KAIST CS101 2020 Fall

Homework #2

Editing image

Due date: 2020. Oct. 12th

Please read the homework description carefully and make sure that your program meets all the requirements stated. The homework is **an individual task**. You may discuss the problems with others, but you should write your own code. **You will get an F for the entire course if your homework includes any plagiarism.**

DO NOT USE any additional Python external libraries except cs1media.
DO NOT WRITE CODE outside the allowed section. Otherwise, your program may not be properly graded.

The homework consists of four subtasks (Tasks 2-1 to 2-4) that will be implemented and graded separately for a maximum total of 50 points.

You will submit the homework online on the Elice course website that performs automatic grading. You may resubmit as many times as you wish. Your score for the assignment will be your most recent submission before the due date. We do not accept any late submissions.

You are encouraged to use the Elice boards to ask questions or to add comments.

Overview

In this homework, you will write several functions that each takes inputs of one or more images and returns a single output image.

Note: In all four subtasks, every division operation should be performed with "//" operator of Python.

Tasks 2-1: Image downsizing (15 pts.)

This task requires taking an image and reducing its resolution. Given an input image, your goal is to create a replicate of the image with a reduced size.

All images have fixed numbers of pixels with predefined widths and heights. However, there are circumstances where you need to fit your image in a smaller frame. To achieve this, you need to reduce the image's size while preserving the ratio with minimal loss of visual information of the original picture.

For task 2-1, you need to reduce the image's size in half (0.5 x width and 0.5 x height).

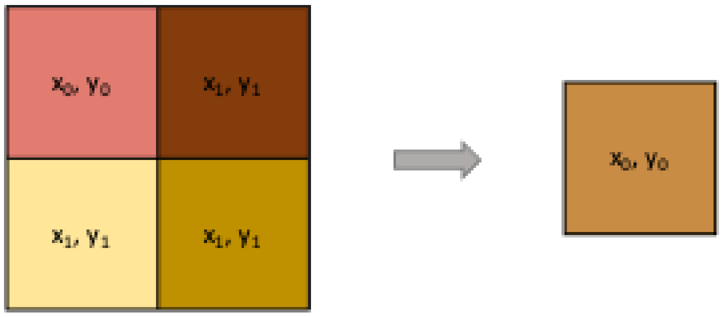


Fig 1. Example of downsizing 2x2 pixels to 1x1 pixel

The average RGB values of 2x2 adjacent pixels are calculated from the input image and used to fill the downsized 'output_image.' The 'create_picture' function creates a new cs1media image object with specified widths and heights. Use this function not to modify the original image. You can refer to the 'Photo processing with cs1media' document in Elice.

※ Note that the provided code already calls this function.

Input

- The reduce function receives a cs1media image object.

  ※ Note that the image's widths and heights will always be multiples of 2 when tested.

Output

- The downsized image (cs1media image object)

Example

Input image
(size: 400 X 400)

Output image
(size: 200 X 200)

Task 2-2: Image downsizing with variable factors (10 pts.)

In Task 2-1, the reduce function downsizes an image by half (factor = 2). In Task 2-2, you complete the reduce function to scale down the image with a variable factor.

Input

- The reduce function receives a cs1media image object.

  ※ Note that the image's widths and heights will always be multiples of a factor when tested. (i.e., During testing, the factor will always be a common divisor of height and width of the input image)
- The factor in downsizing the given image.

Output

- The downsized image (cs1media image object).

Example

Input image
(size: 400 X 400)

Output image when
factor=4
(size: 100 X 100)

Task 2-3: Blurring (15 pts.)

In this task, you will create a blur function that blurs a specific area of an input image.

For blurring, the function receives a blurring area defined by blur_area_center and blur_area_size. You need to assign new RGB values for every pixel within the blurring area in the output_image created using the create_picture function.

For the new RGB value of a specific pixel, you need to calculate the average RGB values from 5x5 pixels surrounding the target pixel. For example, in Figure 2, the RGB value of (3,3) is set by calculating the average RGB values of the blue 5x5 area (25 pixels).

※ Note that the blurring area will always be near the center of the test image when testing. The areas to calculate the average RGB from will always be within the original image size as well.

※ Note that when calculating the average RGB values of the 5X5 pixels surrounding the target pixel, **pixels of the original image (input_image) should be used**.

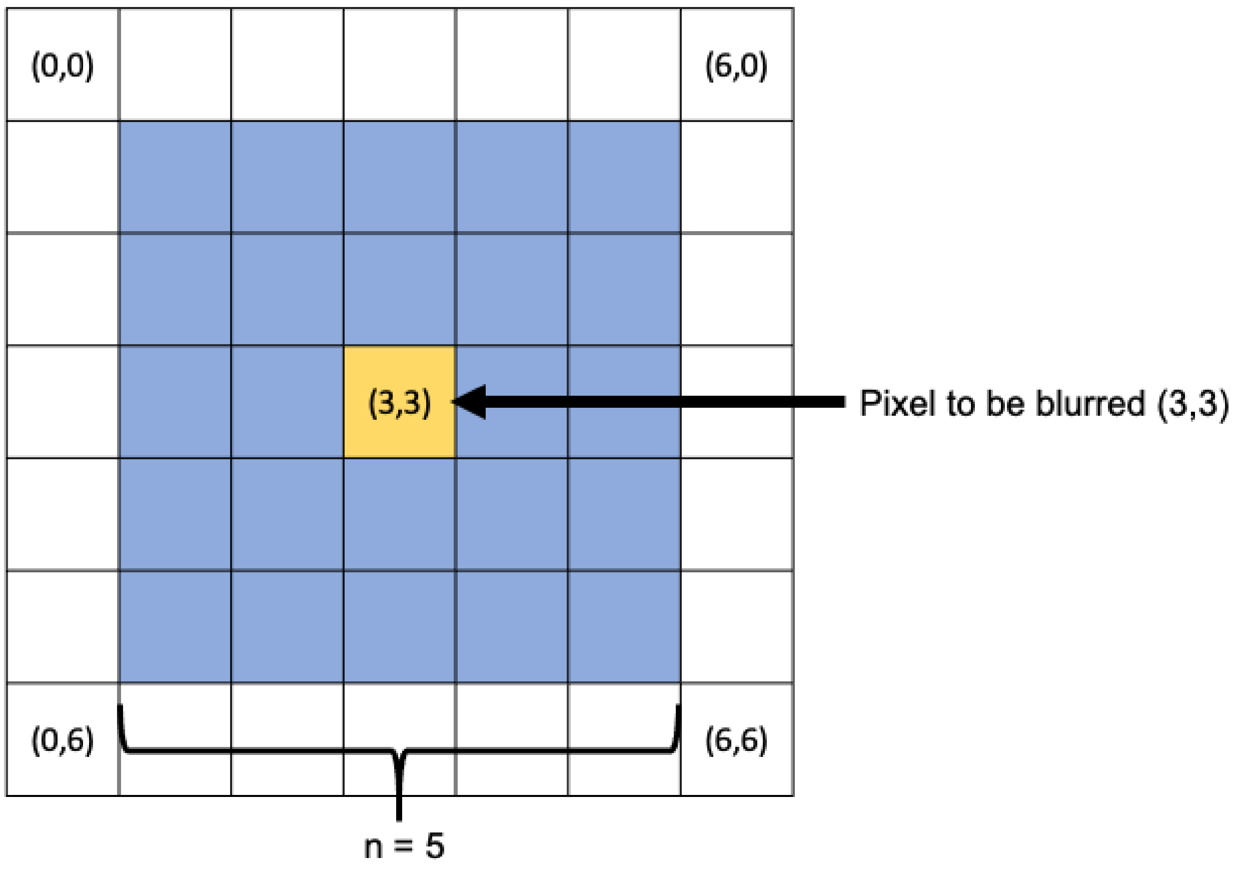


Fig 2. Calculating RGB value for the pixel to be blurred. Yellow represents the target pixel, and blue represents the pixels to calculate the average.

Input

- input_image: cs1media image object.
- blur_area_center: coordinates of the center of the blurring area.
- blur_area_size: size of the blurring area. (ex: if blur_area_size = 101, the blur area is 101 x 101 with its center at blur_area_center.)

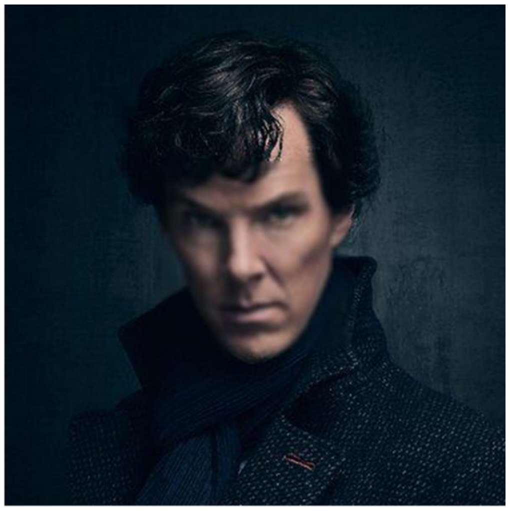
  ※ Note that when testing, blur_area_size will always be an **odd number**.

Output

- A newly created cs1media image object with a blurred effect.

Example

Input image
(size: 400 X 400)


Output image when
blur_area_center=(200, 200),
blur_area_size=151
(size: 400 X 400)

Task 2-4: Varying blurring quality (10 pts.)

In Task 2-4, we introduce a new variable n to the blur function from Task 2-3, which controls how **blurry** the effect will be.

The newly introduced n defines the size of the surrounding pixels used to calculate the RGB value for the blurred pixel. In Task 2-3, n was a fixed value of 5, using only 5x5=25 pixels to calculate the average. Now, we instead use (n x n) surrounding pixels. That is, for the new RGB value of a specific pixel, you need to calculate the average RGB values from (n x n) pixels surrounding the target pixel.

In Task 2-4, you create a blur function that varies the level of blurriness depending on n.

Input

- input_image: cs1media image object.
- blur_area_center: coordinates of the blurring area center.
- blur_area_size: size of the blurring area. (ex: if blur_area_size = 101, the blur area is 101 x 101 with its center at blur_area_center.)

  ※ Note that when testing, blur_area_size will always be an **odd number**.
- n: blurring factor.

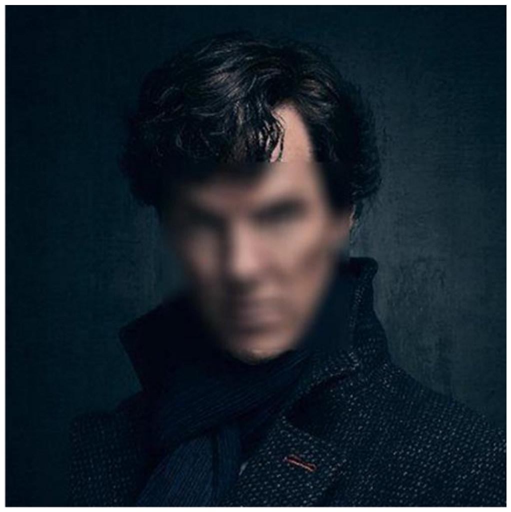
  ※ Note that when testing, n will always be an **odd number**.

Output

- A newly created cs1media image object with a blurred effect.

Example

Input image
(size: 400 X 400)



Output image when
blur_area_center=(200, 200),
blur_area_size=151, n=15
(size: 400 X 400)