CMP717 - Image Processing

Edge Preserving Filters

Serkan UYSAL
N23236714
April 9, 2024

Contents

List of Code Listings

1 Introduction

From the past to the present, camera quality has continuously improved. Older cameras often produced noisy photographs, but scientists developed filters to remove this noise.

1.1 Noises

Image noise is the random variation of brightness or color in electronic images, often caused by a scanner or digital camera's image sensor and circuitry.

1.1.1 Uniform Noise

Uniform noise is an image noise characterized by its uniform distribution, which exhibits constant intensity variations within a specified range across the image.

1.1.2 Gaussian Noise

Gaussian noise follows a normal distribution (*Gaussian*) and appears as random variations in pixel intensity.

1.2 Filters

Filters are used to modify photos in order to achieve a smoothing or sharpening effect on the image.

1.2.1 Median Filter

The median filter is a non-linear digital filtering technique that replaces each pixel's value with the median value of its neighboring pixels, effectively reducing noise while preserving edges in an image.

1.2.2 Kuwahara Filter

The Kuwahara filter is more complex than the median filter. It is a type of filter used for smoothing an image, and it provides edge preservation by utilizing the statistical properties of the surrounding region for each pixel.

2 Tools

The experiments were carried out in Python 3.11.
The functions of the OpenCV library were used to read the images, so the output was sorted as H, W, and C and converted to a grayscale using the OpenCV function. These values were distributed between 0 and 255.
Functions were prepared using the NumPy library to obtain Uniform and Gaussian Noises.
The albumentation library, chosen for its robust Mean Filter, was used. This library offers a wide range of augmentation functions, making it a versatile tool for image processing.
We used the pykuwahara library for the Kuwahara filter, which is interoperable with OpenCV.
Below is a code snippet on setting up the environment; use this link to access the code snippets and more.

```
pip install -r \
https://raw.githubusercontent.com/uysalserkan/MSComp-Season-I/main/Image-Process/HW1/requirements.txt
```

Listing 1: Bash: Install Requirements

3 Results

The original photos are given below, as well as three medical images and three Mahmut Tuncer, who is the best comedian and signer in Turkey.

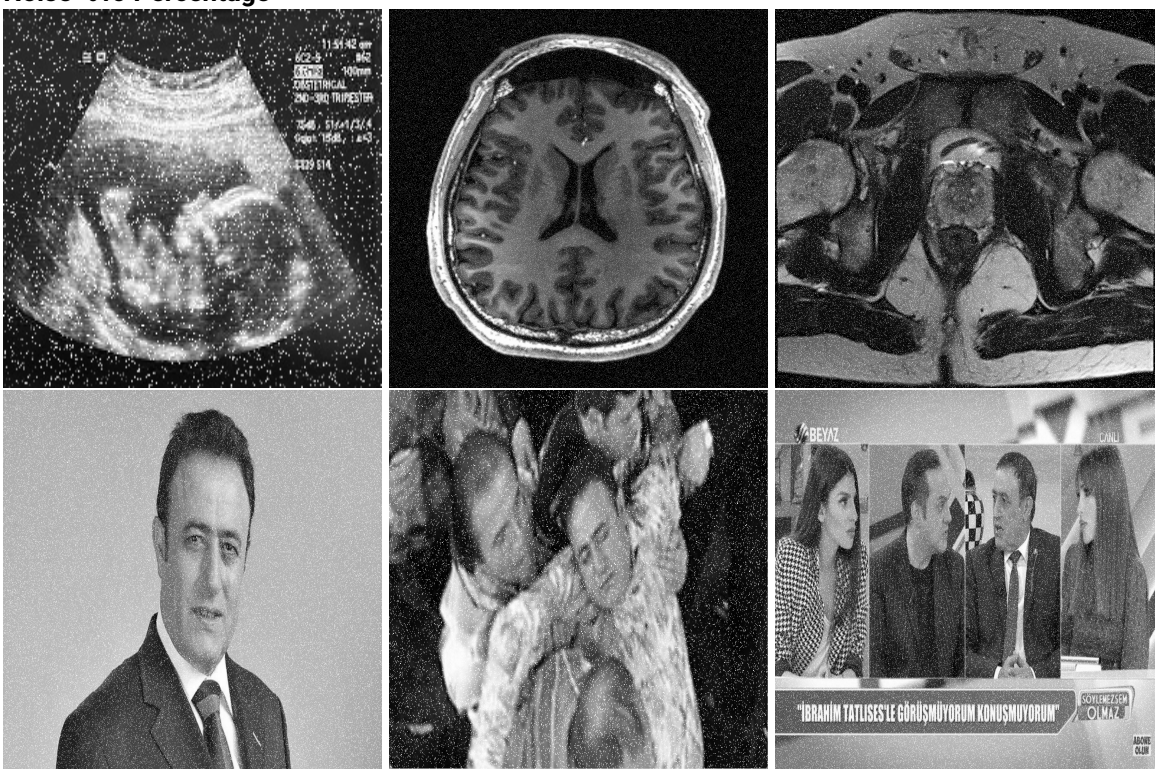
3.1 Noises

3.1.1 Uniform Noise

```python
import numpy as np


def uniform_noise(img: np.ndarray, low: int = 0, high: int = 255, perc: float = 0.1) -> np.ndarray:
    """
    Add uniform noise to the image.
    :param img: Image as numpy array.
    :param low: Uniform noise values low.
    :param high: Uniform noise values high.
    :param perc: Noise apply percentage.
    :return: blurred image as numpy array
    """
    noise = np.random.uniform(low=low, high=high, size=img.shape[:2])

    noised_pixels = np.random.rand(*img.shape[:2]) < perc
    mask = np.where(noised_pixels, noise, 0)

    blurred_img = img + mask

    return blurred_img
```

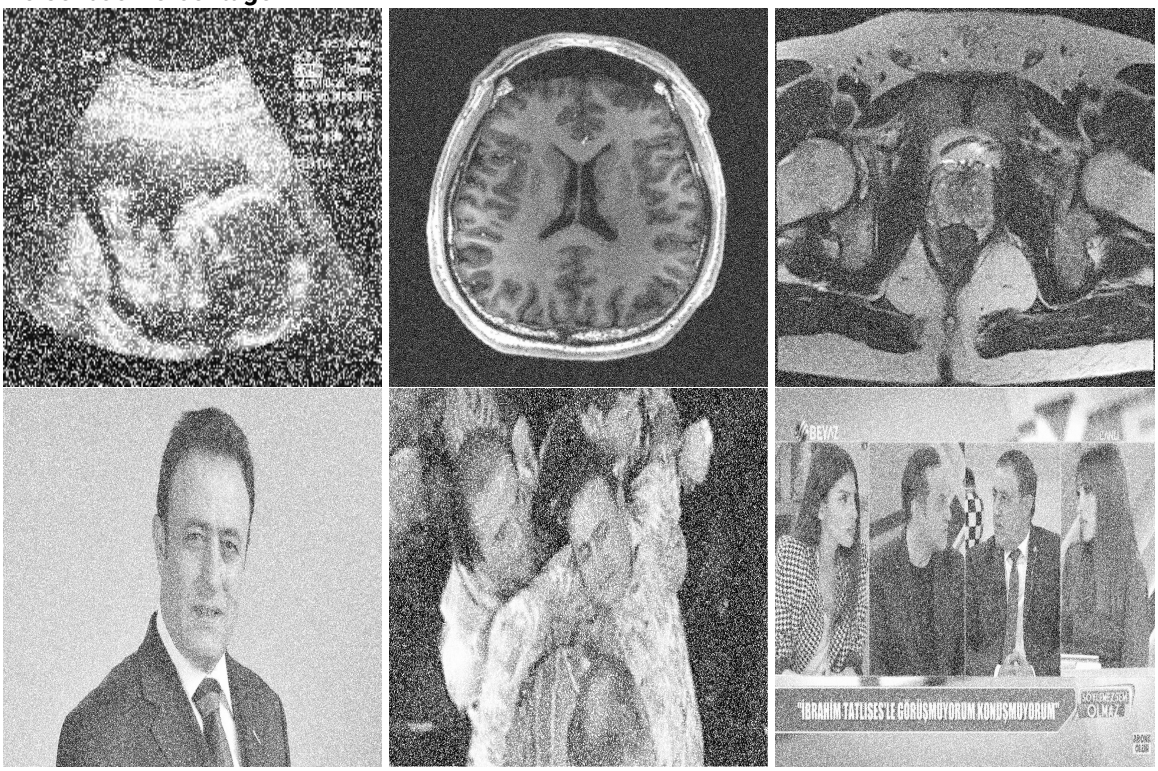
Listing 2: Python: Uniform Noise

1. **Noise %10 Percentage**



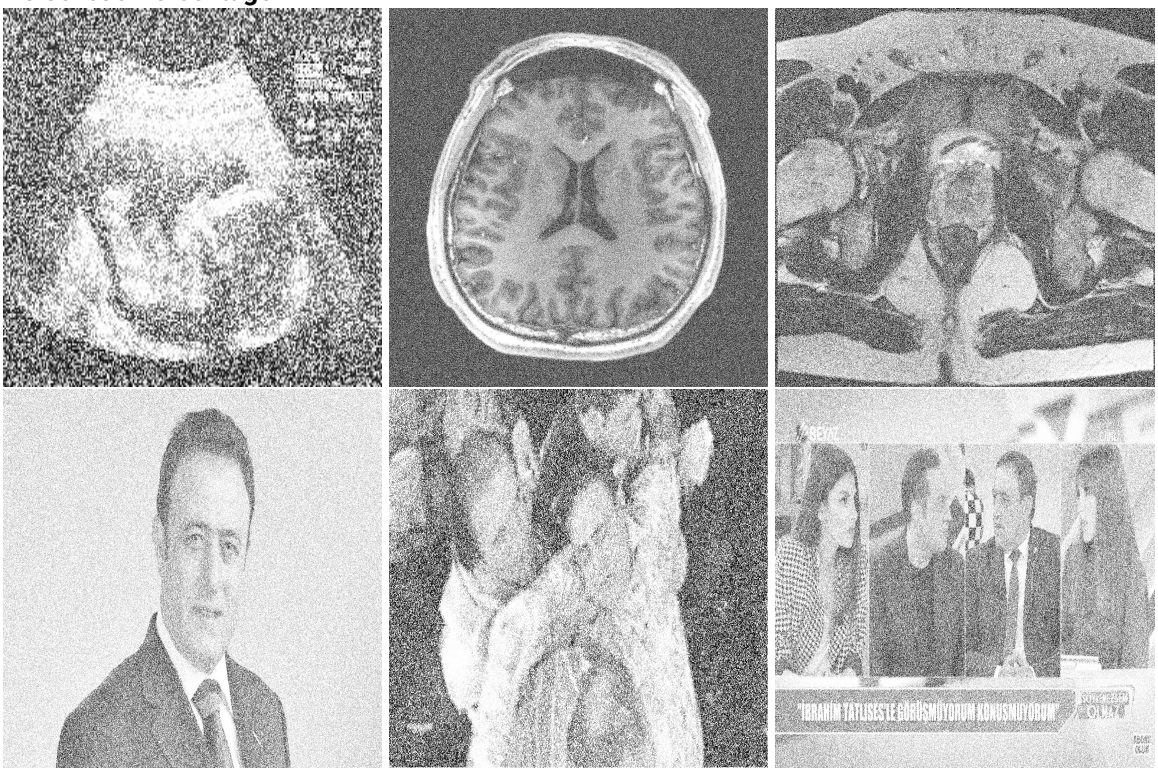
With %10 uniform noise, low and high-dimension photos are understood very well.

2. **Noise %50 Percentage**



Low-dimension photos are hard to understand with %50 uniform noise, and other images are well understood.

3. **Noise %80 Percentage**



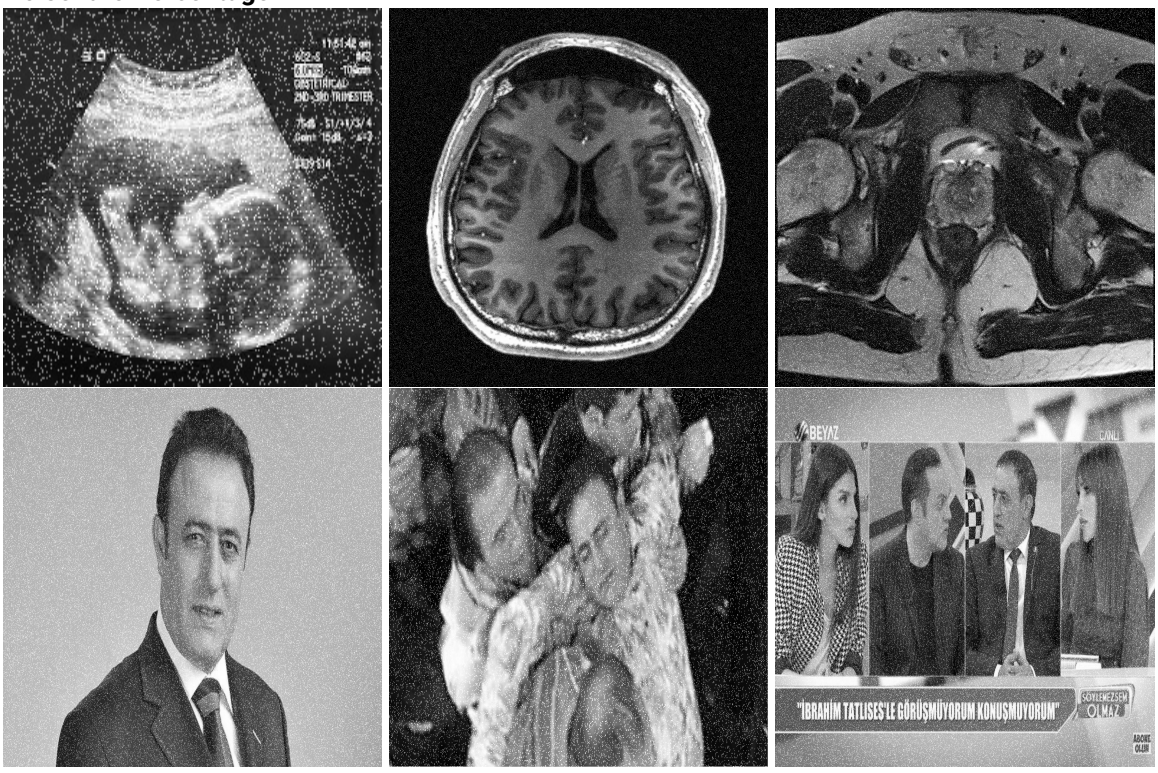
With %80 uniform noise, all low and high-dimension photos are hardly understood due to the high noise value.

3.1.2 Gaussian Noise

```python
import numpy as np


def gaussian_noise(img: np.ndarray, variance: int = 10, mean: int = 128, perc: float = 0.1) -> np.ndarr
    """
    Add gaussian noise to the image
    :param img: Image as numpy array.
    :param variance: Gaussian noise values variance.
    :param mean: Gaussian noise values mean.
    :param perc: Noise apply percentage.
    :return: blurred image as numpy array.
    """
    gaussian_noise_np = np.random.normal(mean, variance, img.shape[:2])
    noised_pixels = np.random.rand(*img.shape[:2]) < perc

    mask = np.where(noised_pixels, gaussian_noise_np, 0)
    blurred_img = img + mask

    return blurred_img
```

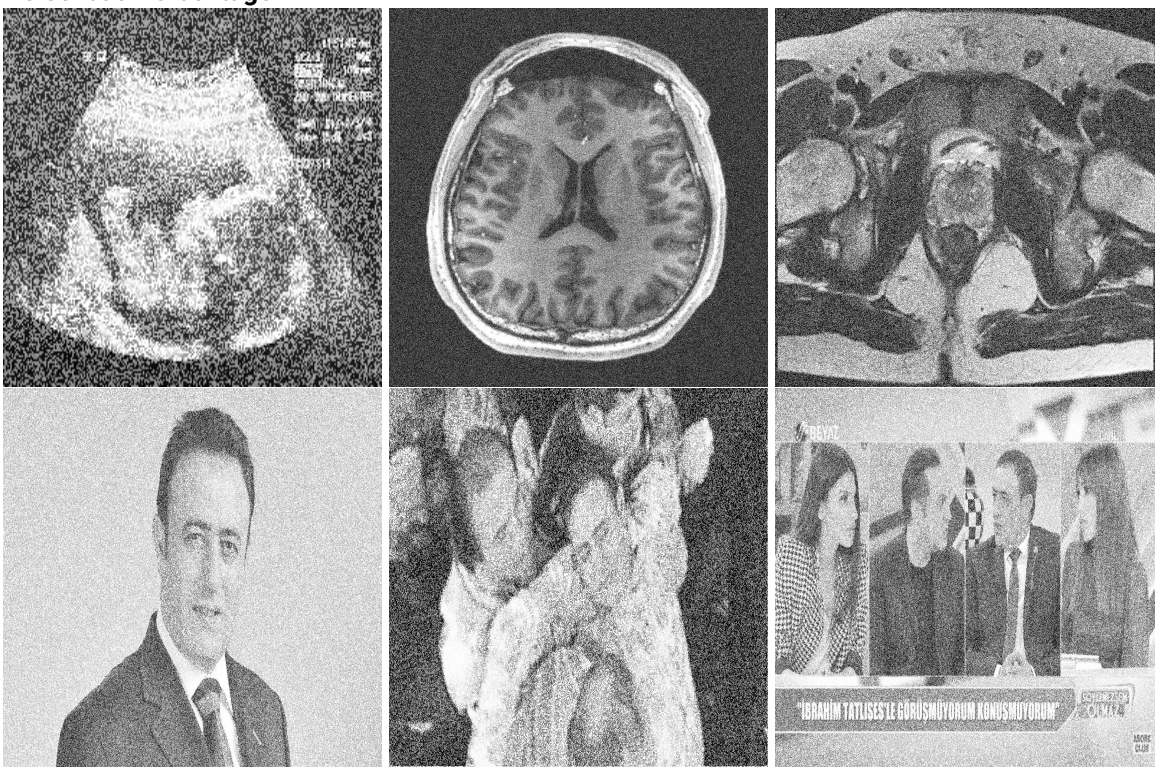
Listing 3: Python: Gaussian Noise

1. **Noise %10 Percentage**



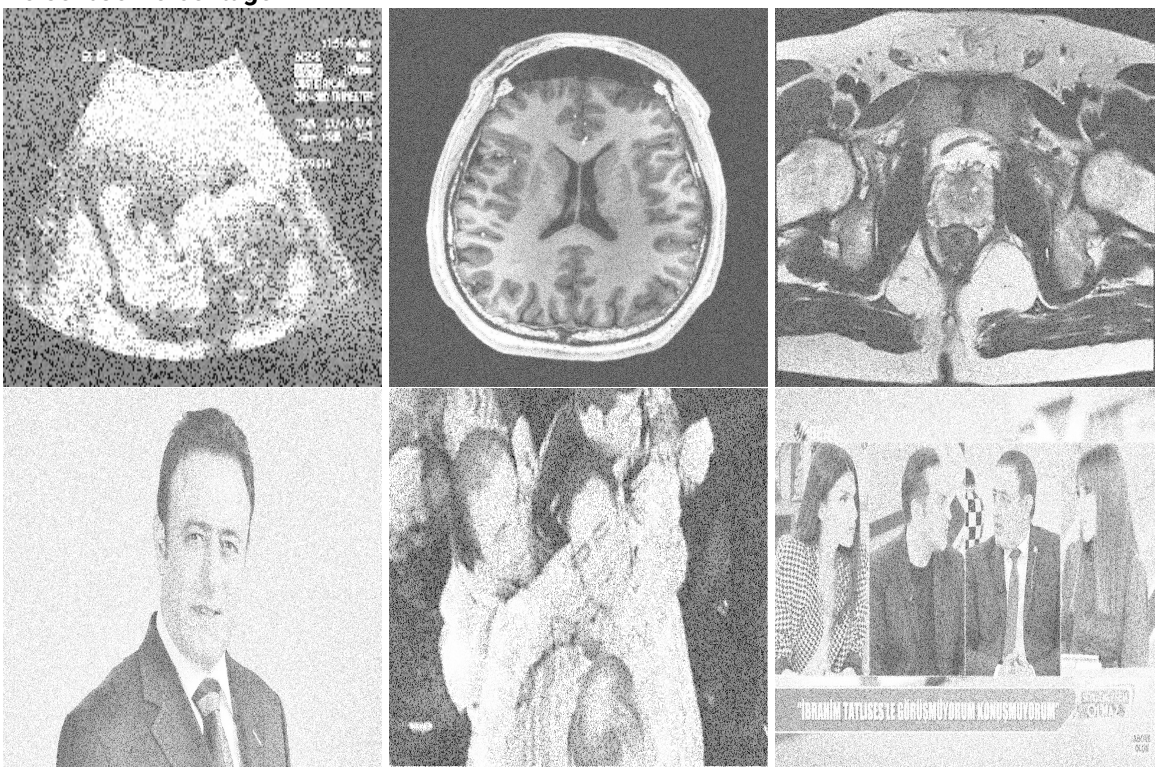
With %10 Gaussian noise, low and high-dimension images are understood very well.

2. **Noise %50 Percentage**



With %50 Gaussian noise, low-dimension images are hardly understood, and high-dimension images are again understood very well.

3. **Noise %80 Percentage**



With %80 Gaussian noise, low and high-dimension images are hardly understood.

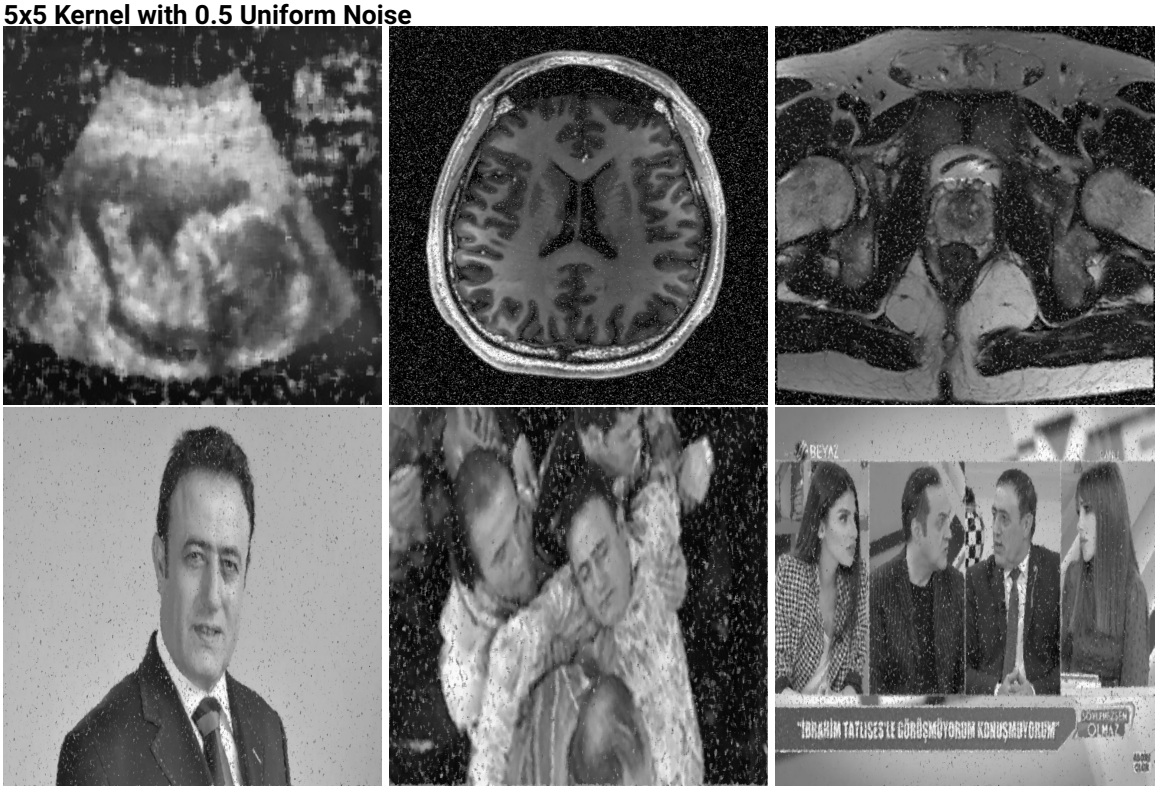
3.2 Filters

3.2.1 Median Filter

```python
def median_ep(img: np.ndarray, kernel_size: int) -> np.ndarray:
    """
    Median edge preserving filter with albumentations library.
    :param img: Image as numpy array.
    :param kernel_size: Kernel size for median filter, 5 or 15.
    :return: Median filtered image as numpy array.
    """
    ep_img = A.augmentations.median_blur(img.astype(np.uint8), kernel_size)

    return ep_img
```

Listing 4: Python: Median Filter

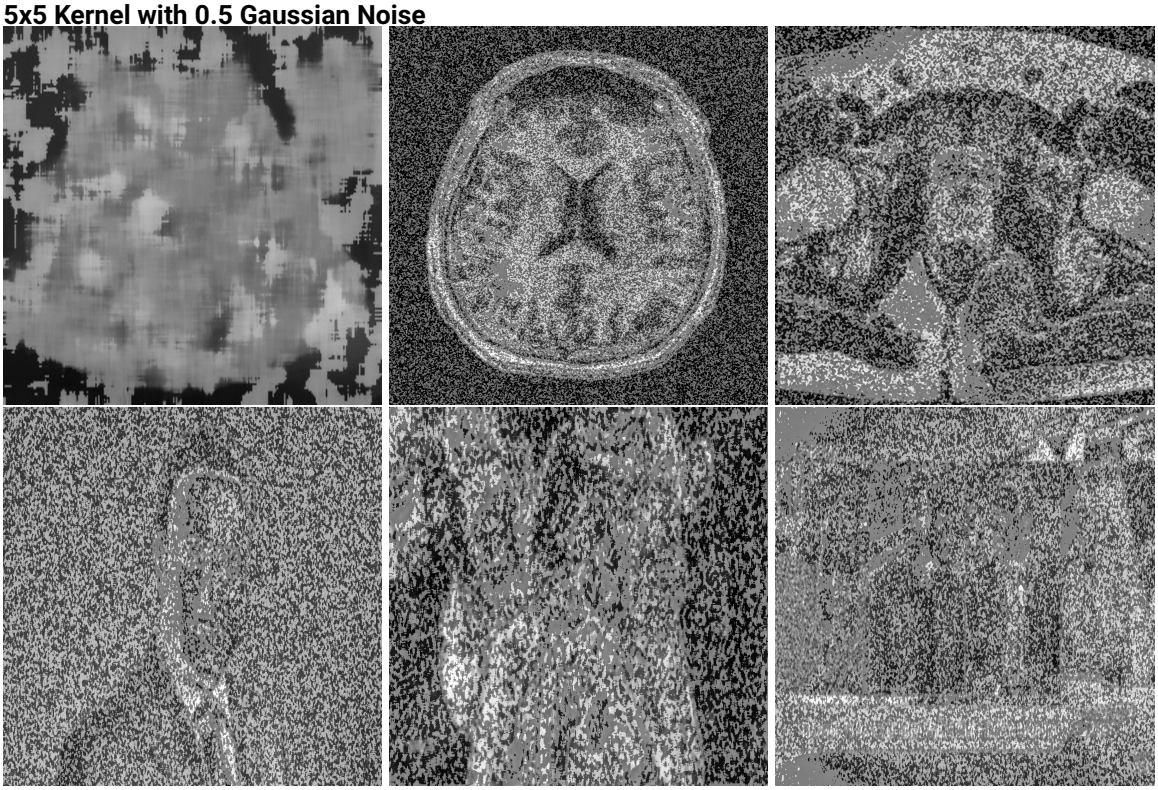
1. **5x5 Kernel with 0.5 Uniform Noise**



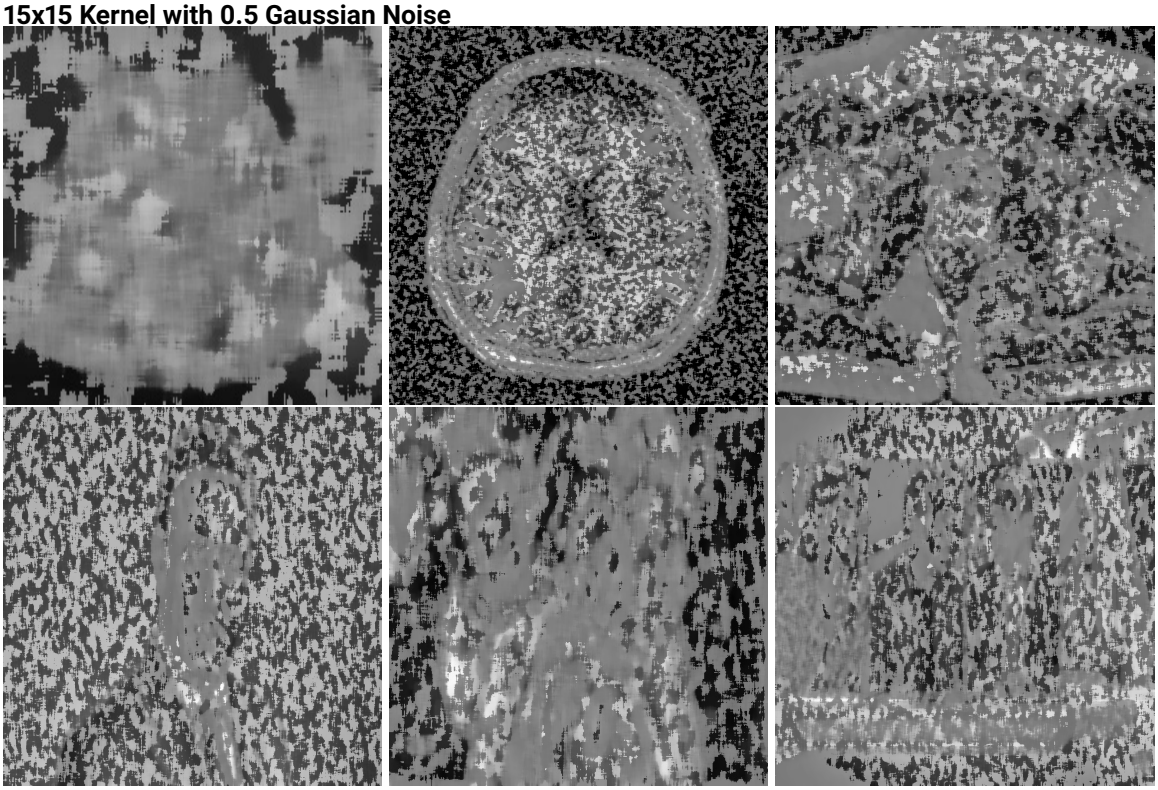
2. **15x15 Kernel with 0.5 Uniform Noise**

Using 5x5 and 15x15 Median filters on uniform noise with %50 percent of photos are above. As can be seen, applied 5x5 filter photos have more detail and visible noise. On the other hand, applied 15x15 filter photos are blurry but have less noise than others.

1. **5x5 Kernel with 0.5 Gaussian Noise**



2. **15x15 Kernel with 0.5 Gaussian Noise**



All the photos are terrible.

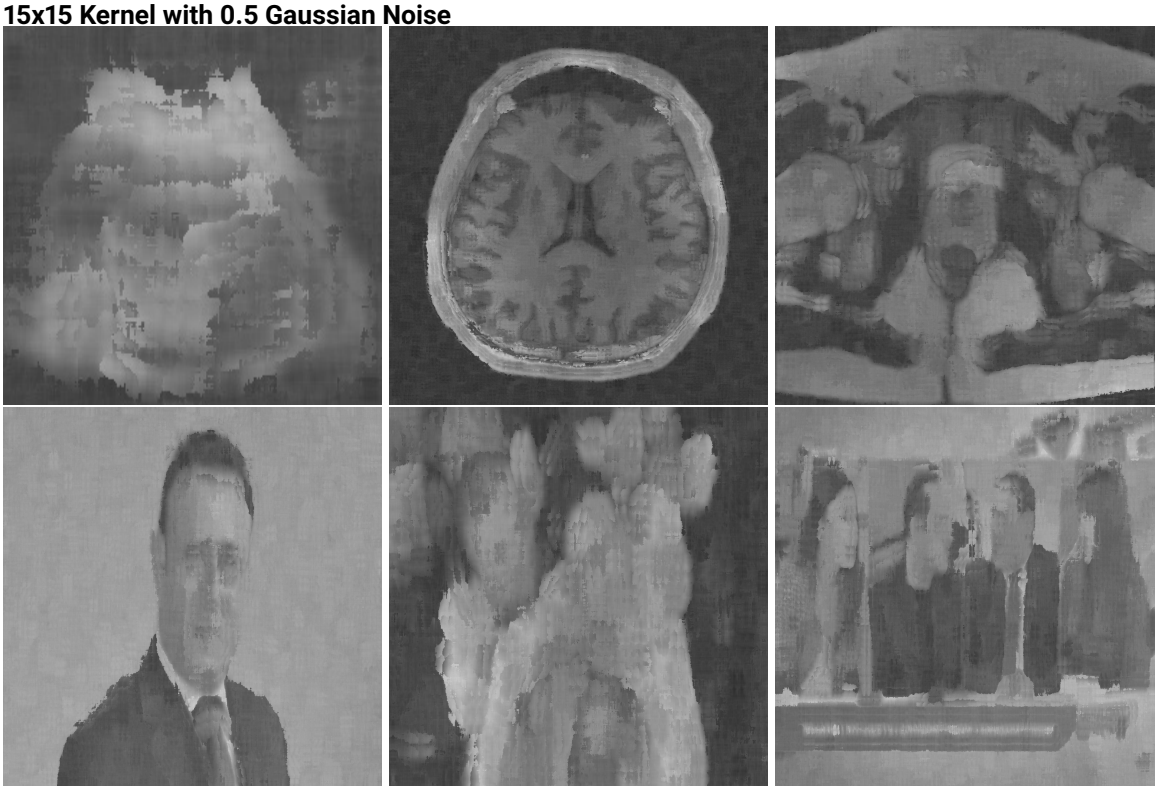
3.2.2 Kuwahara Filter

```python
from pykuwahara import kuwahara


def kuwahara_ep(img: np.ndarray, kernel_size: int, method: str) -> np.ndarray:
    """
    Kuwahara edge preserving filter with pykuwahara library.
    :param img: Image as numpy array.
    :param kernel_size: Kernel size for Kuwahara filter, 5 or 15.
    :param method: Method for Kuwahara filter, `mean` or `gaussian`.
    :return: Kuwahara filtered image as numpy array.
    """
    ep_img = kuwahara(img.astype(np.uint8), method=method, radius=kernel_size)

    return ep_img
```

Listing 5: Python: Kuwahara Filter

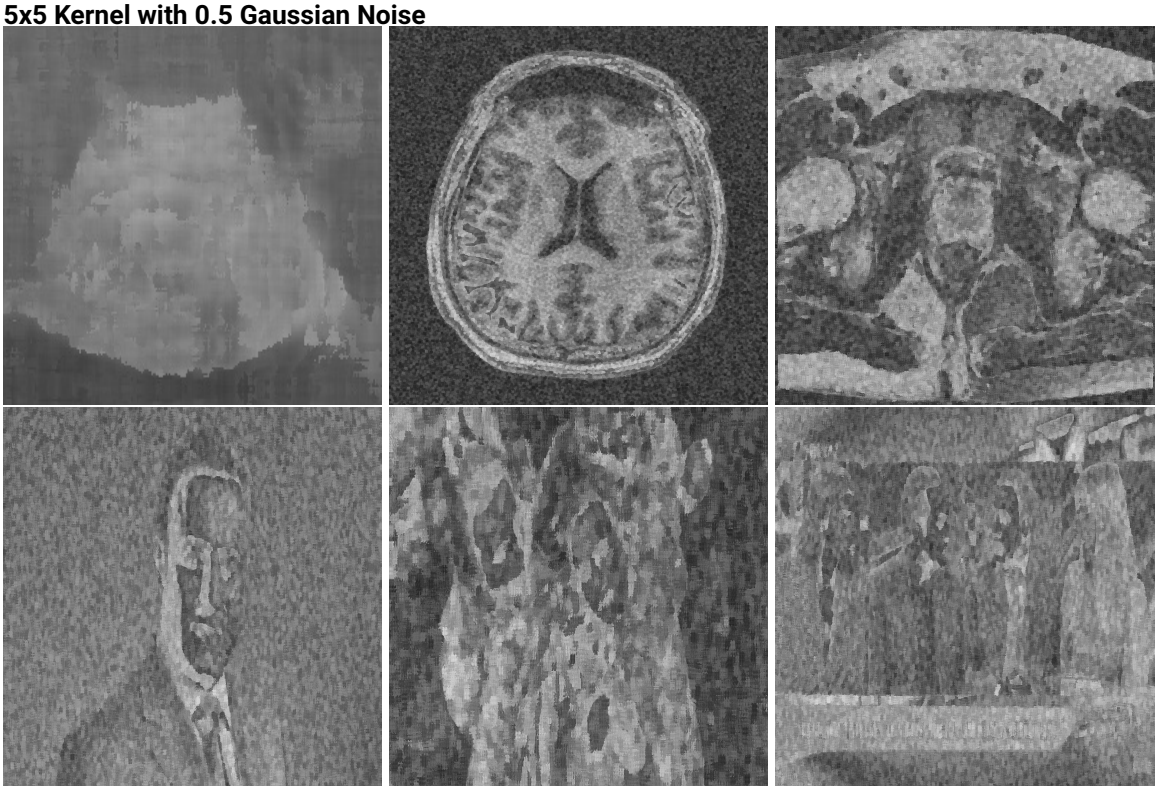
1. **5x5 Kernel with 0.5 Uniform Noise**

2. **15x15 Kernel with 0.5 Gaussian Noise**

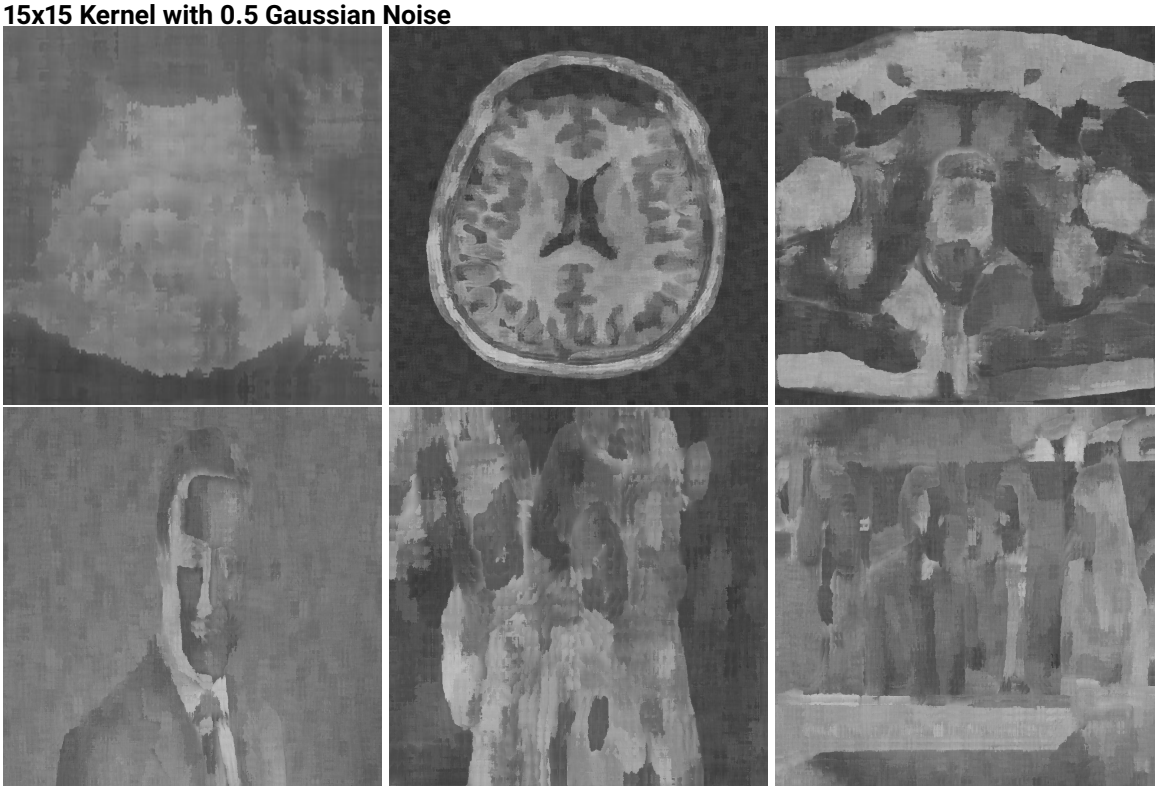


The main concepts preserved around five photos with 5x5 and 15x15 filters applied photos. Just one photo has terrible quality.

1. **5x5 Kernel with 0.5 Gaussian Noise**



2. **15x15 Kernel with 0.5 Gaussian Noise**



The 5x5 filter photos with more detail were applied than the 15x15 filter photos; otherwise, the 15x15 filter photos had primary information, which was not too bad compared to the Median and Gaussian.

4 Discussion

1. The main difference between Median and Kuwahara filters is 0.5 Uniform noise result. Median filter results have more details and are in better shape than the Kuwahara filter.
2. The Kuwahara filter produces better results than the median filter due to its algorithm.
3. Small filter sizes can preserve the shapes of the images, although after applying the filter, noises could be seen.

The GitHub repository provides options to apply various noise levels and filters to photos.

References

[1] https://medium.com/swlh/what-is-a-kuwahara-filter-77921ce286f2

[2] https://en.wikipedia.org/wiki/Bilateral_filter

[3] https://pytorch.org/docs/stable/generated/torch.clamp.html

[4] https://en.wikipedia.org/wiki/Median_filter

[5] https://en.wikipedia.org/wiki/Guided_filter

[6] https://en.wikipedia.org/wiki/Anisotropic_diffusion

[7] https://en.wikipedia.org/wiki/Edge-preserving_smoothing

[8] https://docs.opencv.org/4.x/d4/d13/tutorial_py_filtering.html

[9] https://stackoverflow.com/questions/67270514/why-does-cv2-medianblur-returns-a-format-err

[10] https://github.com/OzgurBagci/fastbilateral/tree/master

[11] https://web.cs.hacettepe.edu.tr/~erkut/bil717.s12/w09-bilateral-nlmeans.pdf

[12] https://dergipark.org.tr/en/download/article-file/384688

[13] https://github.com/yoch/pykuwahara

[14] https://en.wikipedia.org/wiki/Image_noise