National University of Sciences and Technology (NUST)
School of Electrical Engineering and Computer Science

Department of Computing

EE433: Digital Image Processing

Class: BSCS 7AB

Lab 2: Image up-scaling/down-scaling from scratch

Date: 27th January, 2020

Time: 9.00Am to 12.00Pm & 2.00Pm to 5.00Pm

Instructor: Dr. Asif Ali

National University of Sciences and Technology (NUST)
School of Electrical Engineering and Computer Science

Lab2: Image up-scaling/down-scaling from scratch.

Objective:

The objective of this lab is to introduce the concepts of interpolation and averaging for up-scaling or down-scaling images respectively. Furthermore, to build up understanding about opencv2's function for performing these tasks effortlessly.

Theory:

In order to down-scale an image (analogous to zooming out) from bigger form factor to a shorter one (see Figure 1). For instance, all blue pixels (let's call it a ***cell***) from input image should contribute to a single pixel (also blue for convenience) in the output image. This type of down-scaling can be used to create **50%**, **33%** and **25%** etc of the original image by choosing the *cell* size to be **2x2**, **3x3** and **4x4** respectively.

There exist various methods which can be use, such as:

- Take a single pixel value from a cell and place that value as-is in output image
- Take all values of cell and calculate an *average* of all values
- Take all values of cell and calculate *median* of all values.

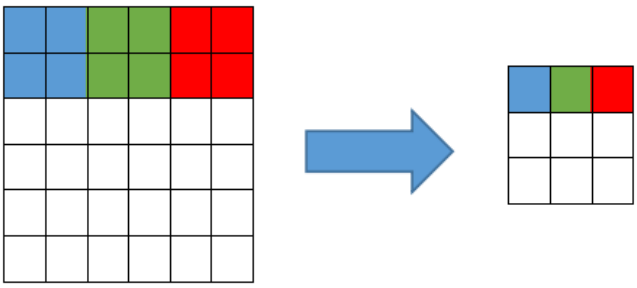

Figure 1: Down-scaling an image of cell 2x2.

Similarly, a smaller input image can be up-scaled to a bigger one by using interpolation techniques (see Linear for 1D data, Bilinear, Bicubic and Spline for image data). For now, image that you are supposed to convert 3x3 input image to a 6x6 output image. For simplicity in future understanding let's call our area of ***cell*** which contains (blue, green, red and purple pixels)

You have following unknown values (for cell 1) which should be approximated/calculated as follows:

- Pixel **1** can be calculated using the blue and green pixels.
- Pixel **2** can be calculated using the blue and red pixels.

- Pixel **3** can be calculated using the blue and purple pixels **and** from green and red pixels. (you should be able to make a good approximation from both diagonal interpolated values)
- Pixel **4** can be calculated using the green and purple pixels.
- Pixel **5** can be calculated using the red and purple pixels.

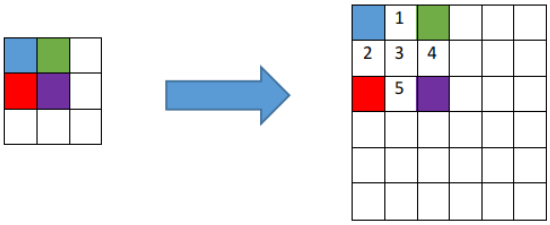

Figure 2: Up-scaling an image to 200%

Once the values are interpolated, you can move your cell to either one column to the right or one row down (depending upon how you implement your loops) and the task of interpolation continues for the complete image. The interpolation performed here is still 1D interpolation but since images are 2D components, an efficient interpolation technique can be implemented which can take all four elements of *cell* instead of two. That type of 2D linear interpolation is known as **Bilinear interpolation.**

Note:-

For details of bilinear interpolation you may refer to the example explained in the lab. For further details you may refer to the following readings:-

https://en.wikipedia.org/wiki/Bilinear_interpolation

https://theailearner.com/2018/12/29/image-processing-bilinear-interpolation/

Following interpolation techniques are facilitated by OpenCV2's ***resize()*** function:

- **interpolation** –

  interpolation method:

  - **INTER_NEAREST** - a nearest-neighbor interpolation
  - **INTER_LINEAR** - a bilinear interpolation (used by default)
  - **INTER_AREA** - resampling using pixel area relation. It may be a preferred method for image decimation, as it gives moire'-free results. But when the image is zoomed, it is similar to the `INTER_NEAREST` method.
  - **INTER_CUBIC** - a bicubic interpolation over 4x4 pixel neighborhood
  - **INTER_LANCZOS4** - a Lanczos interpolation over 8x8 pixel neighborhood

Figure 3:

Concluding remarks: Notice in both up-scaling and down-scaling everything is bound with integer number of pixels and the resulting image affected with that choice. What if someone wishes to resize the input image to some arbitrary size?

Lab Tasks:

Task 1:

- Read "lena.tiff" from directory.
- Write a program which traverses each pixel of image and perform down-scaling as follows:
  - The ratio for down-scaling should be 50%.
  - Use averaging of cell pixels to approximate better output image.
  - Use *select-one* approach to down-scale.
  - Save both images.
  - Compare the two images and express your findings.
- Repeat all of the above for 25% down-scaling
- Use opencv's **resize** function and admire the workings of cv2.resize function especially with interpolation techniques presented in Figure 3.

Task 2:

- Read "lena.tiff" from directory.
- Write a program which traverses each pixel of the input image and perform up-scaling as follows:
  - The ratio for up-scaling should be 200% (i.e. cell size of 2x2)
  - Use interpolation as discussed in theory part and calculate unknown pixel values.
  - Save the output image.
- Express your findings on how interpolation up-scaled an image.
- Use opencv's **resize** function and admire the workings of cv2.resize function especially with interpolation techniques presented in Figure 3.

Submission Guidelines:-

Deliverables:

You have to submit a word document containing all the codes and screen shots of output of all codes. Do not submit any zip folder containing code and screenshots separately.

Penalty for copying code from internet:-

You are strictly not permitted to copy code from internet because this lab will be evaluated on the base of quiz (in next lab) so anyone who has poor understanding of procedure overall will get a penalty in lab submission part as well.