Noise reduction Average filter is given by the command \texttt{ones(N)/N^2}.

Images in this exercise have strong salt-and-pepper noise (random white and black dots). Average filter mainly spreads this noise over the image. Median filter works much better, effectively removing those random dots.

Histogram stretching For linear stretching of the image histogram to the full [0...255] range the following transform of image intensities should be applied:

\[ I' = 255 \cdot \frac{I - \min\{I\}}{\max\{I\} - \min\{I\}}. \]

Gradient You can compute \( x \) and \( y \) derivatives at each image point using the built-in MATLAB function: \([dx,dy] = \text{gradient}(I);\)

You can also compute the gradient explicitly – by applying linear filters. As simple examples, consider formulas:

- (1st order approximation of derivative) \( f(x + \Delta x) \approx f(x) + \frac{df}{dx} \cdot \Delta x \), implying that \( \frac{df}{dx} \approx f(x+1) - f(x) \), which is computed by convolving the function \( f \) with the filter \([0\ -1\ 1]\);

- (2nd order approximation of derivative) \( \frac{df}{dx} \approx \frac{1}{2}(f(x+1) - f(x-1)) \), which corresponds to the filter \( \frac{1}{2} \begin{bmatrix} -1 & 0 & 1 \end{bmatrix} \).

More complex, 2-dimensional filters are: Sobolev, Prewitt (see MATLAB function \texttt{fspecial}).

Take care to apply appropriate constant factors if you use these filters, otherwise you derivatives will be in a scale different from the image.

Gradient magnitude Magnitude of the gradient (the norm of the vector \([dx\ dy]\)) can be computed as follows: \texttt{magn = sqrt(dx.^2+dy.^2)}. Note that you do computations for all image points “at once”: \texttt{dx.^2} gives point-wise square of the matrix \( dx \), and \texttt{sqrt()} computes square root for all points of the input matrix. This is the dominating practice of programming in MATLAB: you try to do operations on a matrix as a whole.
MATrix LABoratory) is optimized for such usage, and matrix operations are computed very efficiently (unlike loops).

**Gradient orientation** Basically, if $\varphi$ is the angle of the gradient vector $[dx\ dy]$ (from the positive $x$-axis), then $\tan \varphi = \frac{dy}{dx}$. This can be computed as $\phi = \text{atan}(dy./dx)$ (here “./” is the point-wise division – don’t confuse it with the operation “/”, which solves linear equations). A more accurate way to compute this angle is to use $\text{atan2}(dy,dx)$ (note the order of the arguments in $\text{atan2}$).

$\text{atan2}$ returns values in the range $[-\pi..\pi]$. To move this range to the required $[0..2\pi]$ *without* changing the angles, you can use either of the following methods (or something else):

- $\phi = \text{mod}(\phi,2*\pi)$;
- $\phi = \phi + 2*\pi*(\phi<0)$;
- $\phi(\phi<0) = \phi(\phi<0) + 2*\pi$;

Note the use of *logical* operations: $\phi<0$ is the matrix of the same size as $\phi$, with 1’s where $\phi$ is negative and 0’s otherwise. See MATLAB help for *find*.