FollowUp 0

1. Run the 01_image_processing_PIL_tutorial.ipynb

2. Masks are geometric filters on an image. For instance, if we want to extract a region of an image, we may do it by multiplying the matrix of the original image by a matrix of equal size containing 1′s in the region we want to keep and 0′s otherwise.

In this exercise we extract a circular region of the image ***lena_gray_512.tif*** of radious 150. Follow the next instructions and report every step:

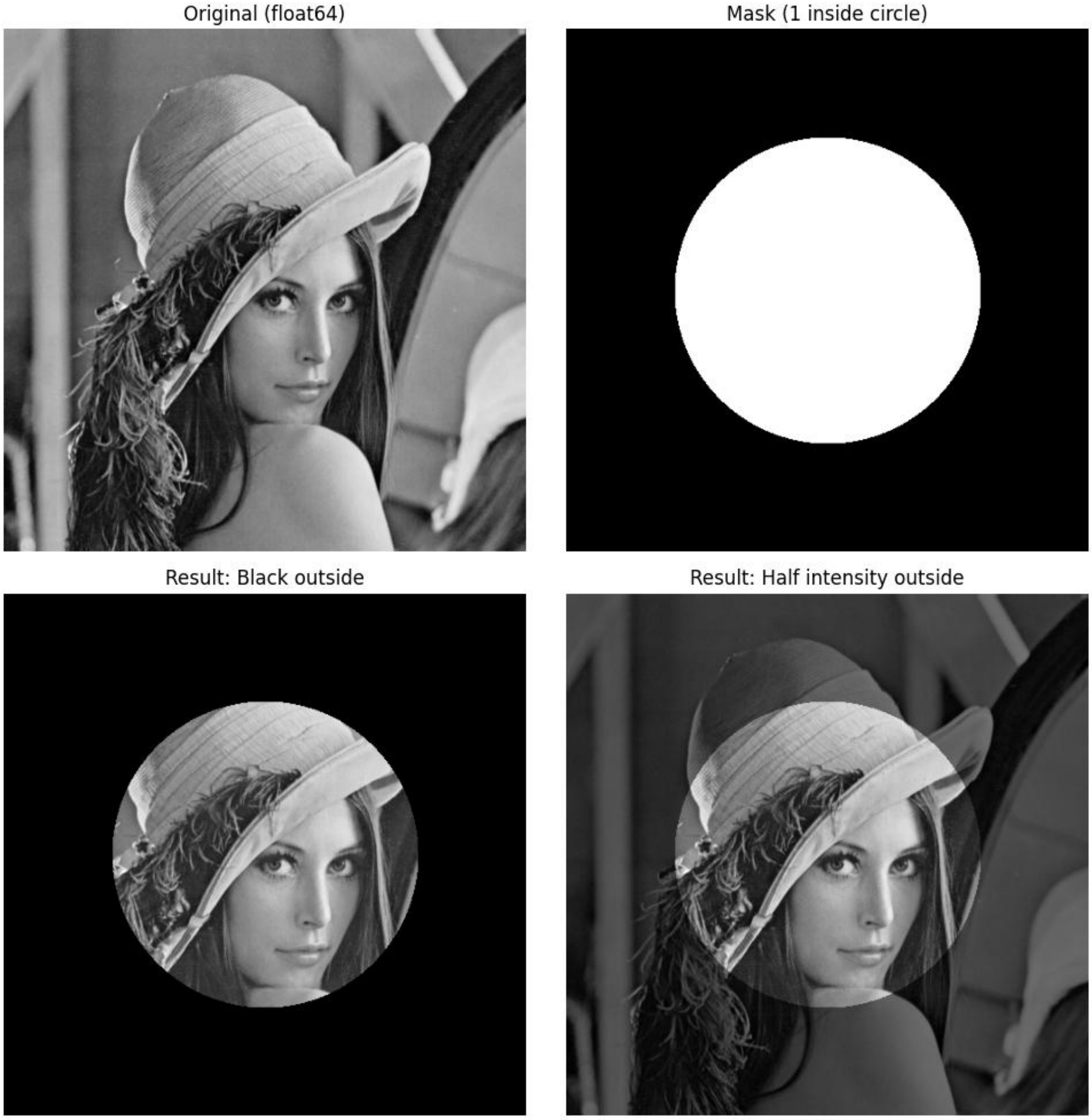
- Read the image and convert it to double.
- Create a matrix of the same dimensions filled with zeros.
- Modify the above matrix to contain 1′s in a circle of radious 150, i.e. if (j−cx)2+(i−cy)2<150exp2, where (cx,cy) is the center of the image.
- Multiply the image by the mask (they are matrices!)
- Show the results.

When multiplying by zero, you set to black the pixels out of the circle. Modify the program to make visible those pixels with half the intensity.

Hint

a.shape[0] is the number of rows of **a** and **a.shape[1]** the number of columns.

3. Briefly compare PIL and CV2 libraries, similarities, strengths and weakness.



Original (float64)

Mask (1 inside circle)

Result: Black outside

Result: Half intensity outside

Brief comparison: PIL vs OpenCV (cv2)

- **Similarities**

  - Both can read, write, and display images.

  - Both support color conversions, resizing, cropping, drawing, and basic filtering.

- **Strengths (PIL / Pillow)**

  - Pythonic, lightweight, easy for simple image I/O and manipulation.

  - Integrates nicely with NumPy; arrays are easy to convert with `np.asarray(Image)` and `Image.fromarray`.

  - Great for pipelines that generate or annotate images for reports/plots with Matplotlib.

- **Weaknesses (PIL / Pillow)**

  - Limited advanced computer vision algorithms (e.g., feature detection, optical flow, DNN inference).

  - Performance not as optimized for large-scale CV tasks.

- **Strengths (OpenCV / cv2)**

  - Very fast C++ backend; broad set of algorithms for computer vision, image processing, and video.

  - Extensive functionality: filtering, morphology, geometric transforms, feature matching, camera calibration, DNN module, etc.

  - Good for production-grade CV pipelines and real-time processing.

- **Weaknesses (OpenCV / cv2)**

  - API can be less Pythonic; color channel order is BGR by default.

  - Heavier dependency; installation can be larger and sometimes trickier.

- **Summary**

  - Use PIL for simple, lightweight image I/O and basic transformations in Python notebooks/scripts.

  - Use OpenCV when you need performance and a comprehensive set of CV algorithms or real-time video processing.