height], :) - u;
  for k=1:niter
    % projection
    % compute gradient in ux, uy
    %[ux, uy]=imgradientxy(u, 'IntermediateDifference');
    ux=u(:, [2:width, width]) - u;
    uy=u([2:height, height], :) - u;
    p=p + sigma*cat(3, ux, uy);
    % project
    normep=max(1, sqrt(p(:, :, 1).^2 + p(:, :, 2).^2));
    p(:, :, 1)=p(:, :, 1)./normep;
    p(:, :, 2)=p(:, :, 2)./normep;
    % shrinkage
    % compute divergence in div
    div=[p([1:height-1], :, 2); zeros(1, width)] - [zeros(1, width); p([1:height-1], :, 2)];
    div=[p(:, [1:width-1], 1)  zeros(height, 1)] - [zeros(height, 1)  p(:, [1:width-1], 1)] + div;
    %% TV-L2 model
    %unew=(u + tau*div + lt*nim)/(1+tau);
    % TV-L1 model
    v=u + tau*div;
    unew=(v-lt).*(v-nim>lt) + (v+lt).*(v-nim<-lt) + nim.*(abs(v-nim)<=lt);
    %if(v-nim>lt); unew=v-lt; elseif(v-nim<-lt) unew=v+lt; else unew=nim; end
    % extragradient step
    u=unew + theta*(unew-u);
    %% energy being minimized
    % ux=u(:, [2:width, width]) - u;
    % uy=u([2:height, height], :) - u;
    % E=sum(sqrt(ux(:).^2 + uy(:).^2)) + lambda*sum(abs(u(:) - nim(:)));
    % fprintf('Iteration %d: energy %g\n', k, E);
  end
  newim=u;
```

Which yields the denoised result (left). Compared to ROF (center) and Gaussian Noised (right):

This produces the best result of the three as far as **STRICTLY NOISE REMOVAL**. However, it reduces sharpness considerably.

Lena, obtains similar denoised results:

TVL1 Denoised ROF Denoised Noisy Original

Conclusion:

The Total Variation noise removal technique has advantages over simple techniques such as linear smoothing or median filtering which reduce noise but at the same time smooth away edges to a greater or lesser degree. By contrast, total variation denoising is remarkably effective at simultaneously preserving edges whilst smoothing away noise in flat regions, even at low signal-to-noise ratios. For this method, the regularization parameter plays an important role in the amount of smoothing for any given noisy image. However, like with the Primal- Dual algorithm, the tradeoff is that for more noise removal, you run the risk of reducing the sharpness of the image. As far as total smoothing, the Primal- Dual algorithm is a superior choice if the goal is to reduce the L1 Norm.

Sources:

ROF Denoising Algorithm

https://www.mathworks.com/matlabcentral/fileexchange/22410-rof-denoising-algorithm

Authored by Carl Löndahl.

TV-L1 Image Denoising Algorithm

https://www.mathworks.com/matlabcentral/fileexchange/57604-tv-l1-image-denoising-algorithm

Authored by Manolis Lourakis

Total Variation Denoising

https://en.wikipedia.org/wiki/Total_variation_denoising