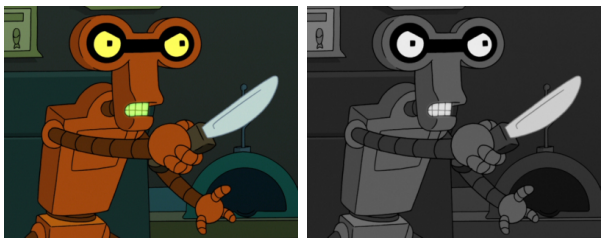
Practice 1
Image Processing with Partial Differential Equations

Kazi Ahmed Asif Fuad and Santiago Herrero Melo

I. INTRODUCTION

During this practice, we will work in the MATLAB environment with grayscale images, applying different functions in order to retrieve noisy images. Afterwards, we will also employ the *Marr_Hildreth* method in order to obtain the edges of the images.
The image selected for developing this practice has been the *roberto.jpg* image, shown in image (6a). As we have to work with grayscale images, we have to turn it from RGB to grayscale, as shown in image (6b).



(a) Original image (b) Grayscale image

Fig. 1: Original and grayscale images

II. EXPERIMENTATION AND RESULTS

A. Denoising with PDEs

In order to work with the selected image, we will add some Gaussian noise to it. For doing so, we will use the provided function textitadd_gaussian_noise.m, which adds a random noise to the image. By default, we will use the provided standard deviation (30) in the instructions of the lab to generate the noise. The noise image can be seen in Figure (2)

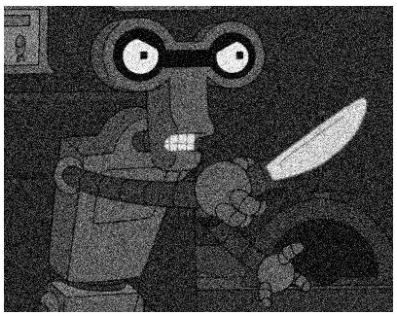


Fig. 2: Image with noise

1) Heat equation: The heat equation is a PDE that realizes a spatial diffusion of the gray values of a given image, denoted by the equation (1).

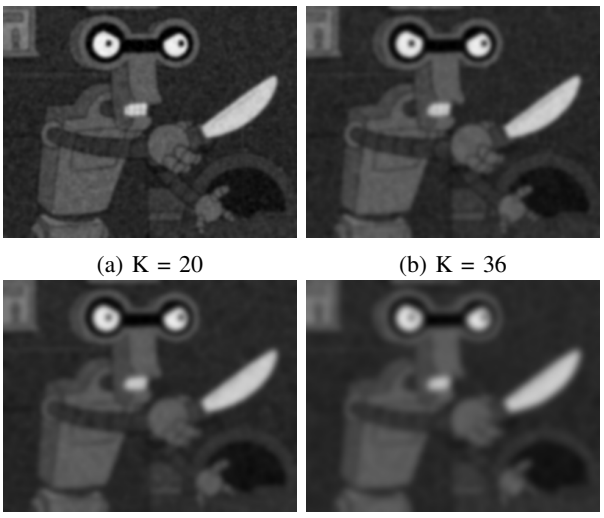
$$\begin{cases} \frac{\delta u(t,x)}{\delta t} = \Delta u(t.x) & \text{for t} \geq 0 \text{ and x} \in \Omega \\ u(0,x) = f(x) & \text{for x} \in \Omega \\ \frac{\delta u(t,x)}{\delta N} = 0 & \text{for t} > 0 \text{ and x} \in \delta\Omega \end{cases} \tag{1}$$

In order to simplify these equations, we use the equation 2. For its calculation, we are provided the scripts *gradx.m*, *grady.m* and *div.m*.

$$\Delta u = div(\nabla u) \tag{2}$$

Once we have the value of the Laplacian, it is very simple to obtain the denoised image by the heat equation method, using equation 3, which means that we will modify each obtained image, starting from the original image, according to the obtained Laplacian value and a defined time step (which we will take as value 1/8 as recommended by the instructions). This process will be performed a defined amount of times, obtaining different values, as can be seen in the images in Figure (3). We established the optimal amount of iterations (K_{opt}) to 36, which corresponds to image (3b).

$$\begin{cases} u_{i,j}^{k+1} = u^k + \delta t(\Delta u^k)_{ij} \\ u_{i,j}^0 = f_{ij} \end{cases} \tag{3}$$



(a) K = 20 (b) K = 36

(c) K = 60 (d) K = 100

Fig. 3: Denoised images with heat equation

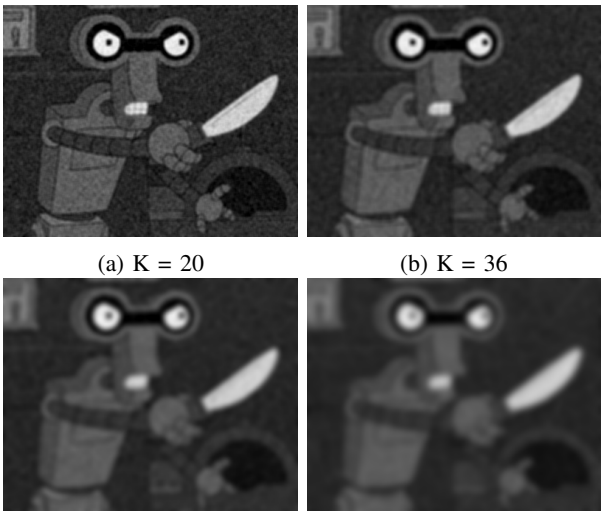
2) Convolution with a Gaussian kernel: This presents a different method for denoising the image. This method uses a Gaussian kernel of dimension 2, defined by equation 4.

$$G_\sigma(x) = \frac{1}{2\pi\sigma^2} exp(-\frac{|x|^2}{2\sigma^2}) \tag{4}$$

The size of this kernel is defined by the number of iterations and the time step, getting the parameter P in equation 5, defining the kernel as a square matrix of size (2P-1) by (2P-1). The standard deviation will be $\sqrt{2K\delta_t}$.

$$P = \lfloor K\delta_t + 1 \rfloor \tag{5}$$

When denoising the noise image with this method, using the same iterations as in the heat equation previously, we obtain the results in Figure (5).



(a) K = 20 (b) K = 36

(c) K = 60 (d) K = 100

Fig. 4: Denoised images with gaussian kernel

As we can see, compared with the images obtained with the Heat Equation method, this outputs are neater, but still removing the noise properly. The more iterations, the close the images are, so we can state that the performance of the heat equation is far from the one with the Gaussian kernel, obtaining better results this last method.

3) Application to contour detection: In this section we are going to look for the significant contours of the image. As we stated previously, the results with the Gaussian kernel are better than with the Heat equation, so we will use this method in order to denoise the image.

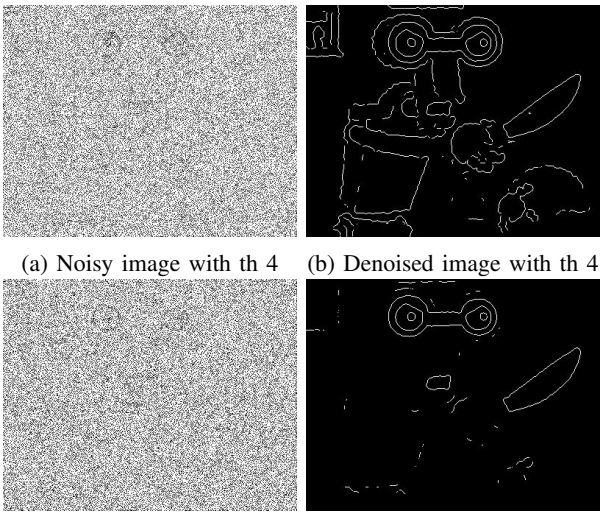
For obtaining the significant contours we will use the Marr Hildreth equation, which is a combination of two criteria:

The first criteria consists con calculating the gradient norm of each pixel, in order to select the pixels of high gradient. For doing so, as illustrated in equation 6, the gradients should be over the value of a determined threshold (η) which should be strictly positive.

$$||\nabla u(i,j)|| > \eta > 0 \tag{6}$$

The second criteria is to find the points in which the laplacian changes its location. In these points, there is a local extrema of the gradient. For detecting this contours, we will use the provided script *change_sign.m*.

When we find a point in which both criteria are true, then we consider it an edge point.



(a) Noisy image with th 4 (b) Denoised image with th 4

(c) Noisy image with th 8 (d) Denoised image with th 8

Fig. 5: Edges results with Marr Hildreth at different thresholds of Noisy and Denoised Images

The denoising has been performed with the heat equation. The value of $\eta = 4$ is quite good as shown in image (8b), so from now on it will be considered as the optimal threshold value for the denoised image. If we use a higher threshold we are asking for a higher gradient, so too many pixels will be ignored, as shown in image (8d).

We have also tested the code in the noisy images (images (8a) and (8c)). As it can be seen, the high amount of noise makes the gradient very high, so many points are mistakenly considered as edges, returning a very noisy image with no defined shapes in it.

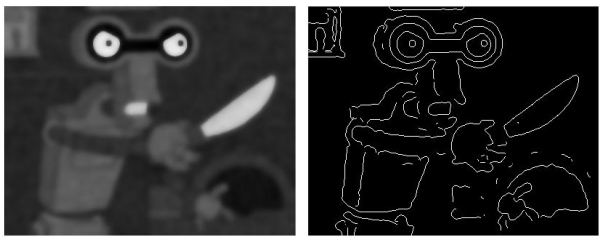
B. Perona-Malik

1) Basic algorithm: This equation is a modification of the heat equation (1) in order to enhance the results obtained with it. For doing so, Perona and Malik proposed the modifications of the first part of the equation 7, where g is a decreasing function that has the form 8, where α is a coefficient that is considered a function of the noise in the image.The Perona-Malik PDE is thus an anisotropic diffusion since specific directions are discouraged.

$$\frac{\delta u(t,x)}{\delta t} = div(g(||\nabla u(t,x)||)\nabla u(t,x)) \tag{7}$$

$$g(\epsilon) = \frac{1}{\sqrt{(\epsilon/\alpha)^2 + 1}} \tag{8}$$

In Fig. 6, Contour Detection of denoised image using Perona Malik with Marr Hildreth is given. From the figures, we can see that Perona-Malik PDE detects contour more accurately than the previous implementation of Heat Equation and Gaussian Kernels as Perona-Malik preserves the discontinuities better in the final images (u_k).



(a) Denoising of the image with Perona Malik algorithm (b) Contour detection with Marr Hildreth

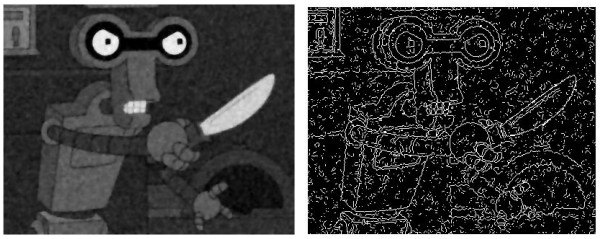
Fig. 6: Denoising and Contour detection with Perona Malik and Marr Hildreth at K=80 and mu=2

2) Enhancement with a convolution of the gradient with a Gaussian kernel: As previously observed, when the original image f is very noisy, its gradients present strong oscillations that perturb the diffusion. The noise may indeed be considered as a significant contour. A simple approach to circumvent this issue is to smooth the gradient that is send to the function c. The improved model is given at equations 9 and 10.

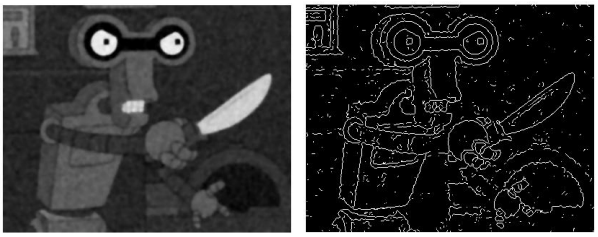
$$A = div(g(||G_\sigma * \nabla u(t,x)||)\nabla u(t,x)) \tag{9}$$

$$\begin{cases} \frac{\delta u(t,x)}{\delta t} = A & \text{, t} \geq 0 \text{ and x} \in \Omega \\ u(0,x) = f(x) & \text{, x} \in \Omega \\ \frac{\delta u(t,x)}{\delta N} = 0 & \text{, t} > 0 \text{ and x} \in \delta\Omega \end{cases} \tag{10}$$

Compared to previous implementation of Perona-Malik, Enhancement with a convolution of the gradient with a Gaussian provides better result for less number of iteration K. We have presented our result in 7 and 8 for K=20 and K=80 respectively. From the figures we can come to conclusion that, at K=20, Enhancement with Gaussian Kernel performs better as it is less prone to noise. But as we increase the smoothing by increasing the number of iteration of K to 80, the image is smoothed a lot where the performance of Perona-Malik and its Enhancement with Gaussian Kernel is almost same. Beside, Enhancement of Perona-Malik with Gaussian Kernel is computation power efficient as it can produce better result in less number of iteration K The σ is dependent of number of iteration K and δt. As we kept δt constant, σ varied only based on K.

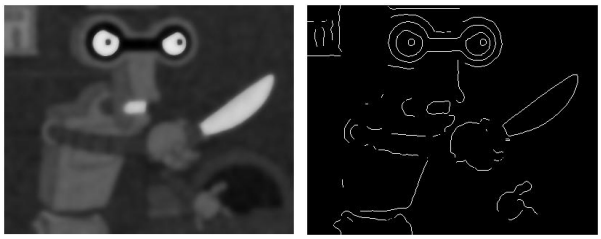


(a) Denoised with Perona-Malik (b) Contour with Marr Hildreth

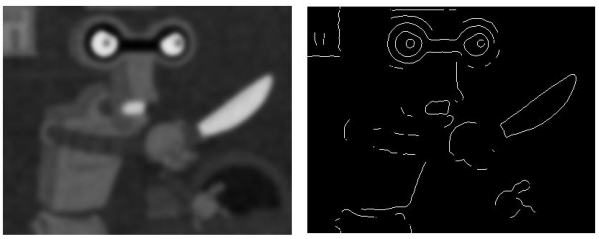


(c) Denoised with Perona-Malik Gaussian (d) Contour with Marr Hildreth

Fig. 7: Denoising with Perona-Malik and Perona-Malik Gaussian and Contour detection with Marr Hildreth at K=20



(a) Denoised with Perona-Malik (b) Contour with Marr Hildreth



(c) Denoised with Perona-Malik Gaussian (d) Contour with Marr Hildreth

Fig. 8: Denoising with Perona-Malik and Perona-Malik Gaussian and Contour detection with Marr Hildreth at K=80

REFERENCES

[1] Practice 1: Image processing with Partial Differential Equations Manual by Jean-François Aujol and Nicolas Papadakis, IMB, Université Bordeaux.