

# Programming Assignment: Image Restoration

Marek Rychlik

April 13, 2025

## 1 Objective

This assignment explores the the image restoration technique ROF in MATLAB:

- **Reading and interpretation of raw image data** – Develop understanding about digital images as they are collected by sensors (CCD); how they are created, represented and converted to known RGB model.
- **Non-linear PDE discretization** – Developing and implementing an algorithm for solving the discretized ROF equation.
- **Measuring noise in images using ROF** - Analyzing statistics of the difference between and image and its smoothed version, using ROF.

## 2 Problem Statement

The *regularized ROF* is given by the functional:

$$\mathcal{F}(u) = \int \sqrt{\epsilon^2 + |\nabla u|^2} + \frac{\lambda}{2} \int (u - f)^2 dx dy$$

where  $u$  is the image to be restored,  $f$  is the observed degraded image, and  $\lambda$  is a regularization parameter.

The Euler-Lagrange PDE derived from this functional is:

$$\frac{u - f}{\lambda} - \frac{\partial}{\partial x} \frac{u_x}{\sqrt{\epsilon^2 + u_x^2 + u_y^2}} - \frac{\partial}{\partial y} \frac{u_y}{\sqrt{\epsilon^2 + u_x^2 + u_y^2}} = 0$$

The Neumann Boundary Conditions on a rectangular domain  $[a, b] \times [c, d]$  are:

$$\begin{aligned} \frac{\partial u}{\partial x}(a, y) = 0 \quad \text{and} \quad \frac{\partial u}{\partial x}(b, y) = 0, \\ \frac{\partial u}{\partial y}(x, c) = 0 \quad \text{and} \quad \frac{\partial u}{\partial y}(x, d) = 0. \end{aligned}$$

Develop a difference scheme (by either discretizing the functional  $\mathcal{F}$  or the PDE) which provides a faithful representation of the ROF technique. Note that the input data is the discretized fersion of the function  $f$  - the observed degraded image. The degradation is a result noise during collection of the data, due to the underlying physics (counting photons, analogue-to-digital conversion, etc.).

The input data is a single image `DSC00099.ARW`, containing a *Bayer mosaic*, with RGGB CFA layout. The synopsis of basic I/O operations on such image data is in the following script:

```
1 %-----
2 % File:      basic_script.m
3 %-----
4 %
5 % Author:    Marek Rychlik (rychlik@arizona.edu)
6 % Date:      Mon Apr  7 14:19:32 2025
7 % Copying:   (C) Marek Rychlik, 2020. All rights reserved.
8 %
9 %-----
10 % Basic operations on raw images
```

```

11 % Source image: https://www.reddit.com/r/EditMyRaw/comments/1jt4ecw/
    the_official_weekly_raw_editing_challenge/
12 raw_img_filename=fullfile('.', 'images', 'DSC00099.ARW')
13 %raw_img_filename=fullfile('.', 'images', 'credit @signatureeditsco - signatureedits.com
    _DSC4583.dng')
14 t=tiledlayout('TileSpacing','compact','Padding','compact');
15 linked_axes=[]
16 cfa=rawread(raw_img_filename);
17 ax=nexttile, imagesc(bitand(cfa, 7)),title('xor with 7');
18 info = rawinfo(raw_img_filename);
19 disp(info);
20 Iplanar=raw2planar(cfa);
21 planes='rggb';
22 ang = info.ImageSizeInfo.ImageRotation;
23 for j=1:size(Iplanar,3)
24     ax=nexttile; imagesc(imrotate(Iplanar(:,:,j),ang)), title(['Plane ',num2str(j),': ',
        planes(j)]), colorbar, colormap gray
25     linked_axes=[linked_axes,ax];
26 end
27 Idemosaic=demosaic(cfa,planes);
28 % nexttile, image(Idemosaic), colorbar;
29 Irgb=raw2rgb(raw_img_filename);
30 % Resized RGB image, so that size matches the size of the planes
31 ax=nexttile; image(imresize(Irgb,.5)), colorbar,
32 title('Demosaiced and scaled RGB image');
33 linked_axes=[linked_axes,ax];
34 linkaxes(linked_axes);

```

Listing 1: Processing of raw images

Your program will operate separately on the planar data (R,G,G,B) to compare the differences in the level of noise of different colors. The entire mosaic is a 2D array, 4024 by 6048. Every plane is half that size. Thus, your implementation of ROF will operate on a “grayscale” image of size 2014 by 3025. This is the array  $f_{i,j}$  which approximates  $f$ . Note that

$$f_{i,j} = u(x_j, y_i) \quad \text{Not } u(x_i, y_j)!!!$$

To evaluate the noise level, you should study the statistics of the difference  $u - f$  as function of the parameters  $(\lambda, \epsilon)$ . The measure of noise will be the *mean square difference*:

$$MSD(f, \lambda, \epsilon) = \sqrt{\frac{\sum_{i=1}^H \sum_{j=1}^W (u_{i,j} - f_{i,j})^2}{H \cdot W}}$$

where  $H$  and  $W$  are the image height and width, respectively, and  $u$  is the solution to the ROF problem.

You should graph  $MSD$  over a range that best illustrates the differences between the level of noise in the planes R, G, G, B.

## 3 Tasks

### 3.1 Design and implement the difference scheme for ROF

- A MATLAB function `smooth_image_rof` ( implemented in the MATLAB function file `smooth_image_rof.m`) with signature given below, solving the discretized problem.

```

1  u = smooth_image_rof(f, lambda, epsilon)
2  % SMOOTH_IMAGE_ROF - perform ROF image restoration
3  % U = SMOOTH(F, LAMBDA, EPSILON) - perform ROF image restoration
4  % Arguments:
5  %   F - the degraded image
6  %   LAMBDA - the 'smoothing' parameter (scalar or vector)
7  %   EPSILON - the 'regularization' parameter (scalar or vector)
8  % Returns:
9  %   U - the restored/smoothed image
10 % If LAMBDA or EPSILON is a vector, the result should be a 4D array of size
11 % H-by-W-by-K-by-L, where H and W are the height and width of F
12 % and K and L are the lengths of LAMBDA and EPSILON, respectively.

```

Listing 2: Processing of raw images

- A MATLAB function `calculate_msd` (in the file `calculate_msd.m`) which yields the measure of noise in  $f$ . The signature of the function is:

```

1  msd = calculate_msd(f, lambda, epsilon)
2  % CALCULATE_MSD - returns MSD for a given degraded image and ROF parameters
3  % MSD = CALCULATE_MSD(F, LAMBDA, EPSILON) - find MSD
4  % Arguments:
5  %   F - the degraded image
6  %   LAMBDA - the 'smoothing' parameter (scalar or vector)
7  %   EPSILON - the 'regularization' parameter (scalar or vector)
8  % Returns:
9  %   MSD - the MSD of the degraded image (scalar or 1D array)
10 % If LAMBDA or EPSILON is a vector, the result should be an array of size
11 % K-by-L, where K and L are the lengths of LAMBDA and EPSILON, respectively.

```

Listing 3: Processing of raw images

You should call `smooth_image_rof` in this function.

Note that both functions must be fully vectorized. Thus, if parameters `lambda` and `epsilon` are vectors, `smooth_image_rof` must return a family of restored/denoised images. The result is a 4D array. Similarly, `calculate_msd` should return a 2D array in this situation, representing the values of the function  $MSD(f, \lambda, \epsilon)$  over the grid  $\lambda \times \epsilon$ . Note that this allows easy plotting using the `meshgrid` function of MATLAB. This will be helpful in constructing the report.

## 3.2 Support for GPU and multiple CPU

Your code should use GPU to accelerate calculations when available. The function `gpuArray` fails if there is no GPU device available on the machine on which the program runs. Unfortunately, the server on which your code is tested does not have a compatible GPU (it is an AMD machine with RADEON graphics card; only NVIDIA graphics cards are supported by MATLAB). Also, you can call `gpuDevice` to test if a GPU is available.

If there is no GPU, your program should try to use multiple CPU. You can detect the number of computational threads by running `maxNumCompThreads`. On the machine on which your program will be tested, the number of computational threads is 4. It is a requirement that your program uses multiple threads.

# 4 Deliverables

## 4.1 Code Implementation

- MATLAB function files `smooth_image_rof.m` and `calculate_msd.m`.
- Other files that your implementation may depend on.

## 4.2 Plots and Analysis

- For the 4 color planes of the supplied image, R, G, G, B, plot the function:

$$(\lambda, \epsilon) \mapsto MSD(f, \lambda, \epsilon)$$

These should be 4 surfaces stacked one upon another. Make sure the surfaces are semi-transparent not to obstruct each other. Pick a good region of parameters.

- Make a definite inference about which color planes are the most and least noisy.
- Speculate (in an educated way) why there are two green planes in the Bayer mosaic.

## 4.3 Written Report

- Brief explanation of methods.
- Interpretation of results.
- Discussion of noise differences between the color planes.