Exercise 02 for MA-INF 2201 Computer Vision WS21/22
21.10.2021
Submission on 28.10.2021

1. **Fourier Transform** In this task, we will show that the convolution in spatial domain can be computed by multiplication in the frequency domain. Read the image `einstein.jpeg`, and create a 7×7 Gaussian kernel with `sigma=1` (you can use `cv2.getGaussianKernel`).

   - Blur the image using convolution in spatial domain (`cv2.filter2D`) with the created Gaussian kernel.
   - Blur the image using Fourier Transform and multiplication. You can use `numpy.fft`.
   - Print the mean absolute difference of the two blurred images.

   (2 Points)

2. **Template Matching** In this task, we will implement template matching using normalized cross-correlation as similarity measures. Read the image `lena.png` and the template `eye.png`

   - Implement normalized cross-correlation.
   - Implement template matching using your implementation of normalized cross-correlation.
   - Draw rectangles where *similarity* ≥ 0.7, you can use `np.where`.

   (2 Points)

3. **Gaussian Pyramid** In this task, we will build the Gaussian pyramid and make template matching faster by utilizing pyramid. Read image `traffic.jpg` and the template `traffic-template.png`

   - Build a 4 level Gaussian pyramid.
   - Build a 4 level Gaussian pyramid using `cv2.pyrDown`. Compare it with your implementation by printing the mean absolute difference at each level.
   - Do the template matching by using your implementation of normalized cross-correlation, print the time taken by this routine.
   - Use the pyramid technique to make template matching faster. Follow the procedure described in the lecture slides. Print the time taken by this routine.
   - Show the template matching results using the pyramid technique.

   (8 Points)

4. **Edges** In this task, we will detect edges in the image using derivative of a Gaussian kernel. Read the image `einstein.jpeg`.

   - Compute the weights of the derivative (in x) of a 5x5 Gaussian kernel with `sigma=0.6`.
   - Compute the weights of the derivative (in y) of a 5x5 Gaussian kernel with `sigma=0.6`.

- To get the edges, convolve the image with the kernels computed in previous steps. You can use `cv2.filter2D`.
- Compute the edge magnitude and the edge direction (you can use `numpy.arctan2`). Display the magnitude and direction.

(3 Points)

5. **Distance Transform** In this task, we will compute the precise Euclidean distance transform.

   - Read the image `traffic.jpg`, convert it to grayscale, extract the edges using `cv2.Canny`. Display the result.
   - Compute the precise Euclidean distance transform of the image by using [1].
   - Compute the precise Euclidean distance transform of the image by using `cv2.distanceTransform` compare it with your implementation by printing the mean absolute difference.

   (5 Points)

[1] Pedro Felzenszwalb and Daniel Huttenlocher. Distance transforms of sampled functions. Technical report, Cornell University, 2004