Tutorial: Image restoration: denoising

Note

This tutorial aims to study some random noises and to test different image restoration methods (image denoising).

The different processes will be applied on the following MR image.

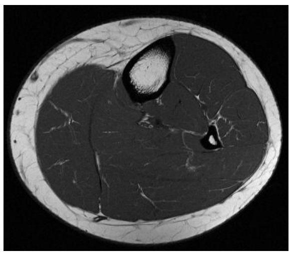

Figure 1: Leg.

1 Generation of random noises

Some random noises are defined with the given functions (corresponding to specific distributions):

- uniform noise: $R = a + (b - a) * U(0, 1)$. Use the matlab function *rand*.
- Gaussian noise: $R = a + b * N(0, 1)$. Use the matlab function *randn*
- salt and pepper noise: $R : \begin{cases} 0 \leq U(0,1) \leq a & \mapsto & 0 \\ a < U(0,1) \leq b & \mapsto & 0.5 \\ b < U(0,1) \leq 1 & \mapsto & 1 \end{cases}$
- exponential noise : $R = -\frac{1}{a} * \ln(1 - U(0, 1))$

?

Generate sample images with the four kinds of random noise and visualize their histograms with the matlab function imhist. Pay attention to the intensity range of the resulting images when calculating the histograms: see the help of the matlab function imadjust.

2 Noise estimation

The objective is to evaluate the characteristics of the noise in a damaged/noisy image (in a synthetic manner in this tutorial).

?

1. Visualize the histogram of a Region Of Interest (ROI) of the image of Fig.1. The ROI should be extracted from a uniform (intensity) region. Use the matlab function roipoly.
2. Add an exponential noise to the original image and visualize the histogram of the selected ROI.
3. Add a Gaussian noise (with the help of the matlab function imnoise) to the original image and visualize the histogram of the selected ROI.

3 Image restoration by spatial filtering

?

1. Add a salt-and-pepper noise to the image of Fig.1. Use the matlab function imnoise.
2. Test the image filters 'min', 'max', 'mean' and 'median' with the help of the matlab functions imfilter , ordfilt2 and medfilt2.

The median filter is efficient in the case of salt-and-pepper noise. However, it replaces every pixel value (first problem) by the median value determined at a given scale (second problem). In order to avoid these two problems, Gonzalez and Woods proposed an algorithm [1].

?

Implement the adaptive median filter from the following algorithm (Alg.1) based on two steps (the operator acts on an operational window of size $k \times k$ where different statistics are calculated: median, min or max).

References

[1] Rafael C. Gonzalez and Richard E. Woods. *Digital Image Processing (2nd Edition)*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 2002.

```
Data: Input (noisy) image $I$
Data: Maximal scale $S_{max}$
Result: Filtered image $F$

Main Function amf($I$, $S_{max}$):
    forall $pixels$ $(i, j)$ do
        $S \leftarrow 1$
        $med \leftarrow$ Med$(I, j, j, S)$
        while isMedImpulseNoise($med$, $i$, $j$, $S$) $AND$ $S \leq S_{max}$ do
            $S \leftarrow S + 1$
            $med \leftarrow$ Med$(I, j, j, S)$
        end

        if $I(i, j)$ =Min($I$, $i$, $j$, $S$) OR $I(i, j)$ =Max($I$, $i$, $j$, $S$) OR $S = S_{max}$ then
            $F(i, j) \leftarrow$ Med($I$, $i$, $j$, $S$)
        else
            $F(i, j) \leftarrow I(i, j)$
        end
    end

Function isMedImpulseNoise($f$, $i$, $j$, $S$):
    if $f$=$min$ $OR$ $f$=$max$ then
        return True
    else
        return False
    end
```

Algorithm 1: Adaptive median filter. The main function takes two arguments: the image I to be filtered and the maximal size of neighborhood S_{max}. For each pixel (i, j), the good scale is the scale where the median value is different from the min and the max (median value is thus not an impulse noise). Then, if the pixel value is an impulse noise or the maximal scale has been reached, it must be filtered. Otherwise, it is kept untouched.