Since the approach I took to make these videos is computationally heavy, I chose to default the function to only render 5 frames for each video. To analyse the results, I played the videos in Windows Movie Maker in loopback mode.

RuiCampos_AvgFilterFFT.avi

This video is a demonstration that an average filter applied in the spatial/image domain acts as a low pass filter in the frequency domain. Each frame results from applying the filter to the previous frame.

In the image, we see that details are lost. It becomes blurred.
In the frequency domain, we see that high frequencies get attenuated with each passage of the filter.

From this, I can conclude that high frequencies components are responsible for constructing the details in the image.

However, in the fourier domain, we can also see two lines passing through the origin. They include a wide range of frequencies. If we consult the image, we see that there is a well defined black border emerging as the filter is applied. To construct such a border, we would need those high frequency components.

The horizontal border is constructed by the vertical line in the frequency domain.
Similarly, the vertical border is constructed by the horizontal line in the frequency domain.

These borders occur because a black padding is added to the image when applying the average filter. The black pixels in the padding “pour” into the image.

RuiCampos_SP_noise.avi

In this video, we see how the function imnoise contaminates the image with salt & pepper noise. Given the following usage

Icontaminated = imnoise(IM,’salt % pepper’, r)

This function will assign to each pixel a pseudorandom number sampled uniformly from the range [0,1[. For a given pixel, if this value falls in the range [0,r/2[, the pixel value will be changed to 0. If it falls in [r/2, r[, it is changed to 255. Otherwise, it remains unchanged.

This means that r will be(approximately) the fraction of pixels affected, half of whom are set to 0, and the other half to 255. This can be seen in the video, as r grows, more and more pixels are taken from their original bin and placed in the first and last bins of the histogram.

RuiCampos_gauss_noise.avi

I tried the same approach here. The imnoise function creates a matrix with the size of the image. The elements of this matrix are drawn from a gaussian distribution. The matrix is then added to the image.

In the video, I repeatedly add this gaussian noise(mean=0, var=.02), resulting in an increased variance of the pixel intensity distribution and also some saturation of the pixels in the image. The saturation happens because the sample matrix is added to existing values in the image, and if the result of that operation exceeds 255 in value, it defaults to 255. Similarly, if the result is less than 0, it defaults to 0.