DOI: https://doi.org/10.15276/aait.07.2024.18
UDC 004.932.4

Analysis of methods and algorithms for image filtering and quality enhancement

Viktor O. Speranskyy[1)]
ORCID: https://orcid.org/0000-0002-8042-1790; speranskyy@op.edu.ua; Scopus Author ID: 54401618900
Daniil S. Balaban[1)]
ORCID: https://orcid.org/0009-0004-2550-0586; danches.balaban@gmail.com
[1)] Odesa Polytechnic National University, 1, Shevchenko Ave. Odesa, 65044, Ukraine

ABSTRACT

This paper addresses the prevalent issue of image noise and presents methods for its mitigation. The paper describes, analyses and tests a variety of image filtering techniques, with specific reference to their use in different contexts. The filtering methods can be classified into two principal categories: linear filters, which include the Gaussian and mean filters, and non-linear filters, which comprise the median filter, the Fast Fourier Transform (FFT), the Non-Local Means (NLM) filter, and the anisotropic diffusion filter. The efficacy of each filter is mathematically described and evaluated on RGB images using the Python programming language. The study delineates the evaluation metrics and their respective advantages and disadvantages. The Root Mean Square Error (RMSE) and Peak Signal-to-Noise Ratio (PSNR) are employed as criteria for the analysis of algorithm efficiency. Furthermore, the mean execution time for each algorithm is also monitored. The experimental data suggests that linear filters are relatively fast but produce inferior results and are best employed as preparatory measures. Non-linear filters have been demonstrated to be more robust and applicable to a variety of noise types, although it has been established that they require parameter fine-tuning. The study demonstrates that anisotropic diffusion is suitable for both manual image processing and real-time applications, offering an optimal balance between processing speed and denoised image quality. NLM is optimal for high-quality single image processing due to its superior results, despite a slower processing speed. FFT is noted for its efficiency in eliminating periodic noise. Further research will be conducted on advancing filtering techniques for different real-world scenarios and autonomous systems.

Keywords: Image filtering; image quality enhancement; linear filters; non-linear filters; filtering metrics

For citation: **Speranskyy V., Balaban D. "Analysis of methods and algorithms for image filtering and quality enhancement". *Applied Aspects of Information Technology.* 2024; Vol.7 No.3: 255–268. DOI: https://doi.org/10.15276/aait.07.2024.18**

INTRODUCTION

Today, there are numerous approaches and methods available for image filtering. These methods can be categorized into traditional and deep-learning based methods. This paper focuses on traditional algorithms and doesn't cover usage and testing of neural networks. The reason behind this is that the filtering might and should be used in light-weight embedded systems, which computational power might be not enough for neural networks.

In general filtering algorithms are divided in two classes: linear and non-linear. Linear algorithms are simple, light-weight and always provide some results, while non-linear algorithms usually are more complex and robust, depend on parameters fine-tuning but also provide much better results. There are several well-known algorithms:

- Gaussian filter – a linear smoothing filter that reduces noise using Gaussian distribution. It means pixels closer to the center of the mask are affected more which helps in preserving edges. Popular filter with wide range of application due to its simplicity. Often is used as a part of more complex algorithms, like Sobel, Canny, Non-Local Means, etc., for cheap noise removal. The major downside is that it is not strong enough to provide meaningful results alone [1, 2], [6].
- Mean filter – a linear smoothing filter that is a downgrade version of Gaussian filter. The mean filter applies universal smoothing which makes it little faster and suitable for easy tasks. It can also be used in constructing image pyramids. The downside is that this filter is very sensitive to high and low intensities, such as salt-and-pepper noise, and can lose edges easily [1, 3].
- Median filter – a non-linear smoothing filter which brings Gaussian and mean filter to the next level. Instead of calculating the mean value, it calculates median value thus eliminating influence of "maxed out" pixels. A good example is "salt-and-pepper" noise, where noisy pixels have either lowest or maximal intensity [2, 5].

- Fast Fourier Transform (FFT) – is not a filtering algorithm on its own, but rather a technique that enables easy and efficient noise removal [4, 5]. FFT shifts image to frequency domain where noise is represented as high frequencies areas. Then any available approach can be used to filter out or completely remove that high frequency areas and image is transformed back into spatial domain using IFFT (Inverse Fast Fourier Transform) [25].
- Non-Local Means (NLM) – a non-linear version of Gaussian filter. Instead of computing mean values of the neighboring pixels, NLM scans the whole picture for the alike pixels. NLM shows great results, but is very computationally intensive [2, 4].
- Anisotropic diffusion – a non-linear filter based on differential equations. This filter leverages the gradients of the image to better detect edges and apply smoothing skipping that areas [3, 4].

For the algorithms testing several noise types were chosen as the most common. Gaussian noise that is usually found in digital images. It can be caused by optical sensors malfunction, high temperatures and/or poor illumination. This type of the noise has no special features and looks like evenly spread grains.

Film grain noise appears in pictures took using analog cameras after film is exposed to the silver halide. Though nowadays it is stylish it is still will be used for a testing purpose.

Salt and pepper noise introduces pixels with high and low intensities, or white and black respectively. This is purely digital errors result, such as faulty sensor, errors during writing, encoding or any other operation with bits.

Periodic noise main feature is its repeating nature. The most common cases are filming another display, low quality signal or electromagnetic interference. Usually, it is seen as repeated lines or waves and is hard to get rid of.

Anisotropic noise can be easily distinguished from others by its “direction”. This type of noise features strong differences in pixels intensity, “texture” of the object and is usually introduced by instability during the filming or sensor malfunction.

ANALYSIS OF EXISTENT ALGORITHMS

Gaussian filter can be used to blur out the image to reduce the noise. The underlying principle of current algorithm is a Gaussian function, usually in the shape of a bell, which is applied to every pixel. Its main goal is to smooth the area around the target pixel reducing the smooth strength closer to the edges of the mask [14, 16].

Gaussian filter is applied using following steps:

1) Define Gaussian function:

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

where σ represent the standard deviation of the Gaussian distribution. It effectively controls the amount and strength of the blurring.

2) Perform convolution – create a matrix of weighs using Gaussian function, then apply the matrix to each pixel.

3) Normalize – all matrix values are normalized to sum up to 1. Thus, image brightness is preserved.

4) Apply filter – Gaussian filter is applied to each pixel to smooth surrounding area, targeting pixels closer to the center more.

Applying Gaussian filter to a pixel can computed as:

$$I(i,j)=\frac{1}{\sum_{m=-k}^{k}\sum_{n=-k}^{k}g(m,n)}\sum_{m=-k}^{k}\sum_{n=-k}^{k}G(m,n)*I(i+m,j+n),$$

where $G(m,n)$ is Gaussian kernel at (m,n), k is Gaussian kernel size, usually equals to $\lfloor 3\sigma \rfloor$.

Gaussian filter is generally used as a preparation for following image processing. It is a good choice when the level of noise is low to average and it doesn’t take much effort to preserve the edges. Though, Gaussian function is radially symmetric, which means smoothing is applied equally in all directions, which prevents high-precision denoising [7, 8].

The NLM filter is another algorithm for reducing noise in images which is particularly effective in the presence of Gaussian noise [9, 10]. This method considers not only the local neighborhood of each pixel but also the entire image when removing noise [15, 18].

Non-Local Means filter performs next steps:

1. Patches formation, where image is divided into patches, one for each pixel.

2. Patches similarity calculation, i.e. computing similarity between patches for each patch, e.g. using Gaussian kernel.

3. Noise reduction, where each pixel in the image is recalculated as a weighted average of the colors of similar patches. Weights are computed based on patches similarity and a smoothing parameter (typically Gaussian).

ISSN 2617-4316 (Print)
ISSN 2663-7723 (Online)

4. Image Restoration, where denoised image is obtained via combining the weighted averages for all pixels.

For I – is a noised image of $M*N$ size, P is area around a pixel that equals $2p+1$.

For each pixel $I(i,j)$ the mathematical model of non-local means will be the next:

$$I(i,j)=\frac{1}{C(i,j)}\sum_{m=-p}^{p}\sum_{n=-p}^{p}\omega\,,$$
$$(i+m,j+n)*I(i+m,j+n)$$

where $\omega(i+m,j+n)$ is a weight which determines block similarity, $C(i,j)$ is a normalization factor that proceeds with weights normalization and summing:

$$C(i,j)=\sum_{m=-p}^{p}\sum_{n=-p}^{p}\omega(i+m,j+n)\,.$$

Weight $\omega(i+m,j+n)$ can be computed as:

$$w(i+m,j+n)=e^{(-\frac{\|P(i,j)-P(i+m,j+n)\|^2}{h^2})},$$

where $P(i,j)$ represents patch centered at (i,j), $P(i+m,j+n)$ represents patch centered at $(i+m,j+n)$, h is a filtering parameter that controls smoothing.

Non-local means algorithm has proven to be effective to reduce different types of noise, like Gaussian, impulse, anisotropic, film grain noise, etc. Though, its eats up a lot of resources to compute block similarities across the whole image [21].

Fast Fourier Transform is a highly efficient method that computes Discrete Fourier Transform of a sequence, while reducing the algorithm complexity from $O(n^2)$ to $O(n\log n)$ [11, 12]. FFT expects $e^{-2\pi/n}$ to be primitive root of unity for n to make the calculations more efficient [13, 17].

Fast Fourier Transform involves next steps:

1. Optionally transform image so that its dimensions values are powers of 2. This might speed up the process.

2. Fast Fourier Transform when image is transformed from the spatial to the frequency domain and the frequency spectrum is shifted to the center for easier analysis and manipulation.

3. Frequency analysis to find high magnitude areas, which usually represent the noise.

4. Noise removal using different mask and/or filters which can be applied to different areas.

5. Inverse FFT to bring image back from frequency to spatial domain.

Let's say, that $I(x,y)$ is an image represented in spatial domain, where (x,y) is its dimensions, $F(u,v)$ is that image represented in frequency domain, where (u,v) is its dimensions. Then, an image filtered using Fast Fourier transform is computed as:

$$\hat{F}(u,v)=H(u,v)*F(u,v)\,,$$

where $H(u,v)$ is an applied filter to remove "bright" pixels.

Fast Fourier transform is especially good in dealing with periodic noise which is easily found in frequency domain. Though other types of noises won't be filtered that effectively [20, 22].

Anisotropic diffusion is advanced iterative filtering algorithm [3, 4]. Unlike linear filters such as Gaussian filter, which apply uniform smooth, anisotropic diffusion can change intensity and/or direction of filtering based on the characteristics of the area [23, 24].

Anisotropic diffusion filtering consists of following tasks:

1. Calculate gradient ∇I for each pixel of the image $I(x,y)$ along both dimensions:

$$\nabla I=[\frac{\partial I}{\partial x},\frac{\partial I}{\partial y}].$$

Gradient is used to find regions with edges.

2. Define conductance $c(\nabla I)$ which controls the strength of the smoothing effect based on the gradient and sensitivity parameter k using first function:

$$c(\|\nabla I\|)=e^{-(\|\nabla I\|/k)^2}\,,$$

or second:

$$c(\|\nabla I\|)=\frac{1}{1+(\frac{\|\nabla I\|}{k})^2}.$$

3. Apply anisotropic filter based on calculated conductance:

$$I'(x,y,t+1)=I(x,y,t)+\Delta t*\nabla*(c(\nabla I)\nabla I)\,,$$

where t is the number of iterations; Δt is the rate of diffusion.

4. Iterate – repeat the process until no iterations are left to run.

Anisotropic diffusion is a powerful algorithm to filter images. Its complexity allows for flexible image filtering and noise removal combined with edges preservation. The downside is that the anisotropic diffusion is very sensitive to parameters and in most cases will need fine-tuning.

FORMAL PROBLEM STATEMENT

The objective of this work is to analyze existing algorithms, experimentally test their efficiency and applicability and find the most effective approach for image filtering. It might be single algorithm as well as several in pair. Results might serve as a potential improvement to video surveillance systems, making them more accessible and less dependent on factors such as environmental influences and computational power levels.

In line with the research objective, a complex of interrelated problems has been addressed:

- defining the means of evaluation image processing results;
- analyzing existing image optimization methods, their effectiveness, and the potential for integration;
- collecting a comprehensive set of methods for processing and optimizing images, utilizing noise filtering, and brightness/contrast adjustment.

Object of the Research – the processes of image filtering.

Subject of the Research – optimizing image enhancement and processing techniques.

TESTING OF FILTERING ALGORITHMS

Algorithms testing is conducted using Python 3.12.4 with next set of imported modules:

- opencv-python==4.10.0;
- matplotlib==3.9.2;
- numpy==2.1.0;
- skimage==0.24.0;
- os;
- time.

An RGB image, 400x366 pixels, of a lamp (Fig. 1) was chosen for testing purposes. The reasoning behind this choice is simple: this picture features uniform background, different colors, shadows and complex texture. All these aspects can be compared between original and denoised images to make visual evaluation more precise. The dimensions are also good, staying in the middle between low resolution images, averaging up to 250 pixels, and HD pictures, which are 720 pixels wide.

Fig. 1. **Original lamp picture**

Source: **https://alivecolors.com/img/examples/alivecolors/lamp.jpg**

Testing stand configuration is the next:

- CPU - AMD Ryzen 7 5800X3D;
- SSD - Samsung SSD 990 PRO;
- RAM - DDR4 Corsair Vengeance;
- No GPU was involved in conducted computations.

For results evaluation several metrics were considered like Structural Similarity Index (SSIM), Mean Absolut Index (MAE), Normalized Cross-Correlation (NCC), Root Mean Square Error (RMSE) and Peak Signal-to-Noise-Ratio (PSNR) [4]. Mean Absolut Index was dropped due to its nature – it completely ignores the visual perception while only evaluating differences in pixels, thus visually denoised images produce incomparable high values. Same goes for NCC – it is mainly used for similarity comparison and ignores such features as pixel intensity, noise, etc. Mean Absolut Index values are always close to 1 indicating near to none difference between the original and the denoised image, which is not always true. Structural Similarity Index (SSIM) is very sensitive to even slight amount of noise and showed consistent results only for salt-and-pepper and periodic types of the noise. Hence, it can't be used as a general metric. Root Mean Square Error represents the average difference in pixels between original and processed image. It produces meaningful values that correlate with visual results. Peak Signal-to-Noise-Ratio (PSNR) represents the quality of filter comparing pixel intensities of the denoised and the original image and is based on RMSE values.

Mean and Gaussian filters are considered ones of the simplest. They are used mostly as a preparation mean before other filtering techniques applica-

ISSN 2617-4316 (Print)
ISSN 2663-7723 (Online)

tion. Key concept behind those algorithms is that they simply apply average value of neighboring pixels to a target one, only that Gaussian filter weighs central pixels more shifting smoothing effect to the center [14, 16]. Thus, slightly loosing picture information and reducing noise intensity.

The image in Fig. 2 proves it – noise lost in density, though some textures of the lamp became less visible.

While PSNR values are pretty good, close to 30, RMSE values are too high. Gaussian noise introduces pixels varying both in color and intensity which makes it difficult to remove them using simple algorithms like mean operation. Increasing kernel values will dramatically increase smoothing effect which leads to loosing texture and sharpness.

Film grain noise is easier to remove because of the nature of the noise. Pixels are only slightly deviated along [0, 255] resulting in greyish noise which makes it easier to enhance the picture even using simple averaging. Though some noise is still visible it is smoothed after the filtering. Increasing kernel values results in the same behavior as for the Gaussian filter.

For example, “salt-and-pepper” noise introduces pixels maxed out to 0 or 255, rendering mentioned filters useless. In this case median filter will shine. It sorts all neighboring pixels and applies median value to the target one. Meaning maxed out in white or black pixels will be outweighed by normal pixels. That is both the advantage and disadvantage, since high quality pixels might be outweighed too.

The image shown in Fig. 3 is a result of a median filtering. It shows outstanding performance dealing with salt and pepper noise. RMSE 3.7 indicates almost no difference between denoised and original image. PSNR 37.7 shows high performance of the filter.

Another algorithm that is highly effective in one specified task is FFT. It is a great tool to help remove periodic noise. Its nature itself makes it highly recognizable in the frequency domain.

Inside its image is shown in form of summed sine and cosine functions with different frequencies, where high frequencies, located at the edges, can represent the noise. Different approaches can be used to modify the picture: low-pass filter to mask edges of the frequency domain removing noise, high-pass filter to suppress low frequencies emphasizing edges, notch filter to remove specific frequencies, for example ones represent the periodic noise.

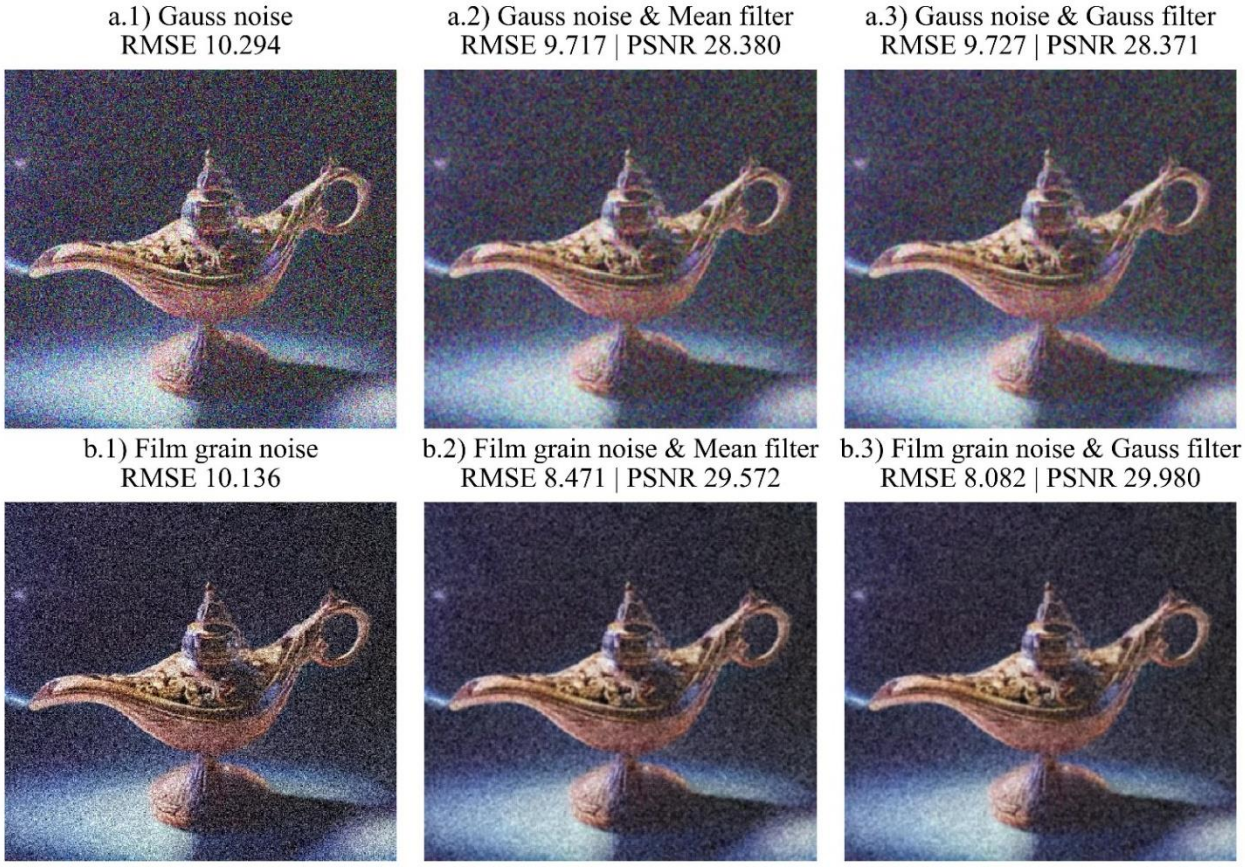

Fig. 2. **Metrics of Gauss and mean filters applied to Gauss and film grain noise**

Source: **compiled by the authors**

Fig. 3. **Metrics of a median filter applied to noisy image**

Source: **compiled by the authors**

The Fig. 4 shows the combined usage of FFT and notch filter used to remove high frequency periodic noise. The color intensity became more even, noise and its effect on the lamp are removed and only slight artifacts are present on the background. RMSE and PSNR are of good values, 2.9 and 38.9 respectively, indicating low difference level between denoised and original image and high efficiency of the algorithm.

Python code below was used to perform FFT, apply notch filter and perform inverse FFT:

```
f_transform = np.fft.fft2(image)
center_shift = np.fft.fftshift(f_transform)
k, l, frequency = 1, 9, 17
rows, cols = image.shape
crow, ccol = rows // 2, cols // 2
noise_frequency = int(frequency * cols / rows)
center_shift[crow - k:crow + k, ccol - noise_frequency - l:ccol - noise_frequency + l] = 0
center_shift[crow - k:crow + k, ccol + noise_frequency - l:ccol + noise_frequency + l] = 0
f_shift = np.fft.ifftshift(center_shift)
denoised_image = np.abs(np.fft.ifft2(f_shift))
return np.clip(denoised_image, 0, 255).astype(np.uint8)
```

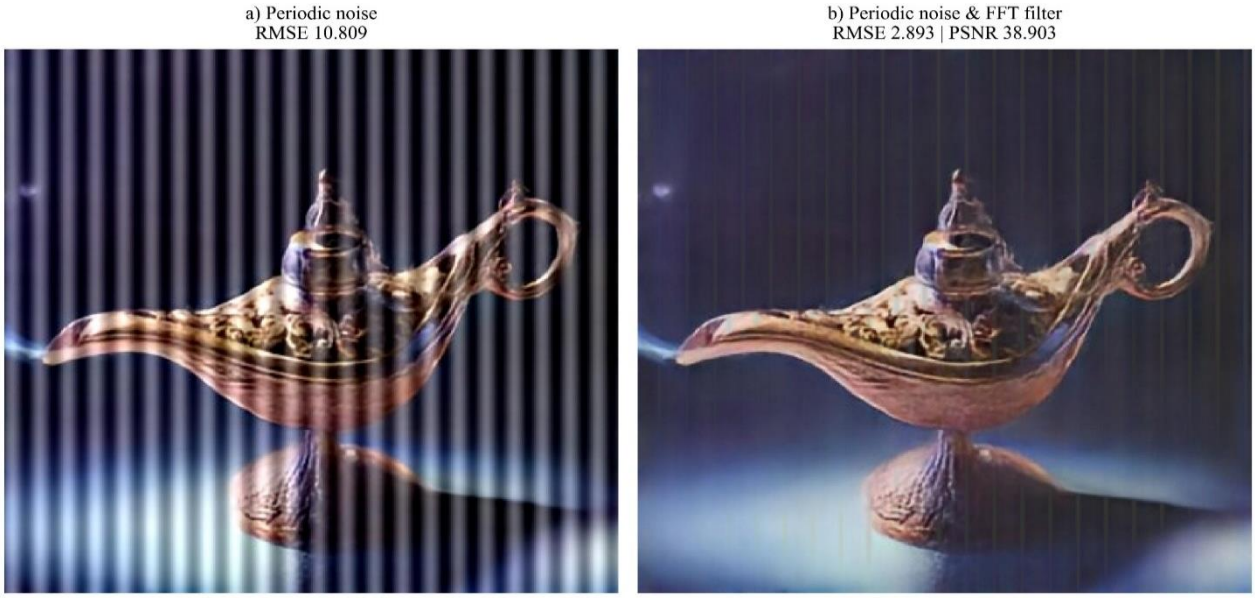

Fig. 4. **Usage of Fast Fourier Transform to filter out periodic noise**

Source: compiled by authors

ISSN 2617-4316 (Print)
ISSN 2663-7723 (Online)

The picture is shifted to frequency domain and transposed to move low frequencies to the middle. Then a notch filter is applied relative to the center of the picture to remove specific frequencies representing the periodic noise.Inverse shift is applied after to bring the image back to the spatial domain. All pixel values are then normalized to fit into [0, 255] range so that image can be saved as 8 bit per channel. The given code snippet is then applied to each color channel of the image.

Parameters *k, l, frequency* are the same for each channel, though can be configured beforehand. Concurrent processing of the channels introduced no gain in performance.

The downsides of the approach are:

- it is not general filtering approach and is usually used specifically to remove periodic noise;
- it is very dependant on the noise features so weighs must be experimentally set for each image.

Though there are advantages of using the FFT:

- it is computationally effective and performs well even when processing of 3 channels pictures;
- using notch filter gives flexibility in composing mask, allowing very precise frequency removal;
- modifying of frequencies allows not only to remove the noise but also emphasize edges and increase sharpness of the image.

Another strong filtering algorithm is NLM. It is close to smoothing algorithms like Gaussian and mean filter. The difference is that instead of computing mean value based on the neighboring pixels, it computes it based on alike pixels in the picture [19].

The below picture (Fig. 5) is a result of NLM filter work. Several pictures with different types of noises were prepared. Looking at RMSE and PSNR values, NLM is a good general-purpose filter.

Visual assessment proves high performance of the filter. Even image with random anisotropic noise not only has good metrics but is visually much cleaner. The only artifact in this case is picture with Gauss noise. While visually it looks like original, RMSE value is high. This happened due to color component of the noise.

Though NLM filter is strong general-purpose filter it has two major disadvantages.

First, it is highly dependable on the parameters. If the sensitivity or window size is set to the wrong value, the filter will perform poorly.

Second, it is very computationally intensive. It is one of the slowest filters. It must run through the whole picture several times to be able to apply accurate mean to the pixels. In practice it is nearly impossible to use NLM filter in real-time systems – processing of single image might take up to 0.7 seconds.

Anisotropic diffusion might be a good substitution for NLM filter. It is somewhat close to Gaussian smoothing, but is a non-linear filter. Anisotropic diffusion can apply smoothing to different directions, which is controlled by the diffusion coefficient calculated from local gradients. The lower the gradient the stronger the smoothing effect is. Also, it works iteratively allowing better control on the smoothing effect.

The Fig. 6 depicts the results of anisotropic diffusion filtering.

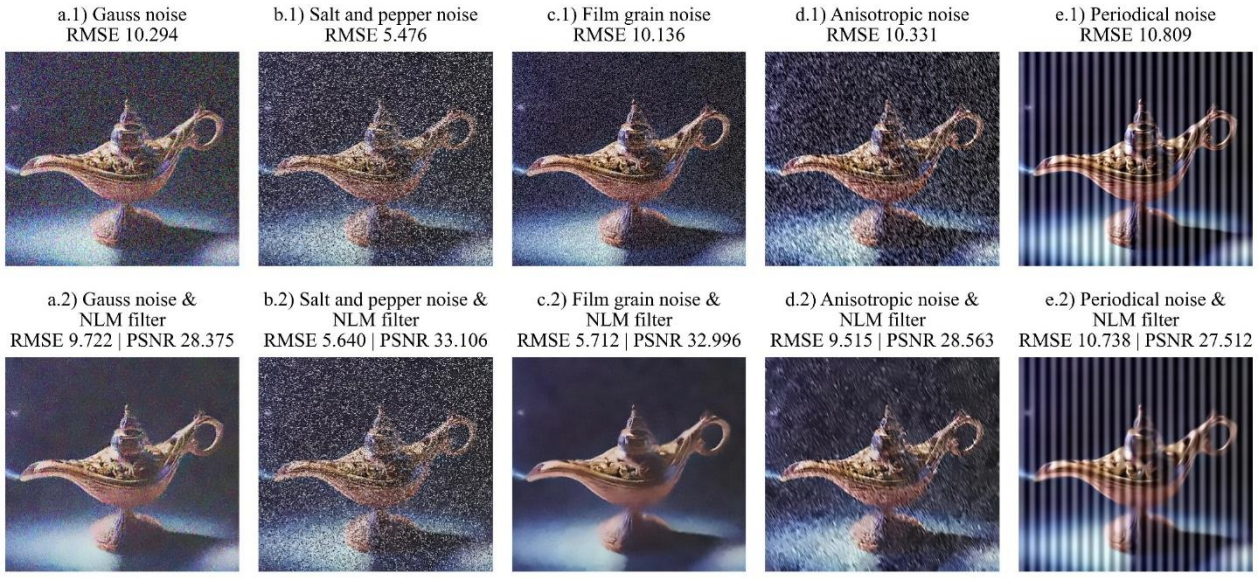

Fig. 5. **NLM filter applied to different noise types**

Source: **compiled by the authors**

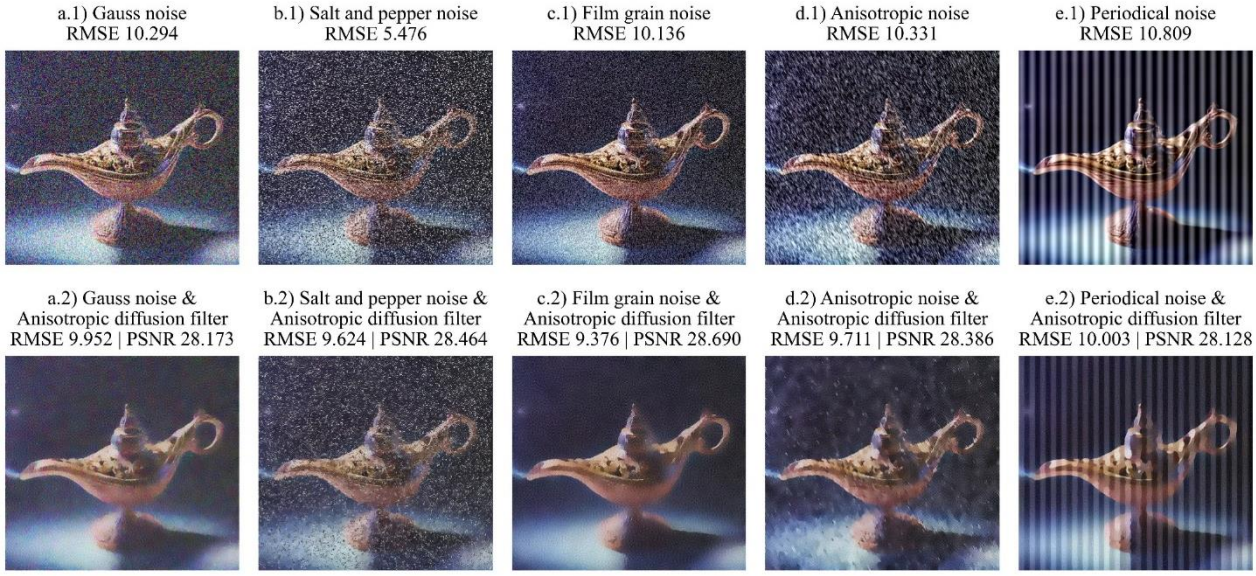

Fig. 6. **Anisotropic diffusion applied to different types of noise**

Source: **compiled by the authors**

Evaluation metrics state anisotropic diffusion is less accurate and efficient - RMSE never drops below 9 and there is not a single PSNR that is equal to 30 or higher. But, the execution time per image is close to half of one tenth of a second, which would be even less in case of grayscale image. And visual assessment shows that the results are very close to the NLM filtering results.

While it might seem dependent on the parameters, there are effective bounds, crossing which will introduce similar image artifacts in any type of the scene. The parameters are simple in configuration:

- ∇t , which controls the time diffusion is applied. Usually, it is set to 0.5 for perfect balance between smoothing and preserving edges;
- *kSize* – kernel size usually equals to 3 because higher values introduce tearing and artifacts;
- *i* – number of iterations. Unless a strong smoothing effect is required and it is allowed to lose edges and texture, it should not be higher than 15;
- *kappa* – sensitivity of a diffusion process. It controls the strength of the smoothing effect and usually should equal to 15.

The below Python code was used to perform the filtering:

```
img = np.asarray(image, dtype=float_data)
for _ in range(iterations):
    img_dx = cv2.Sobel(img, cv2.CV_64F, 1, 0,
ksize=kSize, borderType=cv2.BORDER_DEFAULT)
    img_dy = cv2.Sobel(img, cv2.CV_64F, 0, 1,
ksize=kSize, borderType=cv2.BORDER_DEFAULT)
    grad_mag = np.sqrt(img_dx**2 + img_dy**2)
    c = 1 / (1 + (grad_mag / kappa)**2)
    diff_x = cv2.Sobel(c * img_dx, cv2.CV_64F, 1, 0,
ksize=kSize, borderType=cv2.BORDER_DEFAULT)
    diff_y = cv2.Sobel(c * img_dy, cv2.CV_64F, 0, 1,
ksize=kSize, borderType=cv2.BORDER_DEFAULT)
    img += delta_t * (diff_x + diff_y)
    img = np.clip(img, 0, 255)
    return img.astype(uint_data)
```

First, we transform image to array of floating-point values for a better precision. Then the gradients *img_dx* & *img_dy* are computed using the Sobel operator which are used to identify edges in the image. The gradient magnitude *grad_mag* is calculated using the gradients to evaluate the strength of the edges. Diffusion coefficient *C* is a function that controls the diffusion. *diff_x* & *diff_y* are the actual smooth effects that will be applied using *delta_t* multiplier. All these operations are performed *iterations* times. In the end image is converted back to 8-bit color channels.

DISCUSSION OF THE RESEARCH RESULTS

Finally, all described filters were tested and compared for performance and results are displayed in the Table 1. Each filter was measured 100 times and the mean value was taken.

Table 1. **Filters` mean execution time**

	Filter names					
	Mean blur	Gauss blur	Median blur	FFT transform	NLM filter	Anisotropic diffusion
Mean execution time, sec	0.01	0.01	0.01	0.03	0.7	0.11

Source: **compiled by the authors**

Gaussian and mean filters were tested on Gaussian and film grain noise. They work much faster than other filters, around 0.01 second, though give the least results: RMSE 9.7 and PSNR 28.3 in average for Gaussian filter and RMSE 8.3 and PSNR 29.5 in average for mean filter with poor visual results.

The results were combined using the next formula:

$$q = \frac{PSNR}{RMSE}, \quad (1),$$

where q is a ratio of a clear signal to the difference between original and denoised image, then Gaussian and mean filters would have q 2.92 and q 3.5 respectively.

Median filter is usually used for "Salt-and-pepper" noise and was tested that way. It shows the same execution time as Gaussian and mean filters in average with good metric results: RMSE 3.3 and PSNR 37.7 which, using formula (1), yields high q 11.4.

FFT technique is also fast – 0.03 to 0.05 seconds and provides high level of image denoising – both according to visual assessment and metrics: RMSE 2.9, PSNR 39 and q 13.4. It allows to use different configurable filtering approaches to denoise and/or sharpen the image. The only downside is that it is only applicable for periodic noise.

NLM filter is the slowest one: 0.7 seconds in average while giving the best results – both visually and according to metrics:

- RMSE 10.1, PSNR 28, q 2.8 for Gaussian noise with good visual quality;
- RMSE 6.1, PSNR 32.4, q 5.3 for "Salt-and-pepper" noise with almost no changes in visual quality;
- RMSE 5.5, PSNR 33.4, q 6.1 for film grain noise with good visual quality;
- RMSE 9, PSNR 29, q 3.2 for anisotropic noise with acceptable visual quality;
- RMSE 10.6, PSNR 27.6, q 2.6 for periodic noise with almost no changes in visual quality.

Average metric values will be RMSE 8.3, PSNR 30.1, q *4* proving NLM filter is a good general-purpose tool.

Anisotropic diffusion takes the strong middle – it takes less time to process the image, 0.1 to 0.15 seconds, while can be configured much easier to find a good balance between performance and quality. As for the results, it gives slightly worse metric results and almost the same as NLM filter visual results:

- RMSE 10, PSNR 28.1, q 2.8 for Gaussian noise with good visual quality;
- RMSE 9.4, PSNR 28.6, q 3 for "Salt-and-pepper" with slightly reduced noise intensity;
- RMSE 9.4, PSNR 28.6, q 3 for film grain noise with good visual quality;
- RMSE 9.8, PSNR 28.3, q 2.9 for anisotropic noise with acceptable visual quality;
- RMSE 10.4, PSNR 27.7, q 2.7 for periodic noise with slightly reduced noise intensity.

Average metric values will be RMSE 9.8, PSNR 28.3, q 2.9 showing that anisotropic diffusion might give slightly worse results calculation-wise, while visual assessment proves the opposite. All tested combinations results can be found in following Table 2. Combinations not mentioned in the table produced extremely poor visual quality results and thus were not included at all.

On top of that each filter was tested with different level of chosen noise:

1) Gauss filter for Gauss noise;
2) Mean filter for film grain noise;
3) Median filter for salt and pepper noise;
4) FFT filter for periodic noise;
5) NLM for anisotropic noise;
6) Anisotropic diffusion for anisotropic noise.

All filter were used on appropriate noise types except for NLM and anisotropic filters – they were tested on the most difficult anisotropic noise.

The plots showing the result of filtering different noises types are shown in fig. 7–12.

Table 2. **Filter-noise comparison table**

Noise types	Noised RMSE	Filter types																	
		Gaussian			Mean			Median			FFT			NLM			Anisotropic		
		RMSE	PNSR	q	RMSE	PNSR	q	RMSE	PNSR	q	RMSE	PNSR	q	RMSE	PNSR	q	RMSE	PNSR	q
Gaussian	10.3	9.7	28.3	2.9	9.7	28.3	2.9							10.1	28	2.8	10	28.1	2.8
Salt-and-pepper	5.5							3.3	37.7	11.4				6.1	32.4	5.3	9.4	28.6	3
Film grain	10.1	8.1	29.9	3.7	8.5	29.5	3.5							5.5	33.4	6.1	9.4	28.6	3
Aniso-tropic	10.3													9	29	3.2	9.8	28.3	2.9
Periodic	10.8										2.9	39	13.4	10.6	27.6	2.6	10.4	27.7	2.7

Source: **compiled by the authors**

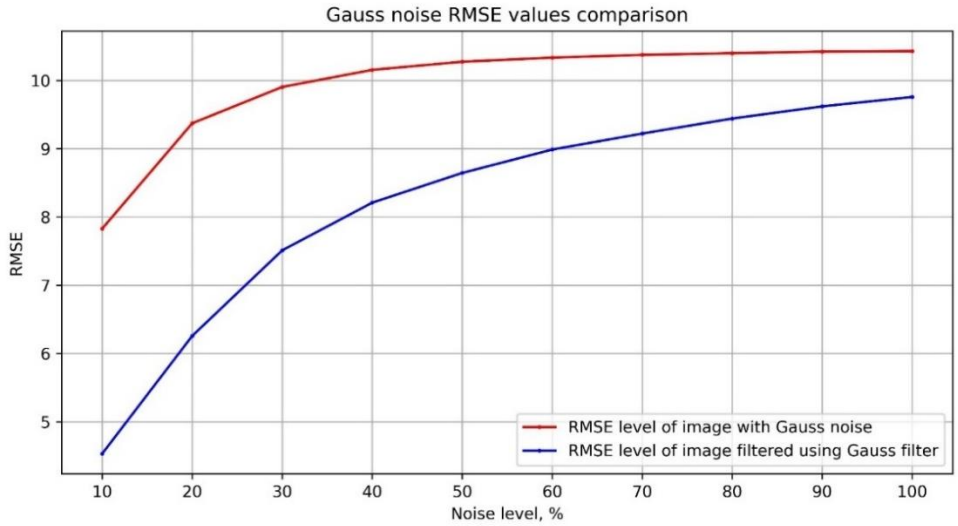

Fig. 7. **Root Mean Square Error values for Gauss noise and Gauss filter**
Source: **compiled by the authors**

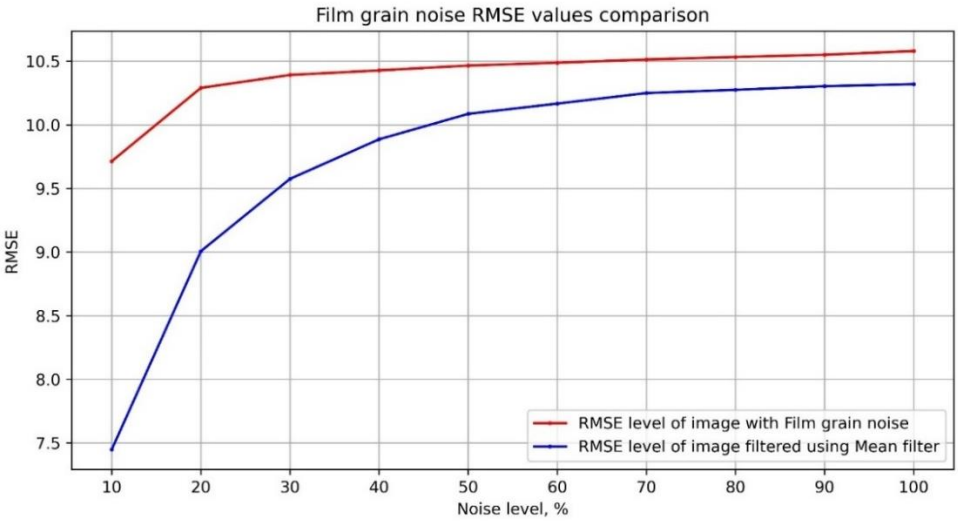

Fig. 8. **Root Mean Square Error values for film grain noise and Mean filter**
Source: **compiled by the authors**

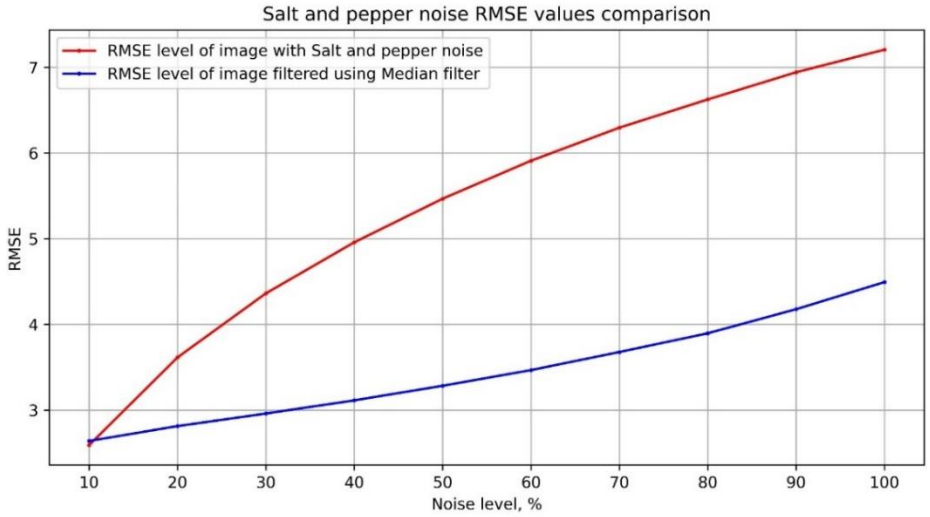

***Fig. 9.* Root Mean Square Error values for salt and pepper noise and Median filter**
***Source:* compiled by the authors**

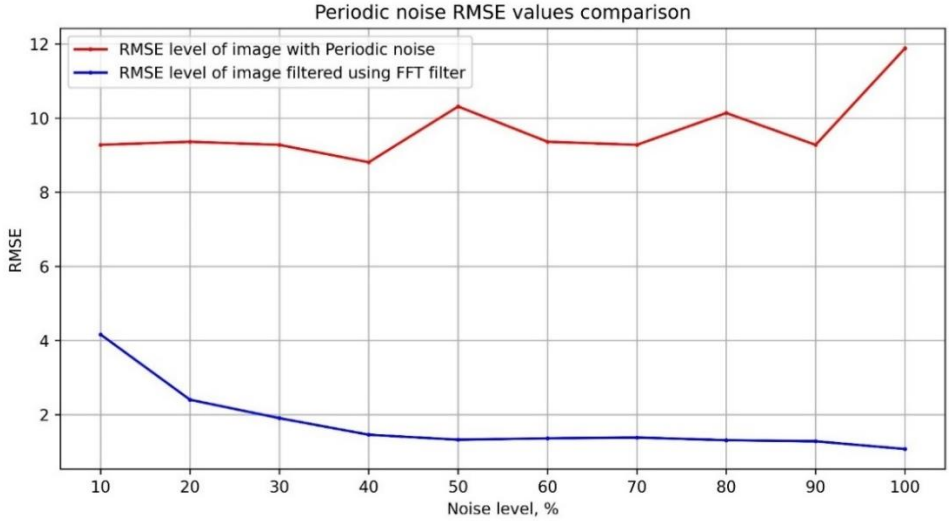

***Fig. 10.* Root Mean Square Error values for periodic noise and FFT transform**
***Source:* compiled by the authors**

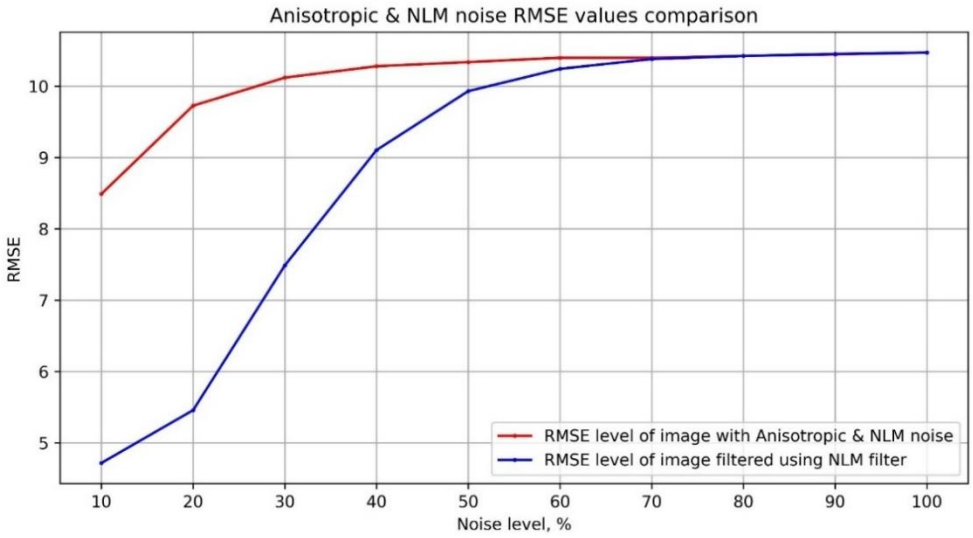

***Fig. 11.* Root Mean Square Error values for anisotropic noise and NLM filter**
***Source:* compiled by the authors**

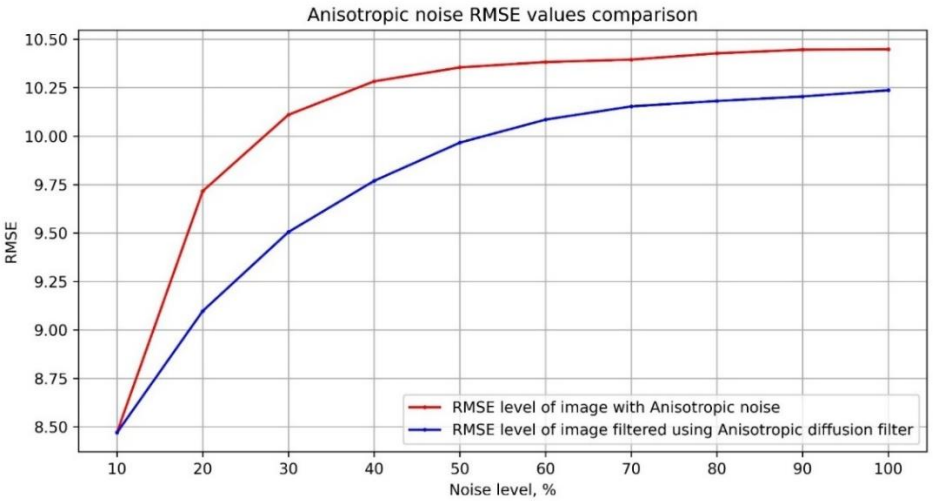

***Fig. 12.* Root Mean Square Error values for anisotropic noise and anisotropic filter**
***Source:* compiled by the authors**

CONCLUSION

Following a series of tests, including quality and performance evaluations, the following conclusions were drawn:

Anisotropic diffusion can be employed as the primary filtering method in both standalone systems and in the manual processing of individual images. The flexibility of the tuning process allows for a variety of results due to the extensive parameterization capabilities. Furthermore, the method is computationally efficient and has the potential for further algorithmic improvements.

The FFT is applicable in both standalone systems and manual processing. Furthermore, the system is capable of self-tuning, whereby the discoloration parameters are set based on the characteristics of the noise, including its frequency and magnitude. Nevertheless, this approach is only applicable to periodic noise and may not be appropriate for other types of noise.

The median filter, which is frequently mistaken for the mean filter, is an effective method for removing impulsive noise, such as "salt and pepper" noise. Such artefacts may be filtered out by hiding pixels with extreme values. Nevertheless, its efficacy is significantly diminished when applied to other forms of noise.

The NLM algorithm is highly efficient, but its computational intensity represents a significant challenge. The method yields satisfactory results with a wide range of noise types and can be parameterized to regulate the ratio of blurring and smoothing effects. The primary disadvantage of this method is its high processing time, which makes it unsuitable for real-time image processing but ideal for single-image analysis.

The Gaussian and mean filters, being the simplest, do not produce meaningful results when used in isolation, as would be expected. However, they are of value when used as components of more complex algorithms. Linear filters are fast but produce poor results and are best used as preparatory measures. Nonlinear filters are more robust and applicable to different types of noise, although they require fine-tuning of parameters. This research will be extended to explore the application of these filtering techniques in various real-world scenarios and autonomous systems.

REFERENCES

1. Russ, J. C. & Brent Neal, F. “The image processing handbook”. *CRC Press;* 7-th edition. 2016.
2. Howse, J. & Minichino, J. “Learning OpenCV 4 Computer Vision with Python3”. *PACKT Publishing*; 3-rd edition. 2022.
3. Szeliski, R. “Computer Vision: Algorithms and Applications”. *Springer;* 2-nd edition. 2022.
4. Krig, S. “Computer vision metrics: Survey, taxonomy, and analysis”. *Apress;* 1-st edition. 2014.
5. Russ, J. C. & Brent Neal, F. “Computer Vision: Principles, Algorithms, Applications, Learning”. *Academic Press;* 5-th edition. 2017.
6. Polyakova, M. V., Kozak, D. Y. & Huliaieva, N. A. “Comparative analysis of classifiers for face recognition on image fragments identified by the FaceNet neural network”. *Herald of Advanced Information Technology.* 2022; 5 (2): 91–101. DOI: https://doi.org/10.15276/hait.05.2022.7.

7. Sheikh, T. & Raghad, R. “A comparative study of various image filtering techniques for removing various noisy pixels in aerial image”. *International Journal of Signal Processing, Image Processing and Pattern Recognition.* 2016; 9 (3):113–124 DOI: http://doi.org/10.14257/ijsip.2016.9.3.10.

8. Kumar, A. A., Lal, N. & Kumar, R. N. “A comparative study of various filtering techniques". *5th International Conference on Trends in Electronics and Informatics (ICOEI)*. Tirunelveli, India. 2021. p. 26–31. DOI: https://doi.org/10.1109/ICOEI51242.2021.9453068.

9. Mirataei, A., Rusanova, O., Tribynska, K. & Markovskyi, O. “Organization of protected filtering of images in clouds”. *Visnik of National Technical University of Ukraine “KPI” Informatics, Control and Computer Engineering*. 2022; 3: 49–55. DOI: https://doi.org/10.20535/2708-4930.3.2022.269132.

10. Hrytsyk, V. & Berezhky, O. “Methods and high-performance parallel systems for real-time image processing and recognition”. *International Journal of Computing*. 2014; 2: 25–34. DOI: https://doi.org/10.47839/ijc.2.1.159.

11. Pavlenko, V., Pavlenko, S. & Speranskyy, V. “Identification of systems using Volterra model in time and frequency domain (Chapter 10)”. *Advanced Data Acquisition and Intelligent Data Processing*. 2014. p. 233–270. DOI: https://doi.org/10.1201/9781003337027-10.

12. Mirataei, A., Khalil, H. & Markovskyi, O. “Protected discrete Fourier transform implementation on remote computer systems”. *Visnik of National Technical University of Ukraine “KPI” Informatics, Control and Computer Engineering*. 2020; 1: 26–33. DOI: https://doi.org/10.20535/2708-4930.1.2020.216050.

13. Roy, V. & Shukla, S. “Spatial and transform domain filtering method for image de-noising: A Review”. *International Journal of Modern Education and Computer Science (IJMECS)*. 2013; 5 (7): 41–49, DOI: https:/doi.org/10.5815/ijmecs.2013.07.05.

14. Ramadan, Z. M. “Reduction of mixed impulsive noise with Gaussian and speckle”. *Journal of Engineering Science and Technology*. 2023; 15 (7): 2722–2735.

15. Ramadan, Z. M. “An innovative kernel function for the NLM filtering”. *ARPN Journal of Engineering and Applied Sciences*. 2020; 18 (6): 882–889.

16. Ramadan, Z. M. “Effect of kernel size on Wiener and Gaussian image filtering”. *TELKOMNIKA*, 2019; 17: 1455–1460, DOI: https://doi.org/10.12928/TELKOMNIKA.v17i3.11192.

17. Ramadan, Z. M. “Optimum Image Filters for Various Types of Noise”. *TELKOMNIKA*. 2018; 16 (3): 2458–2464. DOI: http://doi.org/10.12928/telkomnika.v16i5.10508.

18. Zhou, S. et al. “A Novel Fast NLM denoising algorithm based on improved kernel function parameters: Applied to RFID Labels”. *IEEE 4th Advanced Information Management, Communicates, Electronic and Automation Control Conference (IMCEC)*. Chongqing, China. 2021. p. 1609–1613. DOI: https://doi.org/10.1109/IMCEC51613.2021.9481961.

19. Shreyamsha Kumar, B. K. “Image denoising based on non-local means filter and its method noise thresholding”. *SIViP*. 2013; 7: 1211–1227. DOI: https://doi.org/10.1007/s11760-012-0389-y.

20. Shreyamsha Kumar, B. K. “Multifocus and multispectral image fusion based on pixel significance using discrete cosine harmonic wavelet transform”. *SIViP*. 2013; 7 (6): 1125–1143. DOI: https://doi.org/10.1007/s11760-012-0361-x.

21. Panigrahi, S. K. Gupta, S. & Sahu, P. K. “Curvelet thresholding with multiscale NLM filtering for color image denoising”. *TENCON 2017-2017 IEEE Region 10 Conference*. Penang, Malaysia. 2017. p. 2220–2225. DOI: https://doi.org/10.1109/TENCON.2017.8228230.

22. “Fast Fourier Transform (FFT)”. *In: Padua, D. (eds) Encyclopedia of Parallel Computing. Springer, Boston, MA*. 2011. DOI: https://doi.org/10.1007/978-0-387-09766-4_2077.

23. Nie, Z. & Wang, H. “Application of improved anisotropic diffusion filtering in seismic data denoising”. *European Association of Geoscientists & Engineers - 84th EAGE Annual Conference & Exhibition*. 2023; 2023: 1–5. DOI: https://doi.org/10.3997/2214-4609.202310629.

24. Lennon, M., Mercier, G. & Hubert-Moy, L. “Nonlinear filtering of hyperspectral images with anisotropic diffusion”. *IEEE International Geoscience and Remote Sensing Symposium*. Toronto, Canada. 2002; 4: 2477–2479. DOI: https://doi.org/10.1109/IGARSS.2002.1026583.

25. Chong, Kwen Siong, “Design implementation low energy fast Fourier transform/inverse fast Fourier transform (FFT/IFFT) processor based on asynchronous-logic”. *Doctoral thesis, Nanyang Technological University*. Singapore. 2007. DOI: https://doi.org/10.32657/10356/3447.

Conflicts of Interest: The authors declare that there is no conflict of interest

Received 12.08.2024
Received after revision 16.09.2024
Accepted 21.09.2024

DOI: https://doi.org/10.15276/aait.07.2024.18
УДК 004.932.4

Аналіз методів та алгоритмів фільтрації і покращення якості зображень

Сперанський Віктор Олександрович[1)]
ORCID: https://orcid.org/0000-0002-8042-1790; speranskyy@op.edu.ua. Scopus Author ID: 54401618900
Балабан Даніїл Сергійович[1)]
ORCID: https://orcid.org/0009-0004-2550-0586; danches.balaban@gmail.com
[1)]Національний університет «Одеська політехніка», пр. Шевченка, 1. Одеса, 65044, Україна

АНОТАЦІЯ

Стаття описує актуальну проблему зашумленості зображень і методи її вирішення. Проведено аналіз, тести та опис різних фільтрів, зокрема сфери їх застосування. Методи фільтрації розподілено на дві групи: лінійні, такі як фільтр Гауса та середнього значення, а також нелінійні, такі як медіанна фільтрація, швидке перетворення Фур'є (ШПФ), метод нелокальних середніх (НЛС) та анізотропна дифузія. Кожен фільтр описано математично, реалізовано з використанням мови програмування Python та протестовано на RGB-зображеннях. Робота описує критерії оцінювання, їх переваги та недоліки. Середньоквадратична помилка (СКП) та пікове співвідношення сигнал/шум (ПССШ) використовуються як критерії для аналізу ефективності алгоритмів. Також взято до уваги швидкість роботи кожного алгоритма. Загалом, експериментальні дані свідчать про те, що лінійні фільтри працюють швидше, але дають гірші результати і краще за все використовуються на підготовчому етапі. Нелінійні фільтри є більш надійними і можуть бути застосовані для різних типів шуму, хоча мають недолік у вигляді необхідності тонкого налаштування параметрів. Дослідження показує, що анізотропну дифузію можна використовувати як для ручної обробки зображень, так і для застосувань у реальному часі, оскільки вона забезпечує хороший компроміс між швидкістю обробки та якістю очищеного від шуму зображення. НЛС підходить для високоякісної обробки окремих зображень через свою низьку швидкість та високу якість обробленого зображення. Насамкінець, ШПФ виділяється своєю ефективністю у видаленні періодичного шуму. Ця стаття матиме продовження у вигляді розвитку та використанні методів фільтрації в різних реальних реалізаціях та автономних системах.

Ключові слова: фільтрація зображень; покращення якості зображень; лінійні фільтри; нелінійні фільтри; метрики фільтрації.

ABOUT THE AUTHORS

Viktor O. Speranskyy - Candidate of Engineering Sciences, Associate Professor, Computerized Systems and Software Technologies Department. Odesa Polytechnic National University, 1, Shevchenko Ave. Odesa, 65044, Ukraine
ORCID: https://orcid.org/0000-0002-8042-1790; speranskiyva@ukr.net. Scopus Author ID: 54401618900
***Research field*:** Nonlinear systems; modeling; identification; software development; neural networks

Сперанський Віктор Олександрович - кандидат технічних наук, доцент кафедри Комп'ютеризованих систем та програмних технологій. Національний університет «Одеська політехніка», пр. Шевченка, 1. Одеса, 65044, Україна

Daniil S. Balaban - PhD Student, Computerized Systems and Software Technologies Department. Odessa Polytechnic National University, 1, Shevchenko Ave. Odesa, 65044, Ukraine
ORCID: https://orcid.org/0009-0004-2550-0586; danches.balaban@gmail.com
***Research field*:** Software development; image processing; neural networks.

Балабан Даніїл Сергійович - аспірант кафедри Комп'ютеризованих систем та програмних технологій. Національний університет «Одеська політехніка», пр. Шевченка, 1, Одеса, 65044, Україна