MSEL
Modelling in Science Education and Learning

Modelling in Science Education and Learning
Volume 2, No. 5, 2009.
Instituto Universitario de Matemática Pura y Aplicada

Derivative-based optimization in colour image filtering: an application for derivative learning

A. Sapena, S. Morillas
Universidad Politécnica de Valencia
alsapie@mat.upv.es
smorillas@mat.upv.es

José Camacho
Universitat de Girona
jcamacho@isa.upv.es

Abstract

La noción de derivada de una función y su aplicación a la optimización de funciones es un problema interesante e ilustrativo para los estudiantes de ingeniería. En este trabajo, desarrollamos un aplicación del concepto de derivada para la optimización del filtrado de imágenes en color. Ello implica ajustar el parámetro del filtro para obtener un rendimiento óptimo de filtrado. Proponemos maximizar la calidad de la imagen filtrada representada por la relación señal-ruido (Peak Signal-to-Noise Ratio, PSNR), que es una función del parámetro del filtro. El valor óptimo del parámetro se obtiene mediante un algoritmo basado en la aproximación de la derivada de la función del PSNR de manera que se obtenga la imagen filtrada óptima.

Related to the notion of derivative of a function, its application to function optimization is an interesting and illustrative problem for Engineering students. In the present work, we develop an application of the derivative concept to optimize the filtering of a colour image. This implies to optimize the value of the filter parameter to maximize performance. We propose to maximize the quality of the filtered image represented by the Peak Signal to Noise Ratio *(PSNR), which is a function of the filter parameter. The optimal value for the parameter is obtained by means of an algorithm based on the approximation of the derivative of the PSNR function so that finally the optimum filtered image is obtained.*

Keywords: *Derivative, optimization, image filtering*

1 Introduction

The derivative is a measure of how a function changes as its inputs change. For instance, the derivative in time of the position of a runner is the velocity of the runner. The velocity tell us how much the position of the runner is changing in time. Also, the acceleration is the derivative of the velocity. It tell us how much the velocity is changing in time. The derivative of a function for given input values is the best linear approximation for close input values. Take the example shown in Figure 1. In this figure, a function and its derivatives for two input values, specified by A and B, are represented.

The area in which the derivative of a function is a good linear approximation of this function is variable. It depends very much on the degree of non-linearity of the function around the point where the derivative is defined. For instance, the derivative in point A of Figure 1 is only a good approximation of the function for input values very close to A, whereas the derivative in point B is a good approximation of the function for a much wider interval of input values. The approximation of a function by its derivative is commonly termed linearization and is one of the pillars in modern control systems [15].

Another application of the derivative, which is the one of interest here, is the optimization of a function. The derivative tell us how to modify the input values in order to improve -rise or low- the output. For instance, imagine that it is desired to find the input value where the function in Figure 1 is maximum. If we are in point B, the derivative tell us that we need to reduce the input in order to achieve larger values of the function. If we are in point A, the derivative is 0. As we are in a maximum, this means this is already the input we were looking for. This capability can be used to design a derivative-based optimization algorithm. This algorithm is able to reach the optimum -maximum or minimum- of a function automatically.

The idea of the optimization algorithm is similar to the way a climber climbs a mountain when the top cannot be seen[1]. Every time the climber needs to decide the direction to continue, he or she would select the direction of maximum slope. This approach will lead him or her to the top, provided there is one single top in the mountain. This idea is depicted in Figure 5.2 for a mountain-like function y with two inputs x_1 and x_2. The algorithm starts from an initial point, representing a specific value of input values $[x_1^0, x_2^0]$. The derivative of the function is computed in that point an the algorithm moves forward to point $[x_1^1, x_2^1]$. This operation is repeated several times until the algorithm arrives to the maximum of the function, where the

[1]Another valid example is how water falls from the top of a mountain.

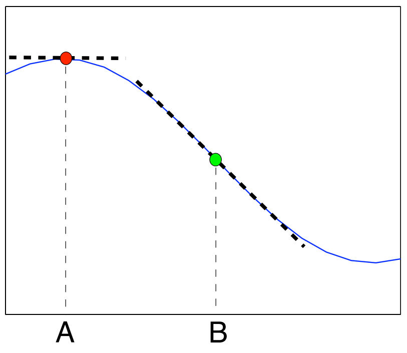


Figure 5.1: Function and derivatives in point A and B.

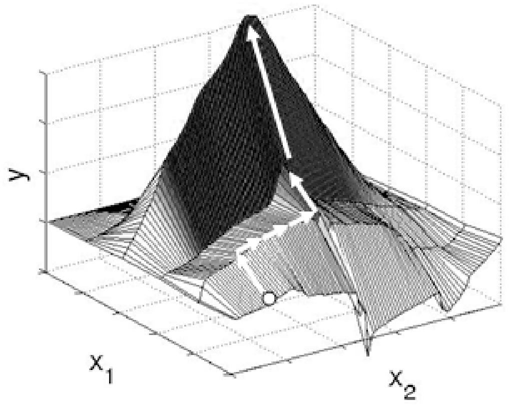


Figure 5.2: Illustration of a derivative-based optimization.

derivative is 0. The idea beneath the derivative-based optimization has been used in a wide number of applications, including the optimization of industrial processes [16], the design of self-tuning control systems [14] and many others.

In this paper, the use of derivatives for the optimized filtering of images is illustrated. In Section 2, the image filtering is introduced and posed as an optimization problem and the filtering method applied is described. The optimization algorithm based on the approximation of derivatives is explained in Section 3. Section 4 shows simulation results and conclusions are provided in Section 5.

2 Image filtering as an optimization problem

An image is a $2D$ representation of the objects in a $3D$ scene which is obtained by projection of the $3D$ objects on an image plane. In digital cameras, this projection is obtained following to the pinhole camera model [1]-[3] (see Figure 5.3(a)). According to this model, the objects, represented in a $3D$ space, are projected through a center of projection on the $2D$ image plane. The center of projection corresponds to the camera optics. The image plane is physically represented as a Charge Coupled Device (CCD) sensor which captures the input radiation. This CCD sensor (see Figure 5.3(b)) comprises an array of $M \times N$ single light sensors. Thus, when a photograph is taken using a digital camera, a representation of the 3D world is captured in an array of $M \times N$ single elements called *pixels* (from picture elements).

In digital colour images, each pixel represents a single colour of the image. Colours are commonly represented in computers using the Red-Green-Blue (RGB) colour space. This colour space follows an additive model so that any colour is obtained by appropriately mixing the three primaries: Red, Green and Blue. Thus, each colour image pixel is associated with a tern of RGB values which represents the appropriate quantity of each primary colour that should be mixed to obtain the colour stored in that pixel. Form a geometrical point of view, each colour can also bee understood as a point in a three-dimensional space, which coordinates are the quantities corresponding to each of the primary colours.

Digital image acquisition process can be affected by many different factors which may degrade the quality of the image. For instance, deficient illumination conditions or CCD sensor malfunctions may introduce irregularities in the image also known as (Gaussian) *noise*. Other

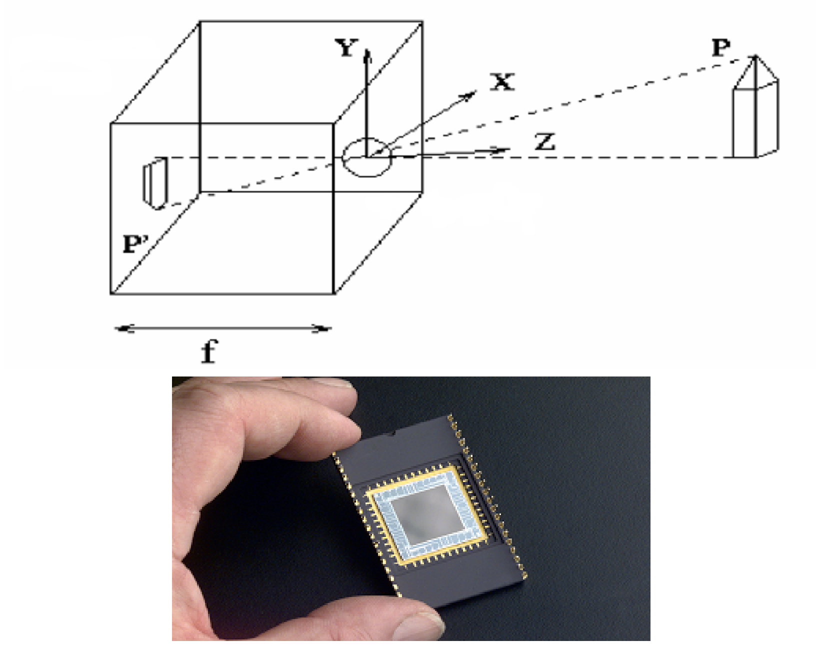


Figure 5.3: (a) Scheme of the pinhole camera model: The object P, represented in the (X, Y, Z) space, is projected through the center of projection (camera optics) on the image plane (CCD) as the object P'. (b) A CCD sensor.

factors able to affect the image quality are, among others, transmission errors or storage faults. Figure 5.4 shows an ideal noise-free image and the same image contaminated with noise.

The presence of irregularities, or noise, in an image is undesirable. This is mainly for two reasons: (i) the perceptual quality of the image is lower, which is critical from the user standpoint, and (ii) the presence of noise is an important drawback for many tools of computer image analysis. Therefore, many techniques to reduce image noise have been developed in the recent years [2, 3]. Classical techniques to approach this problem are based on a linear approach. The Arithmetic Mean Filter (AMF) and the Gaussian Filter (GF) use an average operation among each pixel and its neighbours pixels [2, 3]. This average operation smooths the image, successfully reducing the image noise. Unfortunately, at the same time, the edges and details in the image are blurred. The loss of details also affects the perceptual quality of the image, so that the performance of linear filters may not be satisfactory in many cases. To overcome this, a series of nonlinear methods have been developed [2, 3],[4]-[8]. The basic idea in these techniques is to identify image edges and details. That way, these important parts in an image are smoothen less than the rest of image regions.

The techniques in [5]-[8] propose to average each image pixel with only those neighbour pixels which are similar to it. This is a very efficient method to preserve edges in the image. Nonetheless, it is difficult to differentiate between similar and non-similar pixels. In other words: when a pixel becomes similar to another one is a very subjective matter. A more objective alternative is to define a degree of similarity between two pixels. For instance, coming back to the geometrical interpretation of the colours, the similarity degree may be related to the distance between the colours of two pixels in the RGB space.

In this work, the Fuzzy Bilateral Filter (FBF) [8] will be considered. The FBF employs fuzzy logic to assign the degree of similarity between a pair of pixels. According to fuzzy logic, the degree of similarity between two pixels is a value in $[0, 1]$ that represents in which degree both

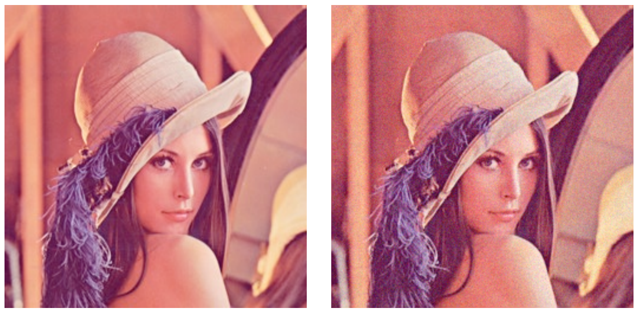


Figure 5.4: (a) Ideal noise-free image, (b) image corrupted with (Gaussian) noise.

pixels are similar, where 1 means total similarity (equality) and 0 total dissimilarity. Since Lofti A. Zadeh introduced the theory of fuzzy sets in 1965, it has become a well-known area of study in the last decades. Fuzzy logic constitutes a generalization of the classical set theory which represents a gradual transition between the classical notions of outside and inside of a set. Fuzzy logic has been successfully employed in many engineering problems. Furthermore, many related research topics as, for instance, fuzzy topology and fuzzy metrics [9]-[13], are still active.

Most nonlinear filters, as it is the case of the FBF [8], have a number of parameters that need to be tuned. The behaviour of the filter, in terms of identification of the details and smoothing capability, can be modified by tuning these parameters. Thus, appropriate tuning is needed to achieve optimal performance. In this paper, the use of derivatives to find an appropriate setting of the FBF [8] parameter is illustrated. For this, the image quality is assessed using the *Peak Signal to Noise Ratio* (PSNR). This index represents the quality of an image, the larger its value the more the quality of the image. Thus, the aim is to find the parameter of the FBF which provides the filtered image of best quality in terms of PSNR. To find this 'optimum' parameter, the derivative of the PSNR function is very useful, as it will be illustrated.

2.1 Fuzzy Bilateral Filtering

In the following, we detail the filtering process applied by the FBF over the image denoted by $\mathbf{F}$. Each image pixel $\mathbf{F_i}$ is characterized by a RGB colour vector $(F^1_{\mathbf{i}}, F^2_{\mathbf{i}}, F^3_{\mathbf{i}})$ and by its location in the image $\mathbf{i}$, which is expressed as a two component vector $\mathbf{i} = (i_1, i_2)$ indicating the row and column of the image where $\mathbf{F_i}$ is located. The objective of the filter is to process the noisy image $\mathbf{F}$ to obtain a filtered image $\mathbf{F}^\lambda$. For each noisy pixel $\mathbf{F_i}$, we obtain the corresponding denoised pixel $\mathbf{F}^\lambda_{\mathbf{i}}$ by means of a weighted averaging of $\mathbf{F_i}$ and its 8 closest neighbour pixels $\mathbf{F_j}$ within a 3×3 filtering window W centered on $\mathbf{F_i}$. Then, to obtain $\mathbf{F}^\lambda_{\mathbf{i}}$ we just have to determine the weight of $\mathbf{F_i}$ and each $\mathbf{F_j} \in W$ to perform the weighted averaging.

To compute these weights we wish to take into account both the colour similarity and the spatial distance between the vectors. For this we will use fuzzy values to represent the degree in which two pixels are similar in colour and spatially close in the image.

For the colour similarity between the colour vectors of the pixels $\mathbf{F_i}$ and $\mathbf{F_j}$, we compare their three RGB components by means of the following fuzzy metric

$$M(\mathbf{F_i}, \mathbf{F_j}) = \prod_{s=1}^{3} \frac{min\{F^s_{\mathbf{i}}, F^s_{\mathbf{j}}\} + K}{max\{F^s_{\mathbf{i}}, F^s_{\mathbf{j}}\} + K}, \tag{5.1}$$

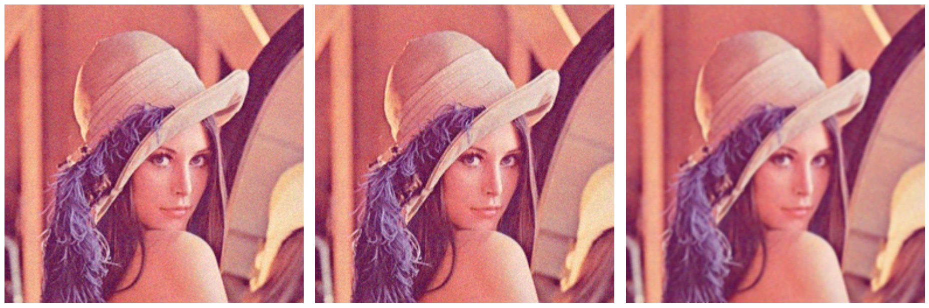


Figure 5.5: Comparative filtering results for two different values of the parameter λ: (a) Lenna noisy image; (b) result with $\lambda = 0.1$; (c) result with $\lambda = 10$.

where K is constant and equal to 1024 [8]. $M(\mathbf{F_i}, \mathbf{F_j})$ provides the fuzzy degree in $[0, 1]$ in which $\mathbf{F_i}$ and $\mathbf{F_j}$ are similar in colour.

In the case of the spatial closeness between pixels $\mathbf{F_i}$ and $\mathbf{F_j}$, we should take into account their locations $\mathbf{i}$ and $\mathbf{j}$ in the image. The following fuzzy metric is used to compute the fuzzy degree in $[0, 1]$ in which $\mathbf{F_i}$ and $\mathbf{F_j}$ are spatially close.

$$M_{||\cdot||_2}(\mathbf{i}, \mathbf{j}, \lambda) = \frac{\lambda}{\lambda + ||\mathbf{i} - \mathbf{j}||_2}, \tag{5.2}$$

where $||\mathbf{i} - \mathbf{j}||_2$ denotes the spatial Euclidean distance computed as

$$||\mathbf{i} - \mathbf{j}||_2 = \sqrt{(i_1 - j_1)^2 + (i_2 - j_2)^2}, \tag{5.3}$$

and λ is used to adjust the importance of the spatial closeness criterion.

To take both criteria into account simultaneously, we make use of a fuzzy metric made up by the combination of these two fuzzy metrics so that the weight of each pixel $\mathbf{F_j}$ is given by

$$CFM(\mathbf{F_i}, \mathbf{F_j}, t) = M(\mathbf{F_i}, \mathbf{F_j}) \cdot M_{||\cdot||_2}(\mathbf{i}, \mathbf{j}, \lambda) \tag{5.4}$$

In this expression, λ is the filter parameter.

Finally, the denoised pixel $\mathbf{F_i^\lambda}$ is calculated as

$$\mathbf{F_i^\lambda} = \frac{\sum_{\mathbf{F_j} \in W} CFM(\mathbf{F_i}, \mathbf{F_j}, t)\mathbf{F_j}}{\sum_{\mathbf{F_j} \in W} CFM(\mathbf{F_i}, \mathbf{F_j}, t)}. \tag{5.5}$$

The parameter λ is used to tune the importance of the spatial closeness criterion respect to the colour similarity criterion. Notice that, when $\lambda \to 0$ the sensitivity to the spatial criterion is very high and so little smoothing is performed. On the other hand, when $\lambda \to \infty$ the FBF gives no importance to the spatial criterion and so the image is too blurred since all pixels in W are involved in the averaging operation. The importance of the λ parameter is illustrated in Figure 5.5 that shows a noisy image and two images obtained after applying the FBF with two very different values of λ. Clearly, very different results are obtained, which implies the necessity of obtaining an optimal value of λ to maximize the quality of the filtered image.

3 Optimization algorithm description

The quality of a filtered image can be measured with several functions. In this case we will use the *Peak Signal to Noise Ratio* (PSNR) which, since the quality of the filtered image depends on λ, we define as a function of λ as follows:

$$PSNR(\lambda) = 20 \log \left(\frac{255}{\sqrt{\frac{1}{NMQ} \sum_{i \in \mathbf{F}} ||\mathbf{F}_\mathbf{i}^o - \mathbf{F}_\mathbf{i}^\lambda||^2}} \right) \tag{5.6}$$

where M, N are the image dimensions, $|| \cdot ||^2$ denotes the square of the Euclidean norm, $\mathbf{F}^o$ is the original noise-free image, and $\mathbf{F}^\lambda$ is the filtered image obtained after applying the FBF, respectively.

As mentioned above, we seek for the value of λ that achieve the optimum PSNR performance. To find it, we apply the following algorithm: First, we take an initial λ_0 for λ as a rough approximation to the optimum. For the current approximation to the optimum, λ_n, we approximate the derivative of $PSNR(\lambda_n)$ as

$$D(\lambda_n) = \frac{PSNR(\lambda_n + \delta) - PSNR(\lambda_n - \delta)}{2 \cdot \delta}, \tag{5.7}$$

where, $\delta > 0$. Now, we check whether the actual approximation to the optimum is already good enough: if $|D(\lambda_n)| < \epsilon > 0$, then the derivative at λ_n is small enough to conclude that it is very close to the optimum, and the method stops. Otherwise, we find the next approximation to the optimum λ_{n+1} from the previous λ_n as

$$\lambda_{n+1} = \lambda_n + \alpha D(\lambda_n), \tag{5.8}$$

where $\alpha > 0$ is called the learning parameter whose importance will be commented in Section 4, and the above procedure is repeated. Notice that, according to Eq. (5), the sign of $D(\lambda_n)$ indicates the direction in which the optimum is located and the *speed* in which the algorithm advances towards it is proportional to $|D(\lambda_n)|$.

4 Simulation Examples

To illustrate the use of the derivative in the identification of the optimum λ for the FBF, the corrupted image of Lenna in Figure 5.5(a) will be used. Figure 5.6 presents the PSNR computed for the image and the complete range of λ values. This figure shows that the optimum filtering performance is attained for $\lambda = 0.24$. To compute this PSNR curve, it was necessary to filter the Lenna image one hundred times - from $\lambda = 0.01$ to $\lambda = 1$ in steps of 0.01. This is a valid but computational overwhelming method to obtain the optimum parameter. Furthermore, we only obtain the optimum with a certain accuracy which depends on the step used. To improve the accuracy we need to reduce the step, which in turn means to increase the number of times the image needs to be filtered. For instance, to reduce the accuracy in one order, it is necessary to increase by 10 the number of times the complete image is filtered.

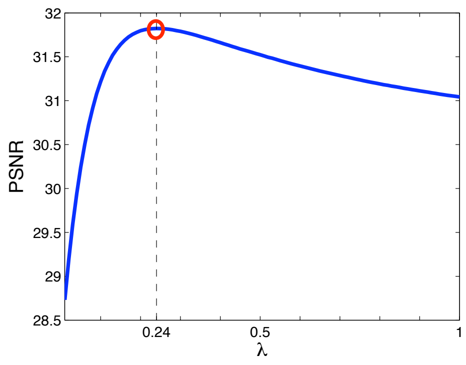
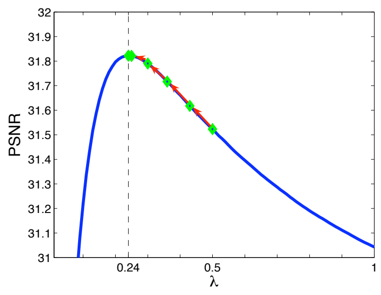


Figure 5.6: PSNR curve for the complete range of λ values in the corrupted image of Lenna in Figure 5.5(a).



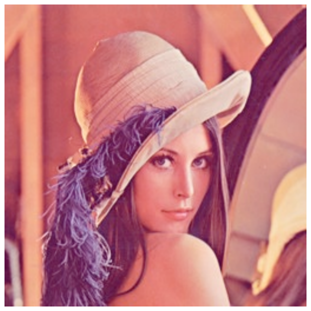


Figure 5.7: Optimization for initial point $\lambda = 0.5$ and $\delta = 0.01$, $\alpha = 0.05$ and $\epsilon = 0.1$: a) optimization steps, b) optimum filtered image.

An alternative to the extensive computation used to obtain Figure 5.6 is the derivative-based -or gradient-based- optimization treated in this paper. Let us illustrate the use of this method with an example. The optimization will be started from the central value $\lambda = 0.5$. The optimization parameters are $\delta = 0.01$, $\epsilon = 0.1$ and $\alpha = 0.05$. The evolution of the optimization is depicted in Figure 5.7(a). We arrive to the optimum filtering the image only 12 times - the number of approximations of the derivative (6) times the 2 applications of the filter per approximation (2 according to Eq. (5.7)). Furthermore, by reducing ϵ, the accuracy of the optimum can be improved with only a few more runs. The optimum filtered image attained is shown in Figure 5.7(b).

The convenient choice of the optimization parameters is of paramount importance to ensure convergence to the optimum. We have already commented that ϵ controls the accuracy achieved in the identification of the optimum. In this paper, we are considering the noise-free optimization case, so that we assume that the function to optimize -the PSNR curve- is free of measurement noise. If this is not the case, ϵ should be high enough so that it exceeds the noise level in the derivative estimation.

The three parameters, δ, α and ϵ, have interrelated effects. For the sake of easy understanding, the influence of the choice of δ and α will be discussed separately, varying the parameters one at a time while maintaining the other fixed. Let us run the optimization for δ values equally distanced in the logarithmic scale over [0.001, 1] and for fixed values of $\alpha = 0.05$ and $\epsilon = 0.1$. The results are presented in Table 5.1. The lower the value of δ, the better the derivative estimation. This is again assuming the PSNR function to optimize is free of measurement noise - it is well known that derivative estimations are highly affected by noise. Since the derivative is more accurately estimated for low δ values, the optimization arrives to higher values of

δ	# points	Attained PSNR
$1 \cdot 10^{-3}$	8	31.822
$3.26 \cdot 10^{-3}$	6	31.822
$1 \cdot 10^{-2}$	6	31.822
$3.26 \cdot 10^{-2}$	6	31.822
$1 \cdot 10^{-1}$	5	31.815
$3.26 \cdot 10^{-1}$	4	31.634
1	∞	Divergence

Table 5.1: Results for different values of δ and $\alpha = 0.05$ and $\epsilon = 0.1$ with starting point $\lambda = 0.5$.

α	# points	Attained PSNR
$5 \cdot 10^{-3}$	55	31.822
$1.58 \cdot 10^{-2}$	18	31.822
$5 \cdot 10^{-2}$	6	31.822
$1.58 \cdot 10^{-1}$	29	31.822
$5 \cdot 10^{-1}$	∞	Divergence
1.58	∞	Divergence
5	∞	Divergence

Table 5.2: Results for different values of α and $\delta = 0.01$ and $\epsilon = 0.1$ with starting point $\lambda = 0.5$.

PSNR. Also, we can see that the value of δ is related to the number of points or λ values in the optimization where the derivative is estimated. The higher the value of δ, the lower the number of points. Notice that if δ is too high, the optimization algorithm may not converge to the optimum. The changes observed in the optimization method due to the use of different values of δ can be easily explained in simple geometric terms. If the current value of λ is far from the optimum, high δ values result in derivative estimations of high magnitude. Therefore, the optimization is speeded up and the number of points evaluated are reduced. If otherwise λ is close to the optimum, the optimum may be included in the interval $[\lambda - \delta, \lambda + \delta]$ where the derivative is estimated. If this is the case, the magnitude of the derivative is underestimated. The higher the value of δ, the higher the interval where this occurs and the further to the optimum we may converge.

Let us run the optimization for α values equally distanced in the logarithmic scale over [0.005, 5] and for fixed values of $\delta = 0.01$ and $\epsilon = 0.1$. The results are presented in Table 5.2. We can see that the value of parameter α is especially important for the optimization. The optimization may easily diverge if the parameter is made too high. Nonetheless, for very low values of α, the number of points where the derivative has to be estimated is much increased. For instance, for $\alpha = 5 \cdot 10^{-3}$, 55 derivative estimations were needed to arrive to the optimum. This means that the Lenna image had to be filtered 110 times, 10 times more than for computing Figure 5.6. The best α value, so that the algorithm converge to the optimum in the lowest number of points, depends very much on the initial point - initial λ value. As a general rule, α should be set high enough for a fast exploration while not too high to avoid divergence.

5 Conclusions

The use of real applications of Mathematics motivates the learning process of students in several ways. In particular, most mathematical notions can be explained by use of different models. In this work we have presented a simulation model which will allow the students to relate the notion of derivative of a function and one of its applications which is optimization problems.

By the simulation process described in this paper, the students will approximate the optimal value of the parameter in a function and, moreover, they will really see how appropriate is the result of this algorithm since the visualization of the output images by using a filter technique will allow the students to search for better results.

Referencias

[1] http://en.wikipedia.org/wiki/Pinhole_camera_model

[2] R.C. González, R.E. Woods, Digital Image Processing, Prentice Hall, 2002.

[3] K.N. Plataniotis, A.N. Venetsanopoulos, Color Image processing and applications, Springer-Verlag, 2000.

[4] P. Perona, J. Malik, Scale-space and edge detection using anisotropic diffusion, IEEE Transactions on Pattern Analysis and Machine Intelligence 12 5 (1990) 629-639.

[5] C. Tomasi, R. Manduchi, Bilateral filter for gray and color images, Proc. IEEE International Conference Computer Vision, 1998, 839-846.

[6] M. Elad, On the origin of bilateral filter and ways to improve, IEEE Transactions on Image Processing 11 10 (2002) 1141-1151.

[7] R. Garnett, T. Huegerich, C. Chui, W. He, A universal noise removal algorithm with an impulse detector, IEEE Transactions on Image Processing 14 11 (2005) 1747-1754.

[8] S. Morillas, V. Gregori, A. Sapena, Fuzzy Bilateral Filtering for Color Images, ICIAR'06, Lecture Notes in Computer Science 4141 (2006) 138-145.

[9] A. George, P. Veeramani, On Some results in fuzzy metric spaces, Fuzzy Sets and Systems 64 3 (1994) 395-399.

[10] A. George, P. Veeramani, Some theorems in fuzzy metric spaces, J. Fuzzy Math. 3 (1995) 933-940.

[11] V. Gregori, S. Romaguera, Some properties of fuzzy metric spaces, Fuzzy Sets and Systems 115 3 (2000) 477-483.

[12] V. Gregori, S. Romaguera, Characterizing completable fuzzy metric spaces, Fuzzy Sets and Systems 144 3 (2004) 411-420.

[13] A. Sapena, A contribution to the study of fuzzy metric spaces, Appl. Gen. Topology 2 1(2001) 63-76.

[14] Wellstead PE, Zarrop MB. *Self Tuning Systems Control and Signal Processing.* England: John Wiley & Sons 1991

[15] Ogata K. *Modern Control Engineering (4th Ed.).* EEUU: Prentice Hall 2002

[16] J. Camacho, J. Picó, A. Ferrer, Self-tuning run to run optimization of fed-batch processes using unfold-pls, AIChE Journal 53 (7) (2007) 1789–1804.