CENG 391 – Introduction to Image Understanding
Homework 2

December 3, 2021

Due Date: December 16, 2021

Download and extract the contents of `ceng391_hw02_code.tar.gz`.

Exercise 1 Image Filtering with a Gaussian Filter

a. Write a new member function `Image::smooth_x` that takes a single-precision floating point number `sigma` and convolves the image in the x direction with a Gaussian filter with the corresponding standard deviation. Assume that the filter coefficients are zero after two standard deviations.

b. Write a new member function `Image::smooth_y` that performs the same operation in the y direction.

c. Write a new member function `Image::smooth` that performs Gaussian smoothing both in the x and y directions with standard deviations `sigma_x` and `sigma_y`.

Hint: To compute the filter first compute the filter length in one direction, say l. The filter length should be $2 * l + 1$ since the Gaussian is symmetric. Fill in coefficient values by sampling the Gaussian form

$$\exp^{-0.5\frac{x^2}{\sigma^2}}$$

by assuming the filter center is at $x = 0$. Then normalize the filter such that the elements sum up to 1. Note that, we did not need the normalization factor $\frac{1}{\sqrt{2\pi\sigma^2}}$ since we normalize the filter coefficients at the end.

Exercise 2 Image Derivatives

a. Write a new member function `Image::deriv_x` that takes computes the image derivative in the x direction using a filter of the form $\begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$. The results should be returned in a newly allocated array of type `short` which can store negative values.

b. Write a new member function `Image::deriv_y` that takes computes the image derivative in the y direction using a filter of the form $\begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$. The results should be returned in a newly allocated array of type `short` which can store negative values.

Exercise 3 Geometric Transforms

a. Write a new member function `Image::rotate` that takes a single-precision floating point number θ and an `Image` pointer `out`. After the function call finishes the image pointed by `out` should contain the result of applying the rotation

$$\mathbf{x}' = \mathtt{R}_\theta \mathbf{x} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \mathbf{x}$$

with nearest neighbor sampling.

b. Add an option to perform bilinear sampling to the function `Image::rotate`.

c. Add an option to perform the rotation around the image center.

Hint: You must not change the size of the image `out`.