Smoothing and Edge Detection

Write a program that smooths an image and then detects and enhances the edges in the image.

1. Input Arguments:

   - **Grey-scale Image Filename**
     Read the input image of arbitrary format into a 1-channel matrix using `imread`.
   - **Filter Width**
     The width of the noise filtering filter kernel.
   - **Filter Type**
     The type of noise filtering: Gaussian smoothing, median filtering.
   - **Standard Deviation**
     If Gaussian smoothing is chosen, use this additional argument for the standard deviation used while creating the Gaussian filter. How is it related to the filter width?
   - **Hysteresis Thresholds**
     Optionally take two arguments for the hysteresis thresholds t_1 and t_h of the edge detector.

   If no input arguments are given or they don't match the expectation, your program should print a brief documentation of how to use it.

2. Algorithm: We mimic the famous *Canny edge detector* with pre-processing.

   - **Filtering**
     Apply either the Gaussian smoothing or median filtering on the image.
   - ***Sobel* Gradient**
     Compute a horizontal J_x and a vertical gradient image J_y using the filter masks

     $$\begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix} \quad \text{and} \quad \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}.$$

     Compute the edge strength image E_s as

     $$E_s(i,j) = \sqrt{J_x(i,j)^2 + J_y(i,j)^2}$$

     and the edge orientation image E_o as

     $$E_o(i,j) = \arctan(J_y(i,j), J_x(i,j))$$

   - ***Nonmax* Edge Enhancement**
     Find the best fitting orientation of the edge d_k in $d_1 \dots d_4$ which correspond to the angles $0°, 45°, 90°, 135°$ using $E_o(i,j)$. Then, enhance the edges by computing an improved image E_m as

     $$E_m(i,j) = \begin{cases} 0 & \text{if } E_s(i,j) \text{ is smaller than at least one of its two neighbors on } d_k, \\ E_s(i,j) & \text{else.} \end{cases}$$

   - **Hysteresis Thresholding**
     In order to remove the remaining noise from the image, thresholding is applied. All values which are below a given threshold are discarded. Given two thresholds t_1 and t_h with $t_1 < t_h$, the algorithm generates the refined image E_h as:

     (a) Locate the next unvisited pixel $E_m(i,j)$ with $E_m(i,j) > t_h$.

     (b) Start from $E_m(i,j)$ and follow the edges $E_m(k,l)$ in E_m in both directions perpendicular to the edge normal, as long as $E_m(k,l) > t_1$. Mark each such pixel $E_m(k,l)$ as visited.

     (c) Set all pixels in E_h to 0.

     (d) Traverse all the points $E_m(k,l)$ in the contour lists and set $E_h(k,l) = 1$.

3. Output:

   Your program should output the final result as well as intermediate results. Store the filtered image, before edge detection is applied; the edge strength image before nonmax suppresion; the edge strength image after nonmax suppresion; the final edge strength image after applying the hysteresis thresholds.

   Give your filenames meaningful names that are composed of the original filename, the type of image, and the parameters used to compute the image (e.g. standard deviation of Gaussian kernel, hysteresis thresholds).