HW2

Preparation - 1

- Please make sure you have read the Lectures 6 & 7 & 9 slide decks, especially about convolutions and the binomial filter.

- Please read Textbook chapters 15 & 17, at least those covered by lectures.

- The notations in this slide deck may be a bit different from the lecture slides! The goal is to make implementation easier.

Preparation - 2

- Please make sure you have read the Lectures 9 & 10 slide decks, especially about the Gaussian pyramid and the Laplacian pyramid.
- Please read the Textbook chapter 23 about the Gaussian pyramid and the Laplacian pyramid.
- The Gaussian pyramid down-samples an image sequentially.
- The Laplacian pyramid records “information loss” during down-sampling.
- By combining them, we can reconstruct the image of the original size.

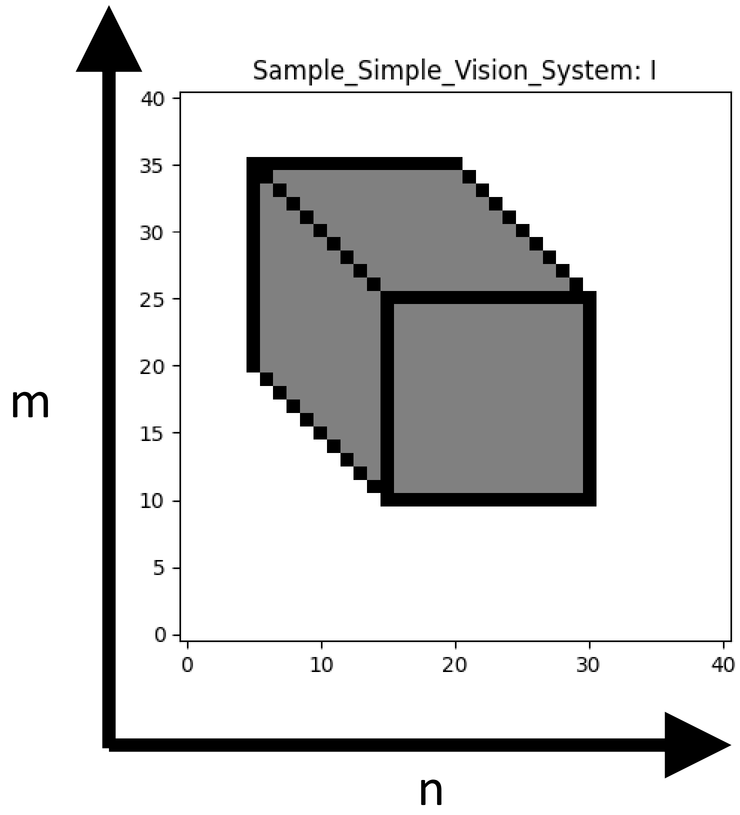
Caution

- Please do not import packages (like scikit learn) that are not listed in the provided code.

- In this homework, you are NOT allowed to use NumPy's or other Python libraries' built-in **convolution, filter functions, down-sampling, up-sampling, Gaussian pyramid, and Laplacian pyramid functions.** If you use them, you will get 0 points for the entire assignment.

Convention

- In this homework, given a map (or a matrix), say I
  - I[n, m] means the i-th horizontal index (left-right) and j-th vertical index (bottom-up)
  - n >= 0, m >= 0



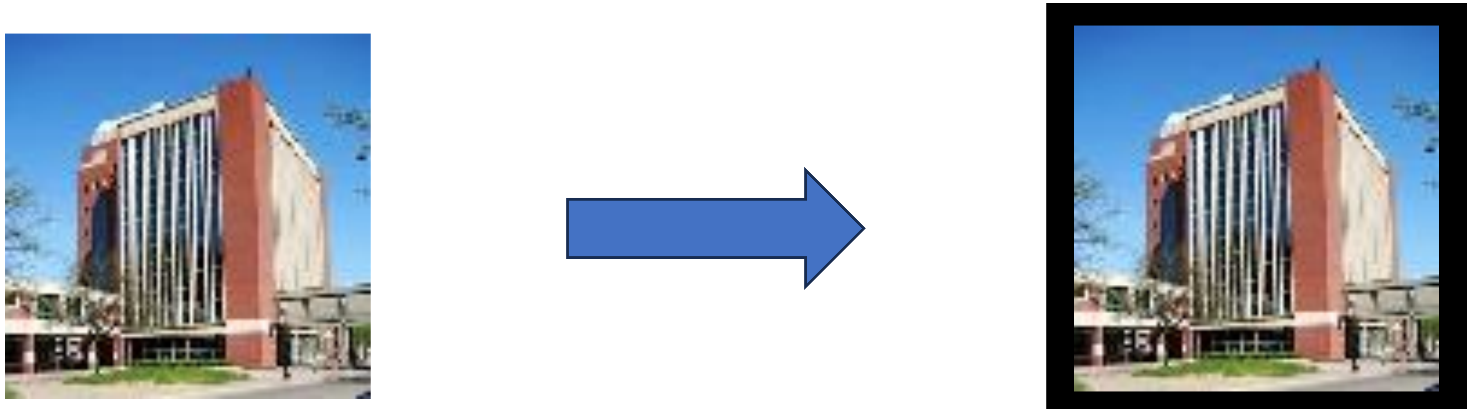
Color images

- Please note that a color image I means that the image I is a 3D tensor. The 3rd dimension corresponds to R, G, and B.

- In this homework, you will process each channel separately. You can extract each by I[:, :, c], where c is between 0 and 2.

Question 1: Convolution

- There are many variants, but in this homework, please follow the formula below.
    - Given an image I, we will first do zero padding (think about why)



I   I_pad

Question 1: Convolution

- Then, we perform
- $I_{out}[n, m, c] = \sum_{k=0}^{K-1} \sum_{l=0}^{L-1} I_{pad}[n + k, m + l, c]\, h[K - 1 - k, L - 1 - l]$
- K and L are the 2D kernel h's shape
- The range of *n* and *m* are [0, N-1] and [0, M-1], respectively, where N and M are the input image I's shape (not I_pad)
- That is, I_{out} will have the same shape as the input image I

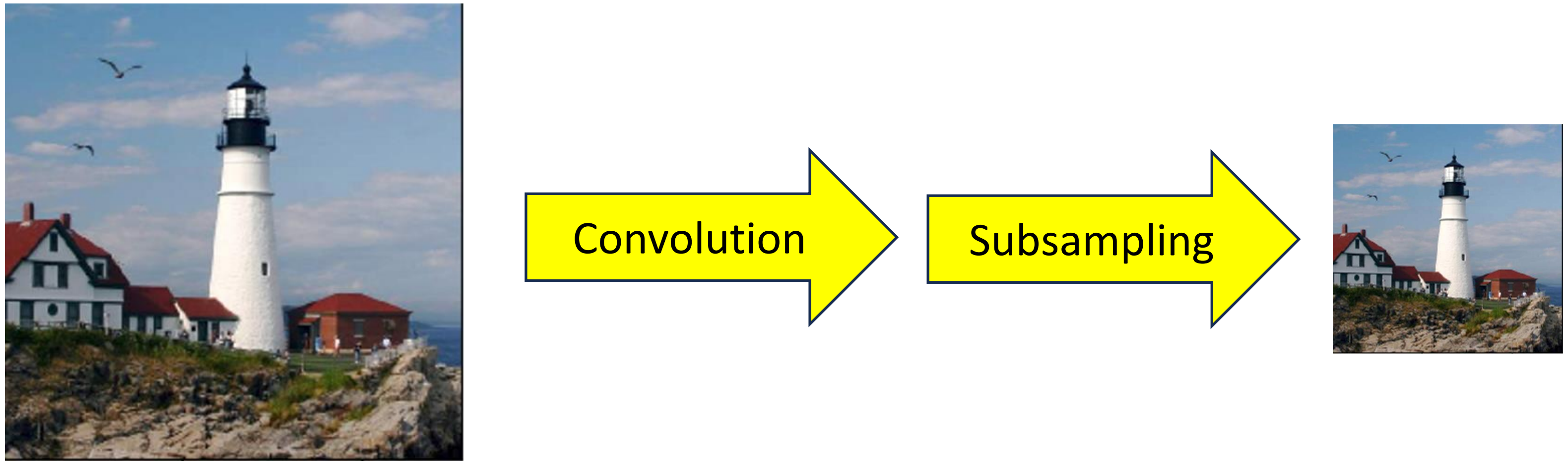
Question 1: Convolution

- In your implementation, if you can implement $\sum_{k=0}^{K-1}\sum_{l=0}^{L-1}[something]$ without using a for loop, it will save a lot of computation time in Python

Question 2: Down-sampling

- There are many variants, but please follow the formula below.
  - Given an image I, we will first convolve it with a 2D binomial filter to obtain I_convolved
    - I and I_convolved have the same size

  - We will then sub-sample the convolved image I_convolved by a factor of two
    - If I_convolved is 128 x 128, the subsampled image I_down is 64 x 64 by keeping pixels at "even" Python indices. That is, I_down[n, m, :] = I_convolved[2n, 2m, :]



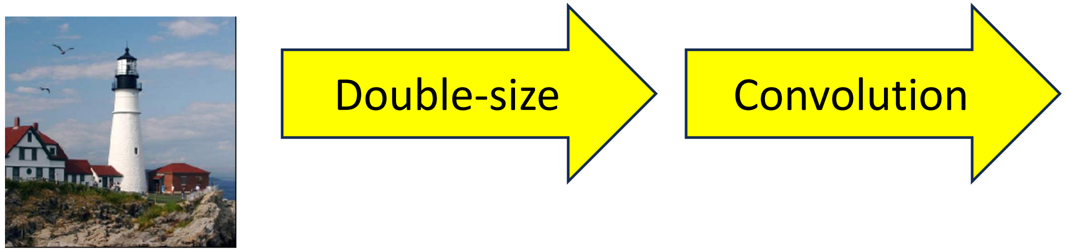
Question 3: Gaussian pyramid

- Create a sequence of sub-sampled images

Question 4: Up-sampling

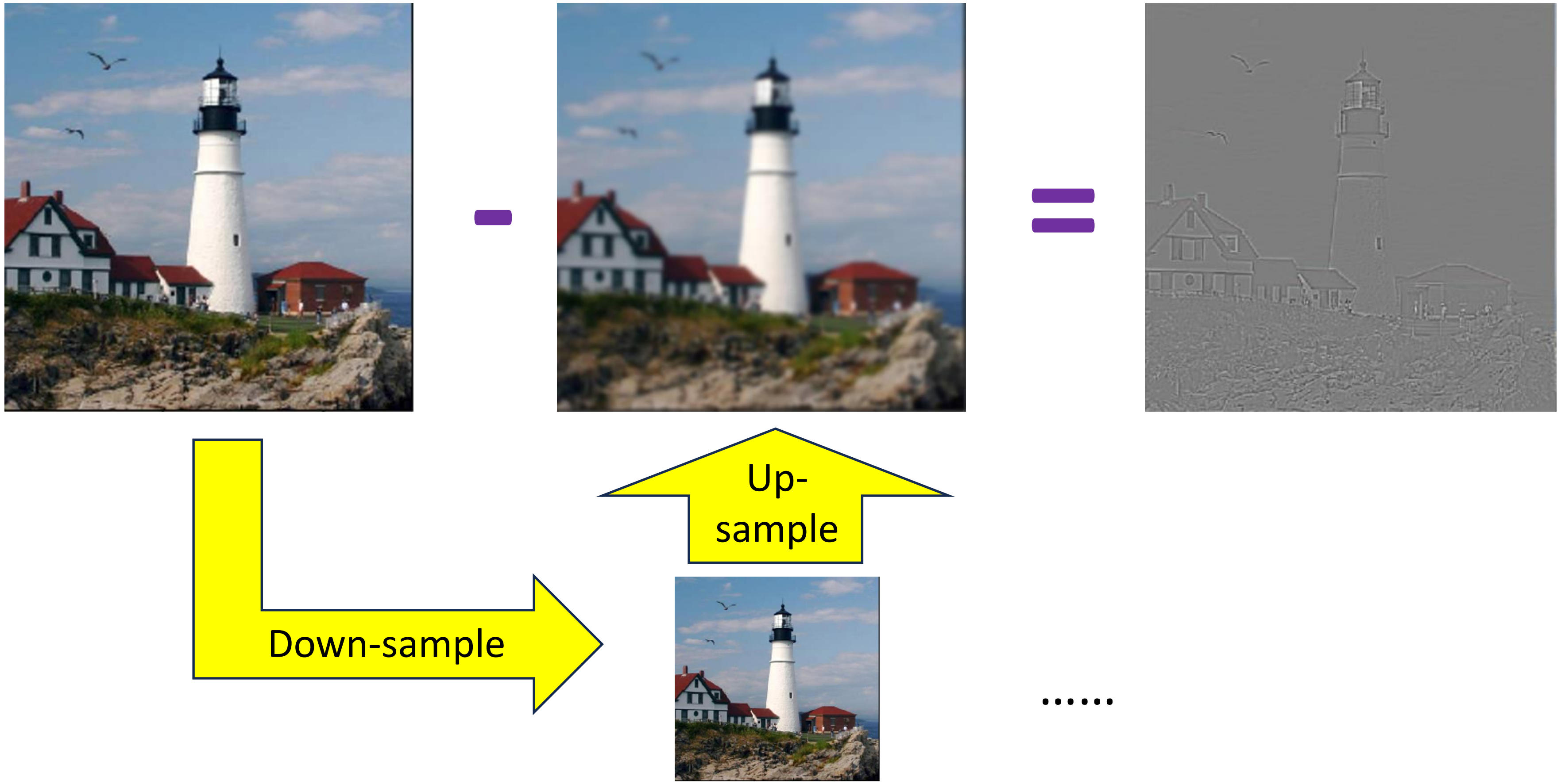
- There are many variants, but please follow the formula below.
  - Given an image I, we will first double its size in each dimension to obtain I_up
    - If I is 64 x 64, I_up is 128.
    - I_up[n, m, :] = 4 * I[n/2, m/2, :] when n and m are “even” Python indices
      - **Please make sure you multiply the pixel values by 4**
    - I_up[n, m, :] = 0 when n and m are “odd” Python indices

  - We will then convolve I_up with a 2D binomial filter to obtain I_convolved
    - I_up and I_convolved have the same size



Question 5: Laplacian pyramid

- Create a sequence of residual images



Question 6: Image reconstruction