

IMPLEMENTATION OF CAPTCHA PATTERN RECOGNITION USING DENOISING AND
OPTICAL CHARACTER RECOGNITION (OCR) METHODS

Presented by:
HAMILTON SMITH GÓMEZ OSORIO
JULIANA LALINDE VELÁSQUEZ
VERÓNICA MENDOZA IGUARÁN
PABLO ALBERTO OSORIO MARULANDA
MARIANA URIBE ORREGO

Subject
NUMERICAL ANALYSIS

Teacher
EDWAR SAMIR POSADA MURILLO

MATHEMATICAL ENGINEERING
DEPARTMENT OF MATHEMATICAL SCIENCES
SCHOOL OF SCIENCES
EAFIT UNIVERSITY
MEDELLÍN
2020

1 INTRODUCTION

The security of websites is of great importance today, as they must protect the information of their users. A CAPTCHA is a test that determines whether a user is a human or a robot. It consists of showing an image with distorted text that the user must recognize. If it is a robot it will not be able to recognize it and the test will detect it. These texts are increasingly difficult to recognize, since most of them come with noises which distort the image. Nowadays, there are multiple ways to reduce or eliminate noise in images, so in this research we discuss and compare different methods for noise reduction in order to understand them in depth and achieve image text recognition. The methods to compare are Median filter, Mean filter, Bilateral filter, Gaussian filter and Wavelet transform. On the other hand, we talk about Optical Character Recognition (OCR), which recognizes characters individually in an image and then passes it to a text, thus detecting the text it contains.

Our product will be an exhaustive analysis of image processing in which, by means of different methods, we will add certain noises to CAPTCHA in order to evaluate which of the studied methods best serves to reduce noise. The comparison is made by interpreting the correlation statistic. The maximization of the correlation coefficient is sought, which is determined by analyzing the set of pixels between the two images. Also, we evaluate which of these processed CAPTCHAs are recognizable by OCR Methods.

2 DENOISING FILTERS

2.1 MEDIAN AND AVERAGE FILTER

The median and average filter work in similar ways, so they are often compared. In this subsection we will explain how both of these filters work and compare their results to see which one works better.

2.1.1 Median filter

The median filter is a non-linear method used to reduce noise from images. It is particularly good at removing “salt and pepper” type noise and preserving the edges of the image intact. It works by moving through the image pixel by pixel and replacing each value with the median value of the neighboring pixels; the number of pixels considered in every step is called the window. Seeing as the windows need to be sorted in every iteration the method can have a high degree of computational complexity if it is not applied properly. Seeing as the window may contain more pixels than there exist before the first pixel the edges of the image are usually excluded from the method.

```

Median filter algorithm
Input: image, window size
Process:
For x from 0 to image length
    For y from 0 to image length
        index = 0
        For fx from 0 to window size
            For fy from 0 to window size
                window[index] = imagePixel(x+fx, y+fy)
                index = index + 1
        end For
    end For
    Sort window
    median = window [ (window size)^2 / 2 ]
    imagePixel(x, y) = median
end For
end For

```

Figure 1: Pseudo code for the median filter

2.1.2 Average filter

The average or mean filter is a non-linear method used to reduce noise from images. It is particularly good at removing the intensity variation between neighboring pixels, but has trouble preserving the edges of the image intact. It works by moving through the image pixel by pixel and replacing each value with the average value of the neighboring pixels; like the median filter, the number of pixels considered in every step is called the window. This method can be inaccurate if you have a single pixel with a very unrepresentative value in the image, seeing as it can affect the average value of all the pixels in its neighborhood.

```

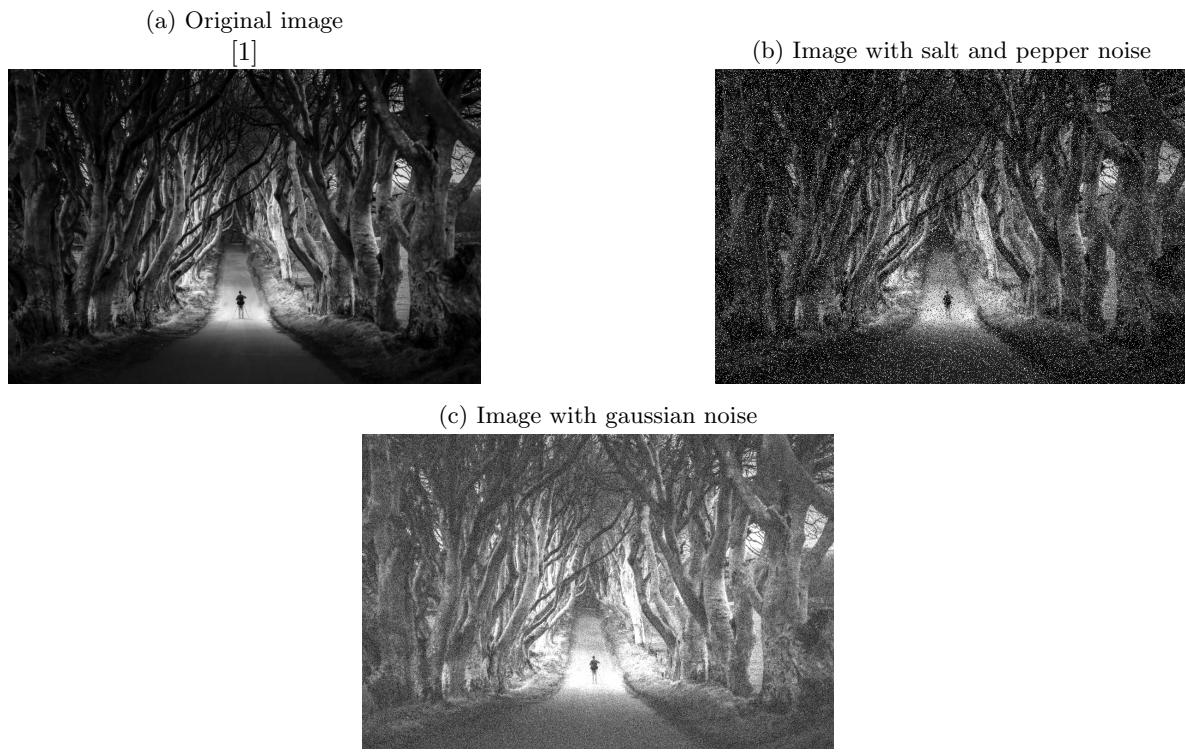
Average filter algorithm
Input: image, window size
Process:
For x from 0 to image length
    For y from 0 to image length
        index = 0
        For fx from 0 to window size
            For fy from 0 to window size
                window[index] = imagePixel(x+fx, y+fy)
                index = index + 1
            end For
        end For
        Sort window
        average = sum(window) / index
        imagePixel(x, y) = average
    end For
end For

```

Figure 2: Pseudo code for the average filter

2.1.3 Matlab implementation

To visualize the behaviour of the filters we tested them using Matlab functions. We first took a normal image and added two different noises: some salt and pepper noise, choosing a noise density of 0.15, and some gaussian noise, with a 0.25 parameter.



For the median filter we used the Matlab function $medfilt2(I, [m\ n])$, which applies the median filter to a 2D given image. In this function $[m\ n]$ represent the size of the window that will be used. Here is the image we got using the median filter with different window sizes for the salt and pepper noise:



Figure 4: Filtered image with 3x3 window

The best image we got using the median filter for the gaussian noise with the different window size is:



Figure 5: Filtered image with 5x5 window

For the average filter we used the Matlab function `conv2`, which applies a personalized convolution to a 2D image. In this case the filter we gave it to represent the average filter was `ones(windowsize)/(windowsize)2`. The best image we got using the mean filter with different window sizes for the salt and pepper noise is:



Figure 6: Filtered image with 7x7 window

The best image we got using the median filter for the gaussian noise with the different window size is:



Figure 7: Filtered image with 3x3 window

Then we proceeded to check the correlations between the original image and the filtered images. As seen in the table below, the median filter is more effective in removing the salt and pepper noise added to the images. We can see in the images that the median filter tends to take better care of the edges of the image and keep the shapes clear in general. When it comes to gaussian noise the mean filter seems to have a better correlation level with the original, but when you look at the images neither of the methos seems to be effectively removing this type of noise. For this reason we can conclude that the median filter has an overral better performance removind salt and pepper noise and neither of the filters seems to do a good job with the gaussian noise.

	3x3		5x5		7x7	
	Mean filter	Median filter	Mean filter	Median filter	Mean filter	Median filter
GAUSS	Filtered	Filtered	Filtered	Filtered	Filtered	Filtered
Norm 1	177.1094	177.2508	176.9865	177.2675	178.8301	180.6516
Euclidean norm	206.6165	207.1286	206.3327	206.6682	206.0730	206.1336
Uniform norm	260.9974	261.5478	261.7705	261.8174	262.3801	261.9081
Correlation	0.9618	0.9545	0.9568	0.9565	0.9449	0.9476

	3x3		5x5		7x7	
	Mean filter	Median filter	Mean filter	Median filter	Mean filter	Median filter
SALT	Filtered	Filtered	Filtered	Filtered	Filtered	Filtered
Norm 1	60.9255	26.2667	63.4635	36.2745	68.6627	48.1098
Euclidean norm	41.1387	3.1515	40.9546	5.3617	40.8882	7.4353
Uniform norm	93.4153	29.6902	89.1564	46.5490	87.6243	56.0471
Correlation	0.8720	0.9787	0.9216	0.9694	0.9283	0.9565

Figure 8: Comparison between the original and the filtered images

2.2 BILATERAL FILTER

Bilateral Filter is an image denoising method [2] that operates in the spatial domain, so it preserves features like edges but cannot preserve low-contrast detail like textures without introducing noise.

The bilateral filter is, then, a smoothing filter for images that replaces the intensity of each pixel with a weighted average of intensity values from nearby pixels. The weights, modeled by a gaussian

distribution, notably depend on differences in color intensity and depth distance, all of which helps preserve sharp edges in the image.

Mathematically the bilateral filter [3] [4] works as follows

$$I^{filtered}(x) = \frac{1}{W_p} \sum_{x_i \in \Omega} I(x_i) f_r(\|I(x_i) - I(x)\|) g_s(\|x_i - x\|)$$

and normalization term, W_p , is defined as

$$W_p = \sum_{x_i \in \Omega} f_r(\|I(x_i) - I(x)\|) g_s(\|x_i - x\|)$$

where

$I^{filtered}(x)$ is the filtered image;

I is the original input image to be filtered;

x are the coordinates of the current pixel to be filtered;

Ω is the window centered in x , so $x_i \in \Omega$ is another pixel;

f_r is the range kernel for smoothing differences in intensities

g_s is the spatial (or domain) kernel for smoothing differences in coordinates

The weight W_p is assigned using the spatial closeness (using the spatial kernel g_s) and the intensity difference (using the range kernel f_r). Consider a pixel located at (i, j) that needs to be denoised in image using its neighbouring pixels and one of its neighbouring pixels is located at (k, l) . Then, assuming the range and spatial kernels to be Gaussian kernels, the weight assigned for pixel (k, l) to denoise the pixel (i, j) is given [3] by

$$(i, j, k, l) = \exp \left(\frac{(i - k)^2 + (j - l)^2}{2\sigma_d^2} - \frac{\|I(i, j) - I(k, l)\|^2}{2\sigma_r^2} \right)$$

where σ_d and σ_r are smoothing parameters, and $I(i, j)$ and $I(k, l)$ are the intensity of pixels (i, j) and (k, l) respectively.

After calculating the weights, normalize them:

$$I_D(i, j) = \frac{\sum_{k,l} I(k, l) w(i, j, k, l)}{\sum_{k,l} w(i, j, k, l)}$$

where I_D is the denoised intensity of pixel (i, j) .

MATLAB offers the command [5] to work with this filter:

`J = imbilatfilt(I)`: It receives the image to which you want to apply the filter

`J = imbilatfilt(I,degreeOfSmoothing)` In addition to the image, it specifies the amount of smoothing.

`J = imbilatfilt(I,degreeOfSmoothing,spatialSigma)` also specifies the standard deviation of the spatial Gaussian smoothing kernel.

2.2.1 Pseudocode of the Bilateral Filter

This pseudocode allows us to briefly understand how the process of this filter can work in matlab because here we explain how to choose the degree of smoothing given a part of the image, but you can enter any value.

```

Input: Image C
Start
Read C
Select patch Z of C without pointed edges
Trim Z
Find Z variance in Euclidean distance
Assign image smoothing value A ( $A > Z$ )
Assign parameter A to the bilateral filter
Calculate normalization term G for each neighborhood of pixels Q
Calculate bilateral kernel K of each Q
Calculate noise-free intensity R of each pixel I
Assign value of the bilateral filter with G, K and R of each I of the image
Output: filtered image with value of the bilateral filter in each I
End

```

2.2.2 Example

We have the original image. We remove it a patch without sharp edges, which is reflected in figure 9 and has coordinates (170.35) with dimensions of $50 * 50$, then we calculate its variance which is 25.96975 and with this we generate a greater degree of smoothing,in this case twice as large as the variance and we can also extend the spatial extension with a sigma for the variance.So, we apply the bilateral filter to this image with a degree of smoothing equal to 51.9395 and a spatial sigma equal to 3 and obtain the filtered image.

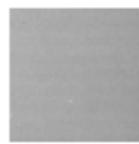


Figure 9: Chosen borderless patch



(a) Original image.

(b) Filtered image.

Figure 10: comparison between original image and filtered image

When calculating the correlation coefficient between the two images, we have a value of 0.9994 that tells us that when applying the filter the images almost completely resemble each other, preserving the details and shapes of the original image, but we can see that the bilateral filter improves the background of the image when noise is removed as it looks flatter and more compact.

To better visualize the behavior of this filter, another test is carried out with a different image where two of the best known noises are applied to it, the Gaussian noise with a parameter of 0.25 and the salt and pepper noise with a noise density of 0.15 . The image to use is [1] that was previously shown. After applying the noise, this image is processed with the bilateral filter, varying the degree of smoothing applied to it and thus knowing the best results for each of the two cases and understanding the behavior of the filter a little more.

Four different norms and an indicator that is the correlation are used to determine the best result.

Best filtering case with Gaussian filter				
GAUSS	Filtered 0.5	Filtered 1	Filtered 5	Filtered 10
Degree of smoothing	0.5	1	5	10
Norm 1	176.2134	176.2282	176.2379	176.2389
Euclidean norm	206.9820	206.9831	206.9846	206.9848
Uniform norm	261.7369	263.8110	261.7119	261.7105
Correlation	0.9686	0.9683	0.9680	0.9679

Best filtering case with Salt and Pepper filter				
SALT & PEPPER	Filtered 2.4	Filtered 5	Filtered 10,0	Filtered 1.5
Degree of smoothing	2.4	5	10,0	1.5
Norm 1	54.3551	55.3565	55.9089	56.1949
Euclidean norm	37.7744	39.4107	40.2982	40.7597
Uniform norm	84.0051	86.7675	88.2895	89.0859
Correlation	0.897	0.8974	0.8971	0.8968

Figure 11: best filtering cases for each type of noise applied



(a) Best filtered response for Gaussian noise with 0.5 degree of smoothing



(b) Best filtered response for Salt and Pepper noise with 5 degree of smoothing

Figure 12: Best image responses for each noise used

When carrying out different tests with these images and others, it can be seen that higher degrees of smoothing are required for salt and paper noise than for Gaussian noise. It can also be seen that this filter presents better results when working with images with well-defined edges, since it allows differentiating the elements of the image, thus giving a better result, since it is already known that this is a filter that preserves characteristics such as edges but cannot preserve low contrast details very well. Other conclusions based on the images analyzed, is that salt and paper noise offers better results when working with images with few elements, not like the image shown in [1] which has many details. Something interesting that is also observed in the cases studied, is that when an image is put on salt and paper noise, although the degree of smoothing is greatly varied, very compact and similar results are obtained, on the other hand, in Gaussian noise when very small smoothing values are assigned, there are very large variations, but between larger values taken as a parameter, more compact and similar results are achieved. It is also important to know that the bilateral filter is good when a image has own noise, not specifically a type of noise like gaussian and can have a small impact, as a background a bit distorted, as in the example of the photographer's image, which is the figure 10.

2.3 GAUSSIAN FILTER

The Gaussian filter is a linear filter that is used in image processing to smooth the details of an image through a convolution process. This process is responsible for transforming two functions into a third one, which will largely represent the magnitude of superposition of the first one on a translated version of the second one, thus eliminating different types of noise that may be in the original image. This Gaussian filter processes images in two different ways with respect to the domain in which you work. For this, it uses the Gaussian function.

$$h(x, y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}$$

as a base and the convolution process is established as $g(x, y) = h(x, y) * f(x, y)$, with a $f(x, y)$ function according to the type of filtering.

2.3.1 Filtering in the frequency domain

In this process the convolution is defined as $G(u, v) = H(u, v) * F(u, v)$ [6] where $F(u, v)$ represents the Fourier transformation of the input image and $H(u, v)$ is the frequency attenuating Gaussian filter. The process that is followed is described as

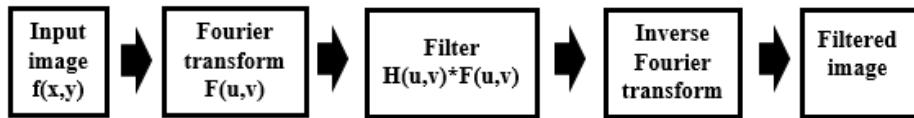


Figure 13: Filtering process with frequency domain

2.3.2 Filtering in the spatial domain

This type of filtering is applied to each of the pixels of the image that is subdivided into $n \times n$ sub-matrices, which are known as masks or convolution kernel. For this process, a nucleus $f(x,y)$ is established as the center of the mask and the corresponding convolution with the Gaussian function is performed for each pixel with respect to the nucleus, that is, we look for the approximation in the Gaussian function of the difference of intensity in the image of each pixel and the nucleus. As a result, a sub-matrix is obtained where the nucleus will have a higher intensity value and as the pixels move away from this nucleus they will have a lower value as the image shows.

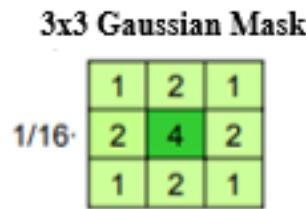


Figure 14: Filtering process with spatial domain[7]

2.3.3 Pseudocode

To implement this filter, we use the Matlab function *imgaussfilt(A, sigma)* which works as follows

```

Input: image nxm matrix A
        Standard deviation: sigma
Output: smoothed matrix A*

if (A is dispersed) then
    └── break;
if sigma=  $\emptyset$  then
    └── sigma=0.5      #default

apply heuristic method to define domain
if (frequency domain) then
    └── A* = frequency convolution to A
    └── A*
if (frequency domain) then
    └── I  $\leftarrow$  set of sub-matrices pxq
    └── While (not every element of I has been analyzed) do
        └── for (i from 1 to p) do
            └── for (j from 1 to q) do
                └── I*  $\leftarrow$  element of I
                └── I' = spatial convolution to I*
                └── A*  $\leftarrow$  add I' and reorganize image
    └── A*

```

Figure 15: Gaussian filter pseudocode

2.3.4 Example

To visualize the behavior of the Gaussian filter we have used Matlab Online with a default web image [8]. Here we can see the behavior for different sigma values



(a) Original image.

(b) Filtered image with $\sigma = 0.5$

(a) Filtered image with $\sigma = 1$ (b) Filtered image with $\sigma = 2$

Figure 17: comparison between original image and filtered images

We can see that as the sigma value is greater, the distortion of the image increases and it becomes more blurred. This occurs since the approximation that the Gaussian filter makes for a very large sigma value makes the data much more dispersed and far from its center.

2.3.5 Correlation

We also make the correlation between the original image and the filtered images with the command `corr2(A, B)` of MATLAB, obtaining the following results.

Correlation between real and filtered image for different sigma	
Sigma	Correlation
0.5	99.73%
1	98.01%
2	95.49%

Figure 18: correlation with different σ

It can be seen that, despite the distortion in the image, the correlation is still very high, since it is a method that works uniformly, however, it is notable that the greater the distortion, the lower the correlation.

2.3.6 Matlab implementation

As we have seen in previous filters, we wanted to perform a statistical analysis of the behavior of the filters used in the elimination of different noises and to compare the best alternatives to use the recovery process to be better. For this case we have used the same image presented above, this to have a more homogeneous behavior, with a standard deviation noise of 0.25 and a noise percentage of 15% for the Gaussian filter and the Salt and Pepper filter, respectively. The results found are shown in figure 19.

We can see in this table that, for the Gaussian noise, when comparing the correlations between the original and the resulting image, after the filtering process, a better behavior was obtained when the standard deviation used is 1.0 in the image recovery. We wanted to make a comparison between the results by applying a larger deviation and we can see that, although some magnitudes of the norms are less than those found for the aforementioned case, the correlation with the original image is less. This is because when working with a parameter of 1.5, the method begins to be a little unstable by distorting the details of the image in a greater proportion, behavior that is evident since the greater the

Best filtering case with Gaussian filter				
GAUSS	Filtered	Filtered	Filtered	Filtered
standard deviation	0,5	0,7	1,0	1,5
Norm 1	176,3401	175,8170	175,6803	176,7579
Euclidean norm	206,9914	206,9850	206,5414	206,9923
Uniform norm	263,8110	262,4580	261,9957	261,9011
Correlation	0,9316	0,9615	0,9653	0,9647

Best filtering case with Salt and Pepper filter				
SALT & PEPPER	Filtered	Filtered	Filtered	Filtered
standard deviation	0,5	0,7	1,0	1,5
Norm 1	66,5954	59,8643	56,4872	59,8291
Euclidean norm	41,6068	41,3498	41,2332	41,1813
Uniform norm	95,2990	92,4634	89,906	88,5867
Correlation	0,7090	0,8357	0,8964	0,929

Figure 19: best filtering cases for each type of noise applied

deviation, the greater the area used by the method to assign the values to the sub-matrices established in the partitioning process and the greater the variation of the averages for each value assigned in the process to each pixel resulting from the filtered image.

For the Salt and Pepper noise we can see that, as in Gaussian noise, better results are obtained when the value of the input parameter that the filter has is increased. Furthermore, in the table presented it can be seen that the correlation increases significantly when using a deviation of 1.5 instead of 1.0, however, when analyzing the improvements that this implies in terms of reducing the magnitudes of the selected standards these are very little significant in comparison to the results obtained when increasing the parameter in the previous cases. On the other hand, when viewing the recovered images for the two best cases presented, these show that the image distortion for the case with the highest value of the deviation parameter is moved a little further from the original image, looking more blurred, since that like the previous case we are over adjusting the filtering. Therefore, and seeking a greater relationship between the image filtering processes for the present filter, we will seek to work in a standard way with a deviation of 1.0 in the filtering process and, in addition, the results will be evaluated by increasing the parameter mentioned in the Salt and Pepper noise when it is presented in other types of images.



Figure 20: Original image



(a) Best filtered response for Gaussian noise with 1.0 standard deviation



(b) Best filtered response for Salt and Pepper noise with 1.0 standard deviation

Figure 21: Best image responses for each noise used

2.4 WAVELET TRANSFORM METHOD

The Wavelet Transform method is explained below as an option for noise reduction in images. Initially, the image containing all the wavelet transform coefficients is taken, that is, it contains all the information, the next step is to discard those coefficients that do not matter, that is, the parts of the image that contain the noise, to finally with the resulting coefficients obtain the image without noise.

The wavelet transform is a special type of mathematical transform that represents a wave in terms of translated and dilated versions of a finite wave. That is to say, Wavelets are generated from the translation and change of scale of the same wavelet wave called "Mother" as seen in Figure 22, where the blue wave is moved to the right and its scale has increased becoming the red wave.[9]

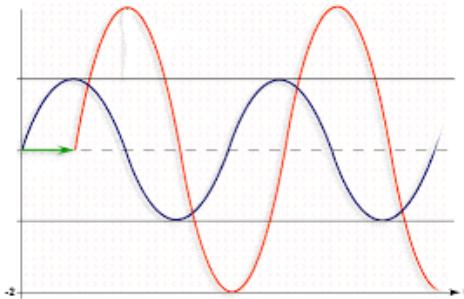


Figure 22: Finite wave, and translated wave and with scale change

For an image, the wavelet transform is a transformation of said image in which it is divided into two types of smaller images, the trend (direction or direction of the wave) and fluctuations (variation in intensity, measurement or quality). The trend becomes a lower resolution copy of the original image and the fluctuations store information regarding local changes in the original image. The most significant trend and fluctuations allow image compression in exchange for discarding irrelevant information and eliminating noise. [10]

2.4.1 Developing

Noise reduction based on the wavelet transform involves three basic steps, which are:

1. Decomposition into wavelet sub bands of the image with noise.
2. Assessment of the wavelet coefficients obtained.
3. Reconstruction of the image from the resulting coefficients using the inverse wavelet transform.

2.4.2 Decomposition into wavelet sub bands of the image with noise

To apply the wavelet transform to images you need to use the two-dimensional (2D) wavelet transform, that is, a dyadic decomposition (2) as shown in Figure 23. The original data matrix B_0 (matrix containing the intensity of the pixels of the image) decomposes into 4 sub-bands B_1 , H_1 , V_1 , and D_1 . Set B_1 contains a smoothed image from the original image while H_1 , V_1 , and D_1 contain the details (reflecting abrupt changes in intensity from one image pixel to an adjacent pixel). Each sub-band is one-quarter of the size of the original image. Then the approximation band B_1 can be transformed to obtain B_2 and additional details H_2 , V_2 and D_2 where the subscripts reflect the level of the transformation. In turn, it is possible to transform B_2 and so on.[11]

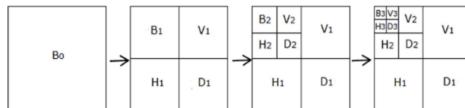


Figure 23: The original image B_0 and the first level, second level and third level of the two-dimensional wavelet dyadic (2D) transformation.

This decomposition is presented in Figure 24 with a 256x256 resolution gray-scale image. The image in the upper left corner is the original smoothed image (that is, at a lower resolution of 128x128); it contains the approximation coefficients resulting from applying the wavelet transform. The other three contain the details of the image in vertical, diagonal and horizontal directions. [12]

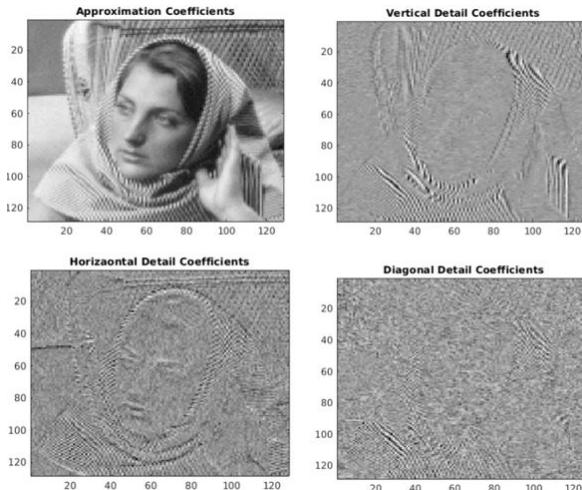


Figure 24: Decomposition of the image by wavelet transform.

2.4.3 Assessment of the wavelet coefficients obtained

In this step the relevant coefficients or those with the most important details of the image are determined. This is done through a threshold, a value that allows determining the significance of the coefficients. The coefficient is compared with said threshold, if the magnitude of the coefficient is less than the threshold, this becomes zero, it is declared insignificant (it is eliminated) and if its magnitude is greater than the threshold, it is considered significant (it is not eliminated).

Experimentally it has been shown that the neighboring coefficients of a significant wavelet coefficient are also significant. This is the NeighShrink method that takes into account the magnitude of the coefficient and that of its immediate neighbors to determine if it is significant or not. The experimental results showed that NeighShrink offers better results than the other classic wavelet methods. To make

the decision to zero or preserve the coefficient, the magnitude of the coefficient and that of its immediate neighbors are used.

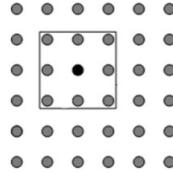


Figure 25: Wavelet coefficient and its neighbors in the same band.

In this way, the threshold distinguishes insignificant coefficients possibly caused by noise from those significant coefficients that are generated by important image structures.

Let $\mathbf{f} = \{f_{ij}\}$ be the matrix that contains the pixels of the original image and $\mathbf{g} = \{g_{ij}\}$ the matrix of the noisy image, we have:

$$g_{ij} = f_{ij} + e_{ij}; i = 1, \dots, N; j = 1, \dots, M$$

N: number of rows and M: number of columns

Where $\mathbf{e} = \{e_{ij}\}$ represents a white noise matrix with normal distribution $N(0, \sigma_n^2)$. The noise elimination can be formulated as the problem of finding an estimate $\hat{\mathbf{f}} = \{\hat{f}_{ij}\}$ of $\mathbf{f} = \{f_{ij}\}$ from the noise image $\mathbf{g} = \{g_{ij}\}$ that minimizes the error.

Color images can be represented using various arrays (one for each channel, for example three for RGB color images), so without losing generality, you can limit the analysis to gray-scale images.

In the wavelet domain, we get:

$$y_{ij}^{(k)} = x_{ij}^{(k)} + e_{ij}^{(k)}$$

where $y_{ij}^{(k)}$ represents the wavelet coefficients of the complete matrix at the level of decomposition k , $x_{ij}^{(k)}$ the wavelet coefficients of the image without noise and $e_{ij}^{(k)}$ the noise. Thus, in this domain, the problem is transformed into finding an estimate $\hat{\mathbf{x}} = \{\hat{x}_{ij}\}$ of $\mathbf{x} = \{x_{ij}\}$ from $y_{ij}^{(k)}$ that minimizes the error.

2.4.4 NeighShrink method

In the NeighShrink method for each wavelet coefficient $\mathbf{y} = \{y_{ij}^{(k)}\}$ we need to consider a neighborhood V_{ij} of it. This neighborhood is taken so that it contains the same number of pixels above, below, to the left and to the right of the pixel to consider. This means that the possible sizes of the neighborhood are 3x3, 5x5, 7x7, etc. Figure 25 shows a neighborhood of 3x3 centered on the wavelet coefficient to consider that it is the most used window size and that, as a rule, offers better results. Be

$$S_{ij}^{(k)} = \sqrt{\sum_{(l,m) \in V_{ij}} \{y_{lm}^{(k)}\}^2}$$

the indicator that tells to what extent the area of the image corresponding to the coefficient presents sudden changes in intensity. The estimate $\hat{\mathbf{x}} = \{\hat{x}_{ij}\}$ is obtained from the following expression:

$$\hat{x}_{ij}^{(k)} = \left(1 - \left\{ \frac{\lambda}{S_{ij}^{(k)}} \right\}^2 \right)_+ y_{ij}^{(k)}$$

Where $(\cdot)_+ = \max(\cdot, 0)$ and $\lambda = \sigma_n \sqrt{2 \log(N^2 M^2)}$ is the threshold named above. The bound requires prior knowledge of the noise variance σ_n^2 . If there is no knowledge of this variance, it can be

calculated from the wavelet coefficients $y_{ij}^{(1)}$ of band D1, using the estimator:

$$\hat{\sigma}_n = \frac{\text{Median}(|y_{ij}^{(1)}|)}{0.6745}, y_{ij}^{(1)} \in D_1$$

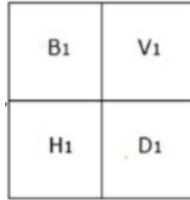


Figure 26: Image of the first level of image decomposition.

With this process the relevant coefficients or those with the most important details of the image are obtained and noise is eliminated.

2.4.5 Reconstruction of the image.

Finally, the estimate of the image is obtained through the inverse wavelet transform of the coefficients $\hat{x} = \{\hat{x}_{ij}^{(k)}\}$. The Inverse Transform takes the found approximation coefficients as input and inverts the decomposition step.

2.4.6 Matlab implementation

Implementation was done in Matlab with the `wdenoise2(IM)` function which does exactly what was described above.

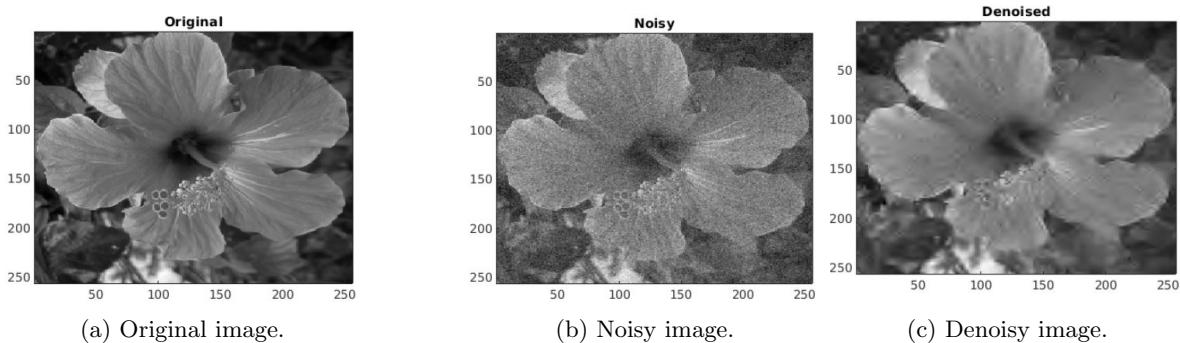


Figure 27: Comparison original image, noisy and denoisy image.

The figure a) shows a gray scale image, b) shows the same image with noisy and c) shows the denoised image. This example was obtained from Matlab.

An analysis is now performed to check the importance of the level of decomposition for this method. An implementation with decomposition levels to the same image of 1,2,3 and 4 was carried out for both the image with Gaussian noise and the Salt and Paper noise. For both types of noise, it is observed that the higher the level of decomposition, the better the reconstruction of the image. This is shown in the figure 25 where the correlation between the pixel arrays between the actual image and the filtered image increases. It is also perceived that there is a limit for the level of decomposition because when it is 4, with the Gaussian noise the correlation does not improve and the norms become higher, which means that the image is distorted because an overfit is being generated, that is, the noise is not so great and this level of decomposition is not necessary because the adjustment is excessive. Analogously for

Best filtering case with Wavelet filter				
GAUSS	Filtered	Filtered	Filtered	Filtered
Decomposition level	1	2	3	4
Norm 1	176,5935	177,0249	176,8059	176,9358
Euclidean norm	206,9869	206,9813	206,9841	206,9915
Uniform norm	263,6215	260,8785	261,5842	261,8530
Correlation	0,9451	0,9608	0,9616	0,9616

Best filtering case with Wavelet filter				
SALT & PEPPER	Filtered	Filtered	Filtered	Filtered
Decomposition level	1	2	3	4
Norm 1	986,5480	952,0690	950,1330	943,4270
Euclidean norm	430,1100	428,8360	428,6200	428,6320
Uniform norm	143,3881	139,2055	138,8174	137,1003
Correlation	0,6471	0,6761	0,6784	0,6763

Figure 28: Image of the first level of image decomposition.

the Salt and Paper noise, it happens that with the decomposition level of 4 the correlation is decreased, which means that the image is no longer improving.

The best filters are presented in the figure 26 (a) and (b) corresponding to decomposition level 3 for both types of noise already mentioned.



(a) Gaussian noise image filtered with Wavelet



(b) Salt and Paper noise image filtered with Wavelet

Figure 29: Best image responses for each noise used.

Finally, the Wavelet transform method decomposes an image into components of different scales, position and orientation, facilitating the statistical analysis of the image properties, however it is observed that for the elimination of noise it is not as effective a method, as seen in figure 26 (b) where the image still has too much disturbance. It is important to note that this method is mainly used for noise elimination but for frequencies that change over time such as lightning electric field signals, that is, this method is good for when the object under study depends on time.

2.5 Method comparison

Each of the previously seen methods has different characteristics and provide different results, so it is necessary to make a comparison of these to establish which is more appropriate for our research purpose. This comparison was made through the correlation between images, which is based on the maximization of the correlation coefficient determined by analyzing the set of pixels between the two images (original and with noise).

The process consists in choosing different noise intensities for each of the chosen types, which were Gaussian noise and Salt and Pepper. For the first type of noise, intensities of 0.1, 0.2, 0.4 and 0.6 were

established, which are equivalent to the variation of the Gaussian noise; This was done for each method obtaining the results presented in the table 24. With the type of noise Salt and Pepper, intensities of 0.05, 0.1, 0.2 and 0.3 were established, equivalent to 5%, 10%, 20% and 30% of noise intensity, respectively. With these results show the images recovered from the noise with the highest correlation values for each case.

The process was done with the figure 22.



Figure 30: Original image

Gaussian noise				
Intensity	0,1	0,2	0,4	0,6
Correlation	0,8761	0,8688	0,8377	0,7442
Salt and Pepper noise				
Intensity	0,05	0,1	0,2	0,3
Correlation	0,7807	0,6461	0,4769	0,3677

Figure 31: image correlation with noise

Image recovery for different noise intensity										
Gaussian noise										
Intensity	0,1					0,2				
Denoise filter	Mean	Median	Gaussian	Bilateral	Wavelet	Mean	Median	Gaussian	Bilateral	Wavelet
Correlation	0,9666	0,9584	0,9715	0,9634	0,9648	0,9640	0,9576	0,9696	0,9592	0,9632
Intensity	0,4					0,6				
Denoise filter	Mean	Median	Gaussian	Bilateral	Wavelet	Mean	Median	Gaussian	Bilateral	Wavelet
Correlation	0,9468	0,9442	0,9551	0,9554	0,9489	0,8885	0,8887	0,9045	0,9039	0,8977
Salt and Pepper noise										
Intensity	0,05					0,1				
Denoise filter	Mean	Median	Gaussian	Bilateral	Wavelet	Mean	Median	Gaussian	Bilateral	Wavelet
Correlation	0,9465	0,9849	0,9575	0,9555	0,7919	0,9381	0,9829	0,9282	0,9142	0,6962
Intensity	0,2					0,3				
Denoise filter	Mean	Median	Gaussian	Bilateral	Wavelet	Mean	Median	Gaussian	Bilateral	Wavelet
Correlation	0,9167	0,9692	0,8619	0,8389	0,6946	0,8872	0,9128	0,7842	0,7826	0,826

Figure 32: Image recovery for different noise intensity



(a) Filtered from gaussian noise with 0.1 intensity



(b) Filtered from gaussian noise with 0.6 intensity



(c) Filtered from salt and pepper noise with 0.05 intensity



(d) Filtered from salt and pepper noise with 0.3 intensity

Figure 33: Best filtered image for the given noise

When doing this analysis, we realize that the correlations obtained are very high, but when observing the visual results of the images, these values, in some cases, seem to be unreliable, since the difference with the original image is remarkable. For this reason, we decided to do a statistical analysis of the images by operating them as double matrices and calculating different statistics for a fixed Gaussian noise intensity of 0.25 and a salt and pepper noise intensity of 15%.

Numerical comparison of images - Gaussian noise						
Type of filter	0,25	Mean filter (3x3)	Median filter (5x5)	Gaussian filter	Bilateral filter	Wavelet transform
Image status	Noised	Filtered	Filtered	Filtered	Filtered	Filtered
Norm 1	177,8769	177,1094	177,2675	175,6803	175,9106	177,1412
Euclidean norm	207,0045	206,6165	206,6682	206,5414	207,0005	207,0148
Uniform norm	265,7632	260,9974	261,8174	261,9957	261,8990	262,0394
Correlation	0,8646	0,9618	0,9565	0,9653	0,9656	0,9616

Numerical comparison of images - Salt and Pepper noise						
Type of filter	0,15	Mean filter (7x7)	Median filter (3x3)	Gaussian filter	Bilateral filter	Wavelet transform
Image status	Noised	Filtered	Filtered	Filtered	Filtered	Filtered
Norm 1	74,4157	68,6627	26,2667	56,4872	56,0204	94,4156
Euclidean norm	42,0269	40,8882	3,1515	41,2332	40,4779	42,9052
Uniform norm	99,0118	87,6243	29,6902	89,906	88,599	137,3251
Correlation	0,5501	0,9283	0,9787	0,8964	0,8969	0,6745

Figure 34: statistical image comparison



(a) best result for Gaussian noise



(b) best result for Salt and Pepper noise

Figure 35: comparison between original image and filtered images

3 Optical Character Recognition (OCR)

“Optical character recognition (OCR) is a process of converting a printed document or scanned page into ASCII characters that a computer can recognize.” [13] For the entire develop of this project it is necessary to improve a process where all the characters get recognized once the denoising filter already applied, some techniques Will be used for this porpoise, one of them is PATTERN MATCHING. This method for OCR (optical character recognition) is able to keep a good processing time while maintaining efficiency and versatility. This algorithm has main functional modules which are:

- Image acquisition module
- Pre-processing module
- Feature extraction module
- Pattern generation

Image acquisition module is about obtain a text image from a file. This image was probably scanned(for this algorithm porpoise), so the data may carry some unwanted noise. So, a pre-processor is used to smooth the digitized characters [13]. In the next figure we can find in detail the process of the algorithm.

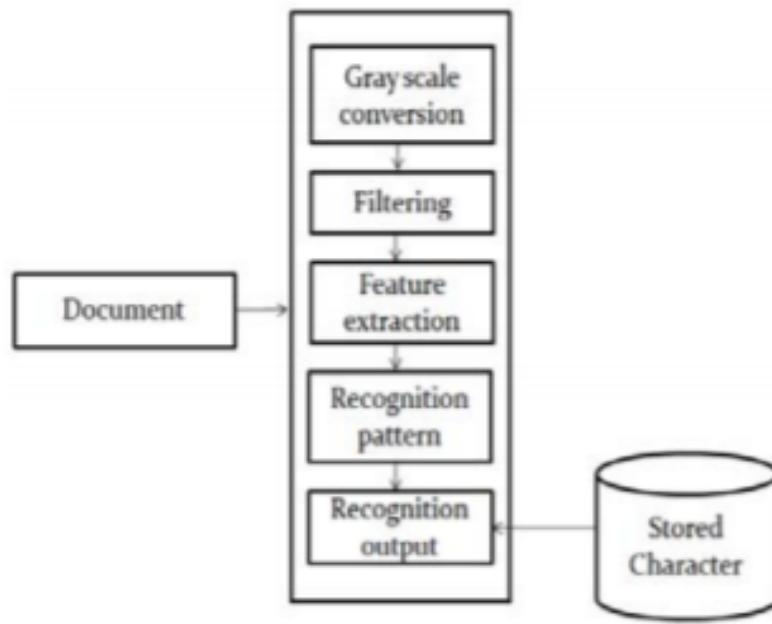


Figure 36: System block diagram

3.1 GrayScale

The first step is to make the image an uniform image in order to analyze every pixel there. To achieve this uniformity in the image it is necessary that the intensity of each pixel is given by general consensus, where a color is converted to an RGB base and from there to a gray scale, which, in this case, is described as follows

$$Y = 0.2126R + 0.7152G + 0.0722B$$

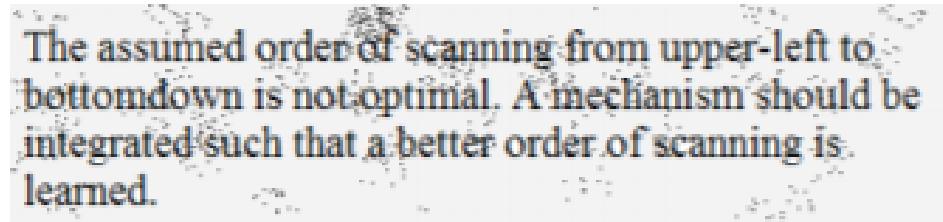


Figure 37: Image without grayscale filter

The assumed order of scanning from upper-left to bottomdown is not optimal. A mechanism should be integrated such that a better order of scanning is learned.

Figure 38: Image with grayscale filter

3.2 Feature Extraction

This is the process of getting information of a specific object in a group of them in order to facilitate classification and test of everyone. The input image might contain some lines of text that needs to be classified into single character for recognition. For this porpoise the following steps are to be applied:

- We analyze the total matrix of the image, scanning from the initial darker pixel, which is about to be named as top of the row.
- Now for the bottom the next blank line is detected. The area between those two items is considered as a row of characters in image.

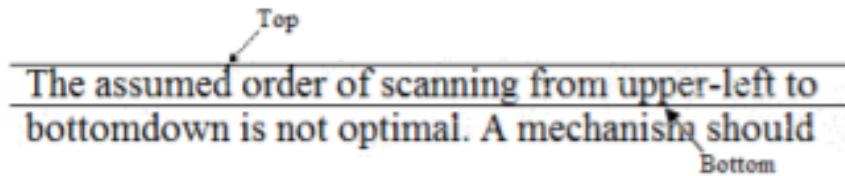


Figure 39: Detecting top and bottom row

- Then, we need to analyze every character in the row. So now this is done scanning the row vertically from top to bottom, the first darker pixel detected is the leftmost pixel in the character of that row.
- If the pixel founded is blank then is the right of the character.

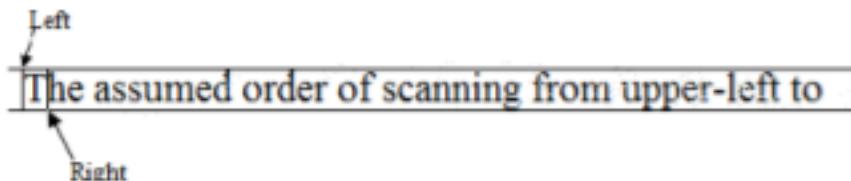


Figure 40: Detecting letter by letter

- Now the image is normalized cutting the edges where the character is white, that's means, cutting the character to the real size of it.



Figure 41: New edge of every letter

- Now we binarize the image we already normalize, where black representing 1 and white representing 0.



Figure 42: Binarization of the image

3.3 Recognition pattern

The goal in this main module is to compare the image of each trimmed character with that of training data. Due to the fact that each image of characters is made up with numerical pixels, we just need to compare pixel by pixel to find the level of similarity. So that, we calculate the correlation two-dimensional between the unknown data with the data we already have. A higher value means a better match. The formula is:

$$r = \frac{\sum_{m=1}^M \sum_{n=1}^N (A_{mn} - \bar{A})(B_{mn} - \bar{B})}{\sqrt{\left(\sum_{m=1}^M \sum_{n=1}^N (A_{mn} - \bar{A})^2\right) \left(\sum_{m=1}^M \sum_{n=1}^N (B_{mn} - \bar{B})^2\right)}}$$

[14]

3.4 Algorithm assumptions

- The algorithm assumes the input image has a clean background, or it has a big contrast with the color of each letter.

- Every character is lined horizontally.
- All the characters have a similar font and they should be machine printed or similar
- The test data contains characters whose font size is less than 42x42 pixels. (this is the size of the training data).

For example



Figure 43: OCR method application in Image (a) and Output in Image (b)

3.5 OCR applied into CAPTCHA recognition

We already know that we have an approach method to recognise characters, but this method need to accomplish a series of ideal assumptions which probably, CAPTCHA images are not able to achieve at all. So, we need to improve this method in order to get a better character recognition. A good way to begin doing this is to increase our training data, considering when characters are rotated, are bigger or smaller, has additional lines, etc.

3.5.1 Rotation and re-shape

In order to increase our training data, one option is to add the characters in a rotated version. To do this, the main character data base will be taken, then we are going to implement the nearest interpolation method.

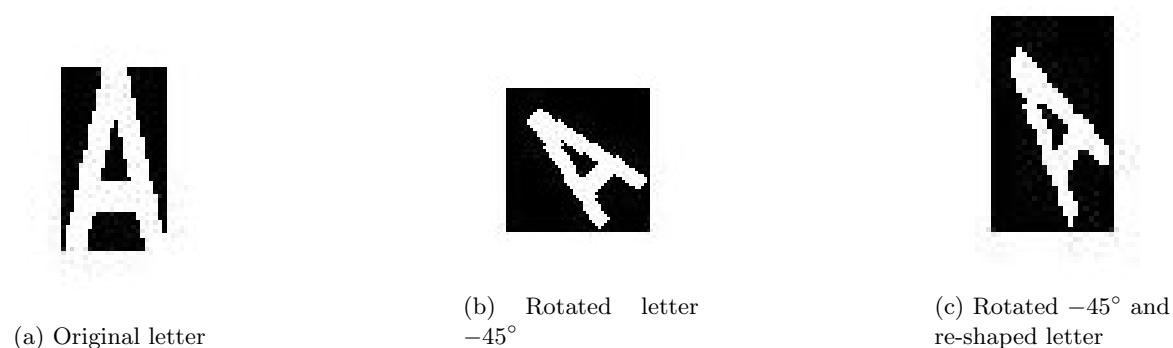


Figure 44: Rotated method process

As it was said before, the main characters to compare with the text need to be all of them in the same size, so it is needed to re-shape the rotated image. Main data base dimensions are 42x24. In the example case we obtain:(figure 32 part c)

In this way, the algorithm will be able to evaluate all this data base fount characters in all possible angles(stressing in angles between -45 and 45 degrees), making it more robust and less sensible to this kind of letter in a text.

3.6 Probability of each option

The algorithm use the Pearson correlation in 2 dimensions to compare the letters, so, this indicator give us a certain number we can evaluate in order to know how feasible and reliable the answer is. So, the algorithm gives a table with the correlation values of each letter. In CAPTCHA applications, this resource could give us a better approximation of how good the algorithm is to certain CAPTCHA formats. For example, given the next image:

JUDAS
PRIEST
775758
HOLA
DIEGO
12312
367945

(a) Image to see correlation index

JUDAS
PRIEST
775758
HOL1
DIEGO
12312
367945

(b) Output after OCR algorithm

Figure 45: OCR method application in Image (a) and Output in Image (b)

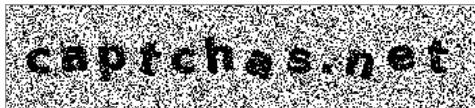
Caracter	Corr	Caracter	Corr
-----	-----	-----	-----
J	0.6137	O	0.68843
U	0.50781	L	0.70215
D	0.49547	1	0.44776
A	0.35101	D	0.52951
S	0.40383	I	0.74761
P	0.57727	E	0.60028
R	0.46249	G	0.36686
I	0.69058	O	0.60264
E	0.54606	1	0.52917
S	0.43232	2	0.38984
T	0.60193	3	0.46953
7	0.41553	1	0.52917
7	0.41553	2	0.55083
5	0.30548	3	0.48063
7	0.41553	6	0.41651
5	0.30548	7	0.41553
8	0.46939	9	0.42805
H	0.62836	4	0.33436
		5	0.42629

In this table we can see every of the correlation values for the example. It is important to remember that every correlation number here is the maximum value of the correlation analysis with all the character data base.

We can conclude that the algorithm is so good at analyzing the simple character, as "I" or "L", same as "T", "O", and "H". The "worst" one was the number "5". It supposed to be because this number is actually complex to draw.

4 Results

Now, with all the methods been explained, it is necesary to implement both of the methods (denoising filter and OCR) in order to see how good they work together to analize CAPTCHA images. The images that will be evaluated by all this methods are the next ones:



(a) CAPTCHA 1 to be evaluated



(c) CAPTCHA 3 to be evaluated



(e) CAPTCHA 5 to be evaluated



(b) CAPTCHA 2 to be evaluated



(d) CAPTCHA 4 to be evaluated



(f) CAPTCHA 6 to be evaluated



(a) CAPTCHA 7 to be evaluated



(b) CAPTCHA 8 to be evaluated



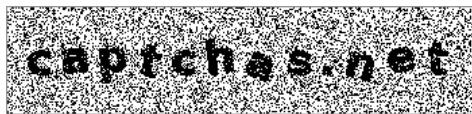
(c) CAPTCHA 9 to be evaluated



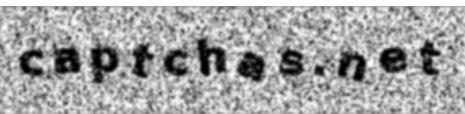
(d) CAPTCHA 10 to be evaluated

Figure 47: CAPTCHAS used in the OCR method

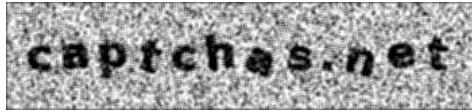
Now, it will be applied to everyone of the CAPTCHA images the filters mentioned before. So, the final analysis is going to be this way to all data:



(a) CAPTCHA 1 with Bilateral filter



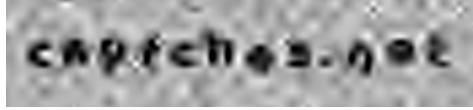
(b) CAPTCHA 1 with Gaussian filter



(c) CAPTCHA 1 with mean filter



(d) CAPTCHA 1 with median filter



(e) CAPTCHA 1 with wavelet filter

Figure 48: Denoising result for CAPTCHA 1

These are some examples of the algorithm output:

- Captcha 1- Gaussian filter: **C09PM54M906P**
- Captcha 3- default: **J0J430**
- Captcha 4- default: **3VO8OO**
- Captcha 5- Median Filter: **QG91Z5**
- Captcha 6- Mean filter: **I05CJE**
- Captcha 7- Bilateral: **2O0456**
- Captcha 8- default: **5C660V**
- Captcha 9- Wavelet filter: **6C97OC**

Finally, every of the results will be evaluated by a performance measure, where the success percentage is evaluated by the number of correct read characters from the algorithm between the total number of characters in the image. The next table is obtained:

	Default	Bfilt	Gfilt	MeaFilt	MedFilt	Wfilt
Captcha 1	0.073	0.073	0.16667	0.1428	0.1428	0.1538
Captcha 2	0	0	0	0	0	0
Captcha 3	0.6667	0.6667	0.6667	0.5714	0.5	0.6667
Captcha 4	0.83	0.83	0.66	0.4785	0.5	0.6667
Captcha 5	0.16	0.16	0.16	0.16	0.16	0.16
Captcha 6	0.5	0.5	0.5	0.5	0.25	0.37
Captcha 7	0.5	0.5	0.1667	0.25	0.25	0.5
Captcha 8	0.5	0.5	0.5	0.33	0.16	0.5
Captcha 9	0.6667	0.6667	0.5	0.2871	0.5	0.6667
Captcha 10	0	0	0	0	0	0

This results let us conclude some things. First of all, the CAPTCHA style like the one of CAPTCHA 3,4,6,7,8 and 9 are the best options to be evaluated with the algorithm. The CAPTCHAS mentioned let us evaluated which filter get the image worse to analyse it. So, the default algorithm of OCR is enough to this type of CAPTCHAs, just because a little filter is applied inside (filter of change into gray-scale).

This let us conclude that, for the cases of CAPTCHA 3,4,6,7,8 and 9 the mean and median filter are the ones that carry the image into a worse analysis. One of the biggest weakness of the OCR algorithm is that is not able to evaluate lower case letters. Evidently that weakness is well known because of the results. The small letter weakness is evident in case of CAPTCHA 5, where the algorithm is just able to evaluate correctly capital letters. Cases as CAPTCHA 1 and 10 are extreme cases of applications of the algorithm. It's seen that for case filters are actually the best way to evaluate the algorithm (apposite of what it's seen in the cases mentioned before). The noise in both of the images is higher than the other evaluation cases. Actually, cases of evaluation in CAPTCHA 10 as default and bilateral filter are not even recognised by the program, same for case of CAPTCHA 2.

5 Conclusions

- It is important to emphasize that the denoising methods have types of noise with which they works better, this could be evidenced in the comparison of methods where they had the same circumstances and the better performance of the Gaussian and median filters could be seen. This happened because gaussian noise and salt and pepper noise were applied . Bilateral noise also performed well with these noises, as it comes from Gaussian filter but can work better in other circumstances and is highly image dependent when it works with these default noises. On the other hand, with the wavelet transform method, the best results were not obtained with the characteristics in common and it is not very efficient for these noises, since it has a high computational load.
- The resulting algorithm by both of the methods applied here (Denoising filters and OCR) is a good combination knowing where every of them are good at. This means, for cases of "easy" CAPTCHA (the ones which do not have too much noise), the OCR algorithm is better if it is working by itself instead of working with denoising filter as, so they carry to a worst result.
- To make OCR and denoising filters work well together it is necessary to look for a CAPTCHA case where OCR inside filter is not enough and denoising filters carry a good result at filtering.
- OCR algorithm has two weakness. The fist one is that it does not evaluate lower case characters; the second one is that the algorithm is not so good at recognising characters in a different font. Additionally, not even the denoising filters are able to take off the lines that sometimes go across the characters. All of these are possible features to be improved in the future.
- The measurement to evaluate the last table is useful, but must be necessary to include another way to measure which include the correlation of every character, this is just because a "bad

measurement” in the table does not mean a mistake from the algorithm, it just involves a low correlation index.

References

- [1] Ritesh Saini. 15 amazing black and white landscape photos that will leave you in awe. In *Photzy*.
- [2] C. Knous and M. Zwicker. Dual-domain image denoising. In *2013 IEEE International Conference on Image Processing*, pages 440–444, Sep. 2013.
- [3] Bilateral filter. In *Wikiwand*.
- [4] A. Pardo. Analysis of non local image denoising methods., Ene. 2009.
- [5] Bilateral filtering of images with gaussian kernels. In *MathWorks Help Center*.
- [6] T. Acharya and A. K. Ray. Image processing., 2005.
- [7] G. García. Procesamiento audiovisual. In *Universidad de Murcia*.
- [8] 2-d haussian filtering of images. In *MathWorks Help Center*.
- [9] Universidad Politécnica de Valencia. Tratamiento de imágenes digitales mediante wavelets.
- [10] Václav Hlaváč. Wavelets transformation.
- [11] Introducción a la transformada wavelet.
- [12] Yanet Cesaire Velázquez Rafael Arturo Trujillo Codorníu and Antonio Cedeño Pozo. Image denoising using interscale and intrascale dependencies between wavelet coefficients. *Revista Cubana de Ciencias Informáticas*, 11(2):1–15, 2017.
- [13] Milan Shingote Pratik Ghanwat Faisal Mohammad, Jyoti Anarase. Optical character recognition implementation using pattern matching. In *ISBM School of Technology*.
- [14] MATPIC.COM. Reconocimiento de caracteres Ópticos (ocr) usando matlab.
- [15] Jian-Feng Ma Xiu-Li Bi Xuan Wanga, Bin Xiao. Scaling and rotation invariant analysis approach to object recognition based on radon and fourier–mellin transforms.
- [16] Pandasecurity. Panda security fed up with captcha? how to avoid it?
- [17] Universidad Eafit (<https://kappa.eafit.edu.co/>). Gestor de contraseñas.