CENG 391 – Introduction to Image Understanding
Homework 2

November 23, 2020

Due Date: December 04, 2020

Download and extract the contents of `ceng391_03T_image_filtering.tar.gz`.

Exercise 1 2D Image Filtering

a. Write a new member function `Image::filter_2d` that takes an odd integer n, and an $n \times n$ single-precision floating point matrix `K`. The function should return a new image that is of size

$$(w() - n + 1) \times (h() - n + 1)$$

and filtered by the kernel `K`.

Hint: If n is even you may take n as the largest odd number smaller than n. Make sure to clamp the filter results to the range $[0, 255]$. The size of the output is smaller so that you do not need to worry about the image borders.

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::warp_affine` that takes a two-by-two transform matrix `A` and a two-by-one translation vector **`t`**, and an `Image` pointer `out`. After the function call finishes the image pointed by `out` should contain the result of applying the affine transform

$$\mathbf{x}' = \mathtt{A}\mathbf{x} + \mathbf{t} = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \mathbf{x} + \begin{bmatrix} t_1 \\ t_2 \end{bmatrix}$$

with nearest neighbor sampling.

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

Hint: You must not change the size of the image `out`. Assume that the matrix and vector entries are stored in the double-precision floating point format and the matrix is stored in the column major order (Its entries are stored in memory in the order $[a_{11}, a_{21}, a_{12}, a_{22}]$).