
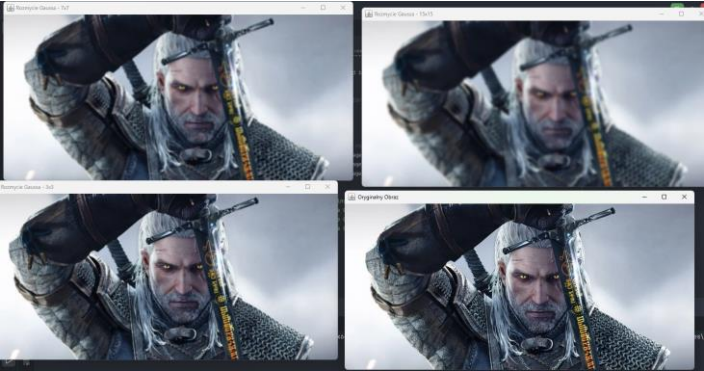
	Politechnika Bydgoska im. J. J. Śniadeckich Wydział Telekomunikacji, Informatyki i Elektrotechniki <b>Zakład Systemów Teleinformatycznych</b>		
<b>Przedmiot</b>	Przetwarzanie obrazów		
<b>Prowadzący</b>	mgr inż. Grzegorz Czczot		
<b>Temat</b>	Filtry		
<b>Student</b>	Paweł Jońca		
<b>Nr lab.</b>	5	<b>Data wykonania</b>	11.11.2024r
<b>Ocena</b>		<b>Data oddania spr.</b>	11.11.2024r

Zad 1.

```

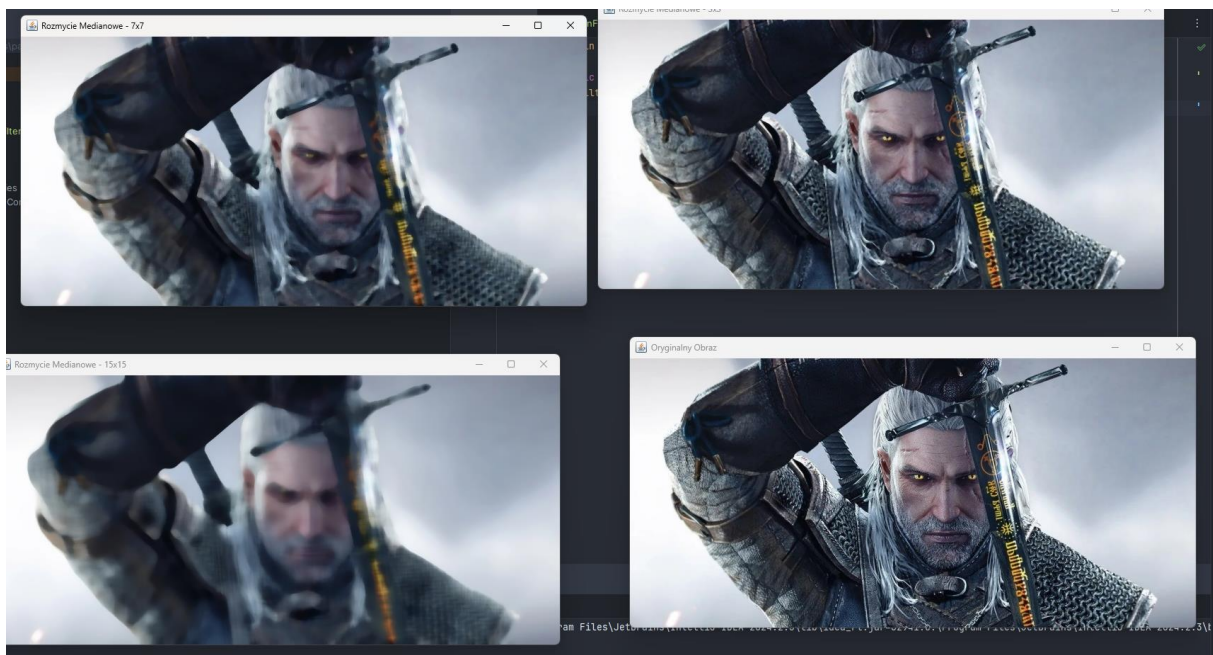
1  import org.opencv.core.Core;
2  import org.opencv.core.Mat;
3  import org.opencv.core.Size;
4  import org.opencv.imgcodecs.Imgcodecs;
5  import org.opencv.imgproc.Imgproc;
6  import org.opencv.highgui.HighGui;
7  //Klasa Gauss zawiera całą logikę przetwarzania obrazu
8  public class Gauss { 1usage & Pawelgabrieljonca
9
10     public static void processImage() { 1usage & Pawelgabrieljonca
11         // Załaduj bibliotekę OpenCV
12         System.loadLibrary(Core.NATIVE_LIBRARY_NAME);
13
14         String imagePath = "geralt.jpg";
15         Mat image = Imgcodecs.imread(imagePath);
16
17         if (image.empty()) {
18             System.out.println("Failed to load image!");
19             return;
20         }
21
22         // Tworzenie obiektów Mat dla przetworzonych obrazów
23         Mat blurredImage1 = new Mat();
24         Mat blurredImage2 = new Mat();
25         Mat blurredImage3 = new Mat();
26
27         // Rozmycie Gaussa dla trzech różnych rozmiarów filtra
28         Imgproc.GaussianBlur(image, blurredImage1, new Size( width: 3, height: 3), sigmaX: 0);
29         Imgproc.GaussianBlur(image, blurredImage2, new Size( width: 7, height: 7), sigmaX: 0);
30         Imgproc.GaussianBlur(image, blurredImage3, new Size( width: 15, height: 15), sigmaX: 0);
31
32         // Wyświetl oryginalny obraz oraz obrazy po rozmyciu
33         HighGui.imshow( winname: "Oryginalny Obraz", image);
34         HighGui.imshow( winname: "Rozmycie Gaussa - 3x3", blurredImage1);
35         HighGui.imshow( winname: "Rozmycie Gaussa - 7x7", blurredImage2);
36         HighGui.imshow( winname: "Rozmycie Gaussa - 15x15", blurredImage3);
37
38         HighGui.waitKey( delay: 0);
39         System.exit( status: 0);
40     }
41 }
42

```



## Zad 2.

```
1 > import ...
6 public class MedianFilter { 1usage new *
7     public static void processImage() { 1usage new *
8         System.loadLibrary(Core.NATIVE_LIBRARY_NAME);
9         String imagePath = "geralt.jpg";
10        Mat image = Imgcodecs.imread(imagePath);
11
12        if (image.empty()) {
13            System.out.println("Failed to load image!");
14            return;
15        }
16        // Tworzenie obiektów Mat dla przetworzonych obrazów
17        Mat medianBlurredImage1 = new Mat();
18        Mat medianBlurredImage2 = new Mat();
19        Mat medianBlurredImage3 = new Mat();
20
21        // Rozmycie medianowe dla trzech różnych rozmiarów filtra
22        Imgproc.medianBlur(image, medianBlurredImage1, ksize: 3); // 3x3
23        Imgproc.medianBlur(image, medianBlurredImage2, ksize: 7); // 7x7
24        Imgproc.medianBlur(image, medianBlurredImage3, ksize: 15); // 15x15
25
26        // Wyświetl oryginalny obraz oraz obrazy po rozmyciu medianowym
27        HighGui.imshow( winname: "Oryginalny Obraz", image);
28        HighGui.imshow( winname: "Rozmycie Medianowe - 3x3", medianBlurredImage1);
29        HighGui.imshow( winname: "Rozmycie Medianowe - 7x7", medianBlurredImage2);
30        HighGui.imshow( winname: "Rozmycie Medianowe - 15x15", medianBlurredImage3);
31
32        HighGui.waitKey( delay: 0);
33        System.exit( status: 0);
34    }
35 }
36 }
```



### Zad 3

```
> import ...

public class BilateralFilter { 1 usage new *
    public static void processImage() { 1 usage new *
        System.loadLibrary(Core.NATIVE_LIBRARY_NAME);
        String imagePath = "geralt.jpg";
        Mat image = Imgcodecs.imread(imagePath);

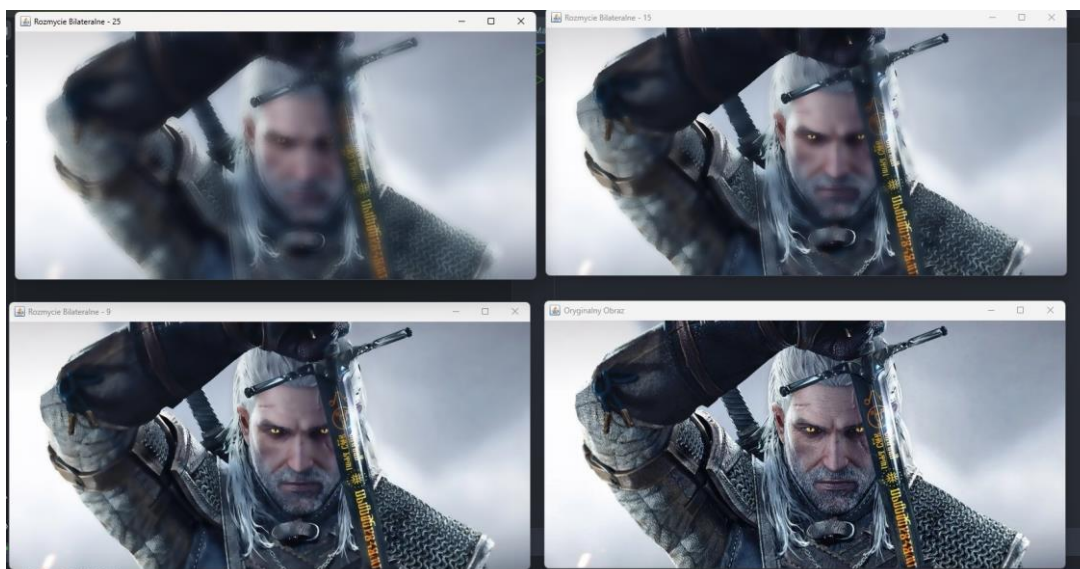
        if (image.empty()) {
            System.out.println("Failed to load image!");
            return;
        }

        // Tworzenie obiektów Mat dla przetworzonych obrazów
        Mat bilateralBlurredImage1 = new Mat();
        Mat bilateralBlurredImage2 = new Mat();
        Mat bilateralBlurredImage3 = new Mat();

        // Rozmycie bilateralne dla trzech różnych rozmiarów filtra
        Imgproc.bilateralFilter(image, bilateralBlurredImage1, d: 9, sigmaColor: 75, sigmaSpace: 75);
        Imgproc.bilateralFilter(image, bilateralBlurredImage2, d: 15, sigmaColor: 150, sigmaSpace: 150);
        Imgproc.bilateralFilter(image, bilateralBlurredImage3, d: 25, sigmaColor: 300, sigmaSpace: 300);

        // Wyświetl oryginalny obraz oraz obrazy po rozmyciu bilateralnym
        HighGui.imshow( winname: "Oryginalny Obraz", image);
        HighGui.imshow( winname: "Rozmycie Bilateralne - 9", bilateralBlurredImage1);
        HighGui.imshow( winname: "Rozmycie Bilateralne - 15", bilateralBlurredImage2);
        HighGui.imshow( winname: "Rozmycie Bilateralne - 25", bilateralBlurredImage3);

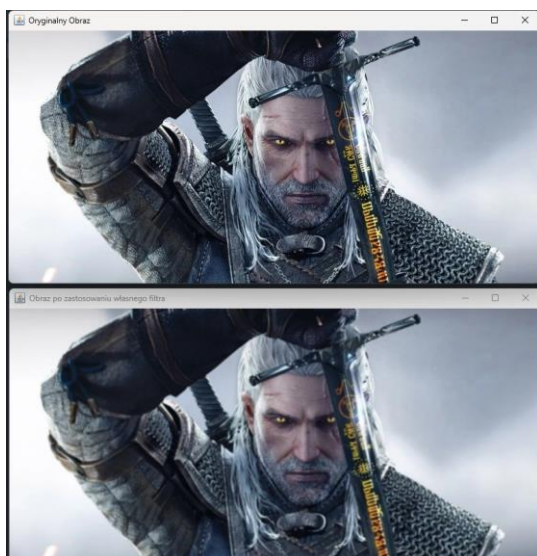
        HighGui.waitKey( delay: 0);
        System.exit( status: 0);
    }
}
```



Filtr medianowy rozmywa cały obraz, w tym krawędzie, więc działa dobrze przy usuwaniu szumu sol-pieprz, ale rozmywa też detale. Filtr bilateralny zachowuje krawędzie, usuwając szumy tylko w obszarach o podobnym kolorze. Jest używany, gdy chcemy usunąć szum, ale zachować ostre krawędzie obiektów, dzięki czemu obraz wygląda bardziej naturalnie.

## Zad 4

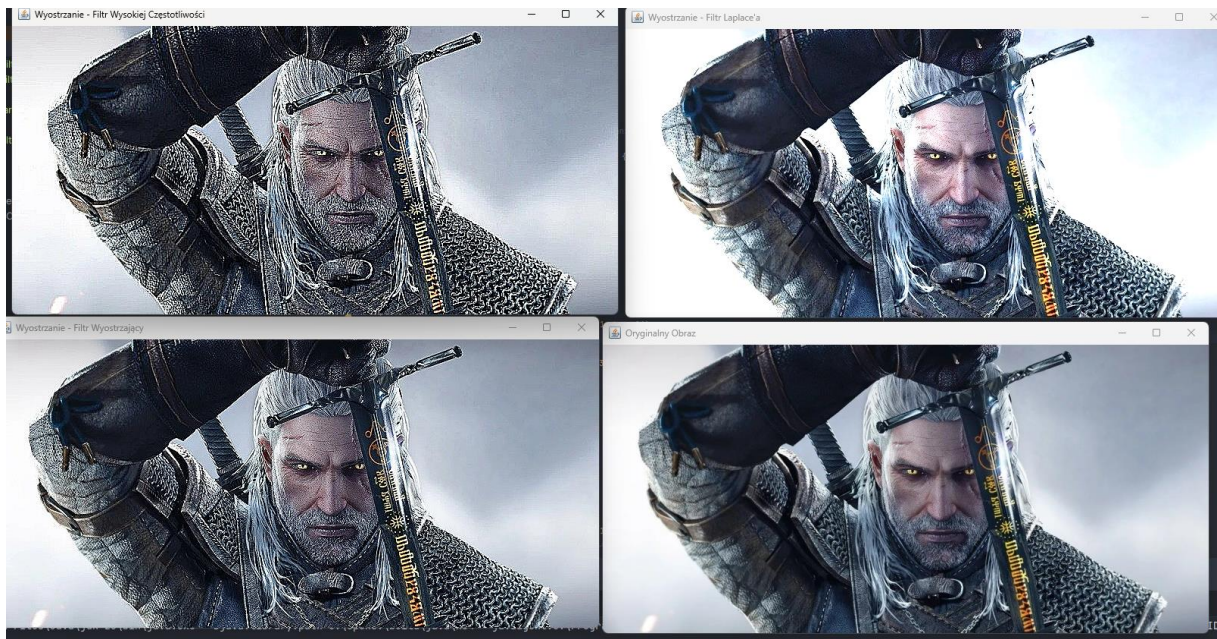
```
1 > import ...
2
3 public class CustomFilter { new *
4
5     public static void processImage() { 2 usages new *
6
7         System.loadLibrary(Core.NATIVE_LIBRARY_NAME);
8         String imagePath = "geralt.jpg";
9         Mat image = Imgcodecs.imread(imagePath);
10
11         if (image.empty()) {
12             System.out.println("Failed to load image!");
13             return;
14         }
15
16         // Tworzenie własnego jądra filtra (np. 3x3 filtr rozmycia)
17         Mat customKernel = new Mat( rows: 3, cols: 3, CvType.CV_32F) { new *
18         {
19             put( row: 0, col: 0, ...data: 1.0 / 9);
20             put( row: 0, col: 1, ...data: 1.0 / 9);
21             put( row: 0, col: 2, ...data: 1.0 / 9);
22             put( row: 1, col: 0, ...data: 1.0 / 9);
23             put( row: 1, col: 1, ...data: 1.0 / 9);
24             put( row: 1, col: 2, ...data: 1.0 / 9);
25             put( row: 2, col: 0, ...data: 1.0 / 9);
26             put( row: 2, col: 1, ...data: 1.0 / 9);
27             put( row: 2, col: 2, ...data: 1.0 / 9);
28         }
29     };
30
31     // Stworzenie macierzy dla obrazu po przefiltrowaniu
32     Mat customFilteredImage = new Mat();
33     // Zastosowanie własnego filtra na obrazie
34     Imgproc.filter2D(image, customFilteredImage, ddepth: -1, customKernel);
35     // Wyświetlenie oryginalnego obrazu oraz obrazu po zastosowaniu filtra
36     HighGui.imshow( winname: "Oryginalny Obraz", image);
37     HighGui.imshow( winname: "Obraz po zastosowaniu własnego filtra", customFilteredImage);
38     HighGui.waitKey( delay: 0);
39     System.exit( status: 0);
40 }
41
42 public static void main(String[] args) { new *
43     processImage();
44 }
45
46
```





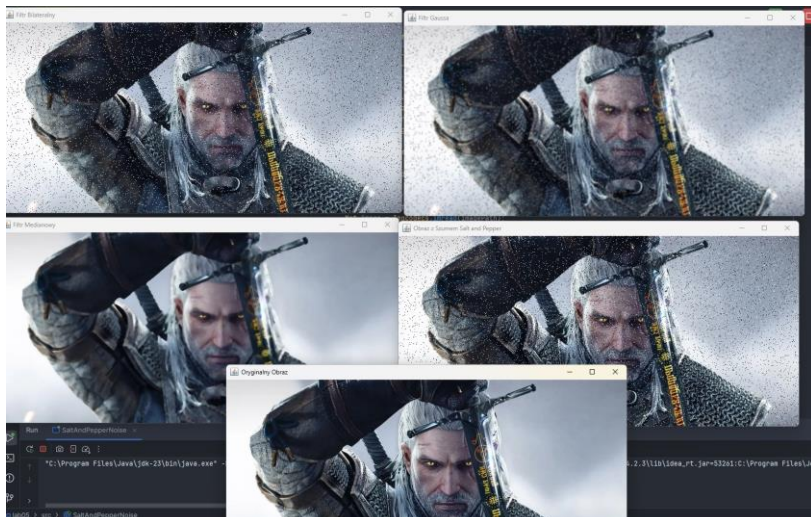
## Zad 5

```
1 > import ...
2 public class ImageSharpening { new *
3     public static void main(String[] args) { new *
4         System.loadLibrary(Core.NATIVE_LIBRARY_NAME);
5         String imagePath = "geralt.jpg";
6         Mat image = Imgcodecs.imread(imagePath);
7         if (image.empty()) {
8             System.out.println("Failed to load image!");
9             return;
10        }
11        // Metoda 1: Filtr wyostrzający (proste jądro wyostrzania)
12        Mat sharpenedImage1 = new Mat();
13        Mat kernel1 = new Mat( rows: 3, cols: 3, CvType.CV_32F) { new *
14            {
15                put( row: 0, col: 0, _data: 0);
16                put( row: 0, col: 1, _data: -1);
17                put( row: 0, col: 2, _data: 0);
18                put( row: 1, col: 0, _data: -1);
19                put( row: 1, col: 1, _data: 5);
20                put( row: 1, col: 2, _data: -1);
21                put( row: 2, col: 0, _data: 0);
22                put( row: 2, col: 1, _data: -1);
23                put( row: 2, col: 2, _data: 0);
24            }
25        };
26        Imgproc.filter2D(image, sharpenedImage1, ddepth: -1, kernel1);
27        // Metoda 2: Filtr wysokiej częstotliwości
28        Mat sharpenedImage2 = new Mat();
29        Mat kernel2 = new Mat( rows: 3, cols: 3, CvType.CV_32F) { new *
30            {
31                put( row: 0, col: 0, _data: -1);
32                put( row: 0, col: 1, _data: -1);
33                put( row: 0, col: 2, _data: -1);
34                put( row: 1, col: 0, _data: -1);
35                put( row: 1, col: 1, _data: 9);
36                put( row: 1, col: 2, _data: -1);
37                put( row: 2, col: 0, _data: -1);
38                put( row: 2, col: 1, _data: -1);
39                put( row: 2, col: 2, _data: -1);
40            }
41        };
42        Imgproc.filter2D(image, sharpenedImage2, ddepth: -1, kernel2);
43        // Metoda 3: Wyostrzanie z wykorzystaniem filtru Laplace'a
44        Mat laplacianImage = new Mat();
45        Imgproc.Laplacian(image, laplacianImage, CvType.CV_8U);
46        Mat sharpenedImage3 = new Mat();
47        Core.addWeighted(image, alpha: 1.5, laplacianImage, beta: -0.5, gamma: 0, sharpenedImage3);
48        HighGui.imshow( winname: "Oryginalny Obraz", image); // Wyświetlenie oryginalnego obrazu i obrazów po wyostrzeniu
49        HighGui.imshow( winname: "Wyostrzanie - Filtr Wyostrzający", sharpenedImage1);
50        HighGui.imshow( winname: "Wyostrzanie - Filtr Wysokiej Częstotliwości", sharpenedImage2);
51        HighGui.imshow( winname: "Wyostrzanie - Filtr Laplace'a", sharpenedImage3);
52        HighGui.waitKey( delay: 0);
53        System.exit( status: 0);
54    }
```



## Zad 6

```
1 > import ...
10 public class SaltAndPepperNoise { new *
11     public static void main(String[] args) { new *
12         System.loadLibrary(Core.NATIVE_LIBRARY_NAME);
13         String imagePath = "geralt.jpg";
14         Mat image = Imgcodecs.imread(imagePath);
15         if (image.empty()) {
16             System.out.println("Failed to load image!");
17             return;
18         }
19         // Dodajemy szum typu "salt and pepper" do obrazu
20         Mat noisyImage = addSaltAndPepperNoise(image, noiseRatio: 0.05);
21         // Stosujemy filtr Gaussa
22         Mat gaussianFiltered = new Mat();
23         Imgproc.GaussianBlur(noisyImage, gaussianFiltered, new Size(width: 5, height: 5), sigmaX: 0);
24         // Stosujemy filtr medianowy
25         Mat medianFiltered = new Mat();
26         Imgproc.medianBlur(noisyImage, medianFiltered, ksize: 5);
27         // Stosujemy filtr bilateralny
28         Mat bilateralFiltered = new Mat();
29         Imgproc.bilateralFilter(noisyImage, bilateralFiltered, d: 9, sigmaColor: 75, sigmaSpace: 75);
30         // Wyświetlenie wyników
31         HighGui.imshow(winname: "Oryginalny Obraz", image);
32         HighGui.imshow(winname: "Obraz z Szumem Salt and Pepper", noisyImage);
33         HighGui.imshow(winname: "Filtr Gaussa", gaussianFiltered);
34         HighGui.imshow(winname: "Filtr Medianowy", medianFiltered);
35         HighGui.imshow(winname: "Filtr Bilateralny", bilateralFiltered);
36
37         HighGui.waitKey(delay: 0);
38         System.exit(status: 0);
39     }
40     // Metoda dodająca szum "salt and pepper" do obrazu
41     public static Mat addSaltAndPepperNoise(Mat image, double noiseRatio) { usage new *
42         Mat noisyImage = image.clone();
43         Random rand = new Random();
44         int totalPixels = (int) (noiseRatio * image.rows() * image.cols());
45         for (int i = 0; i < totalPixels; i++) {
46             int row = rand.nextInt(image.rows());
47             int col = rand.nextInt(image.cols());
48             // Ustawiamy biały lub czarny piksel (szum)
49             double[] color = rand.nextBoolean() ? new double[]{255, 255, 255} : new double[]{0, 0, 0};
50             noisyImage.put(row, col, color);
51         }
52         return noisyImage;
53     }
54 }
```



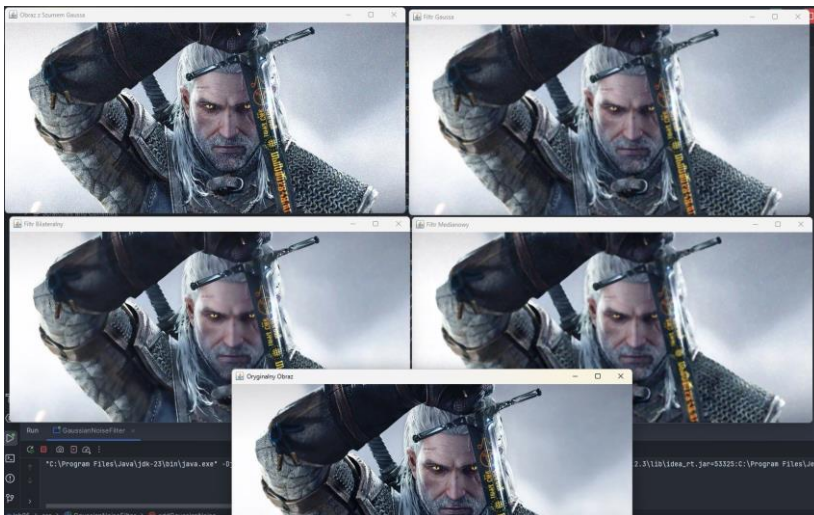
Filtr Medianowy zazwyczaj jest najlepszy do usuwania szumu typu "salt and pepper", ponieważ skutecznie usuwa punkty białe i czarne, nie rozmywając przy tym krawędzi.

## Zad 7

```

1  > import ...
9  public class GaussianNoiseFilter { new *
10 public static void main(String[] args) { new *
11     System.loadLibrary(Core.NATIVE_LIBRARY_NAME);
12     String imagePath = "geralt.jpg";
13     Mat image = Imgcodecs.imread(imagePath);
14     if (image.empty()) {
15         System.out.println("Failed to load image!");
16         return;
17     }
18     // Dodajemy szum Gaussa do obrazu
19     Mat noisyImage = addGaussianNoise(image, stdDev: 20);
20     // Stosujemy filtr Gaussa
21     Mat gaussianFiltered = new Mat();
22     Imgproc.GaussianBlur(noisyImage, gaussianFiltered, new Size( width: 5, height: 5), sigmaX: 0);
23     // Stosujemy filtr medianowy
24     Mat medianFiltered = new Mat();
25     Imgproc.medianBlur(noisyImage, medianFiltered, ksize: 5);
26     // Stosujemy filtr bilateralny
27     Mat bilateralFiltered = new Mat();
28     Imgproc.bilateralFilter(noisyImage, bilateralFiltered, d: 9, sigmaColor: 75, sigmaSpace: 75);
29     // Wyświetlenie wyników
30     HighGui.imshow( winname: "Oryginalny Obraz", image);
31     HighGui.imshow( winname: "Obraz z Szumem Gaussa", noisyImage);
32     HighGui.imshow( winname: "Filtr Gaussa", gaussianFiltered);
33     HighGui.imshow( winname: "Filtr Medianowy", medianFiltered);
34     HighGui.imshow( winname: "Filtr Bilateralny", bilateralFiltered);
35     HighGui.waitKey( delay: 0);
36     System.exit( status: 0);
37 }
38 // Metoda dodająca szum Gaussa do obrazu
39 @ public static Mat addGaussianNoise(Mat image, double stdDev) { 1 usage new *
40     Mat noisyImage = image.clone();
41     Mat gaussianNoise = new Mat(image.size(), image.type());
42     Random rand = new Random();
43     // Generowanie szumu Gaussa
44     Core.randn(gaussianNoise, mean: 0, stdDev);
45     // Dodanie szumu do oryginalnego obrazu
46     Core.add(noisyImage, gaussianNoise, noisyImage);
47     return noisyImage;
48 }
49 }

```



## Wnioski:

Każdy z użytych filtrów ma swoje specyficzne zastosowanie, filtr Gaussa nadaje się do wygładzania i usuwania szumu Gaussa, ale rozmywa krawędzie. Filtr medianowy jest idealny do eliminacji szumu „salt and pepper” zachowując krawędzie. Filtr bilateralny to najbardziej wszechstronny wybór, dobrze usuwający różne typy szumu przy zachowaniu szczegółów krawędzi. Filtry własne można dostosować do specyficznych zadań, ale wymagają dokładnej konfiguracji i dopasowania. Zadania wymagały praktycznego zastosowania filtrów, przez co w atrakcyjny sposób pokazały różnice w ich działaniu.