1. NumPy Functions

`np.mean(image, axis=2)`

- **Purpose**: This converts an RGB image to grayscale.
- **Explanation**:
    - An RGB image has three color channels: Red, Green, and Blue. These are stored as separate 2D arrays within the 3D image array.
    - The `axis=2` means we're taking the mean (average) of the pixel values across the 3rd dimension, which represents the color channels.
    - For each pixel in the image, it averages the Red, Green, and Blue values to produce a single grayscale intensity value.

`np.clip(image, 0, 255)`

- **Purpose**: Ensures that pixel values in the image stay within the valid range of `[0, 255]`.
- **Explanation**:
    - Pixel values in an image can range from 0 (black) to 255 (white) in grayscale or each channel in an RGB image.
    - If any pixel values exceed 255 (too bright) or drop below 0 (too dark), `np.clip` brings them back into the valid range.
    - For example, if a pixel value is `300`, it will be clipped to `255`, and if it's `-10`, it will be clipped to `0`.

`image[::int(1/scale), ::int(1/scale)]`

- **Purpose**: Resizes the image by selecting pixels at regular intervals.
- **Explanation**:
    - In Python, the notation `::k` in an array slices that array, selecting every `k`-th element.
    - `int(1/scale)` calculates the stride. If `scale` is 0.5, the stride will be `2`, which means every 2nd pixel is selected in both height and width.
    - This effectively reduces the image dimensions by half, creating a downsampled (smaller) version of the original image.

`np.convolve()`

- **Purpose**: Applies a filter (Sobel operator) to detect edges by convolving the image with the filter.
- **Explanation**:
    - Convolution is an operation where a filter (kernel) is applied to an image to extract certain features (in this case, edges).

    - In the context of edge detection, the Sobel filter calculates the difference in intensity between neighboring pixels in both horizontal and vertical directions, highlighting areas where these changes are significant (i.e., edges).
    - The result is an image where edges are highlighted, and the rest is dark.

`np.sqrt()`

- **Purpose**: Combines the horizontal and vertical edge responses into a single gradient magnitude.
- **Explanation**:
    - The Sobel filter calculates edge intensity separately for horizontal (`edges_x`) and vertical (`edges_y`) directions.
    - The final edge strength at each pixel is the magnitude of these two gradients, calculated using the Pythagorean theorem: edge magnitude=horizontal edge2+vertical edge2\text{edge magnitude} = \sqrt{\text{horizontal edge}^2 + \text{vertical edge}^2}

`np.deg2rad(angle)`

- **Purpose**: Converts the angle from degrees to radians.
- **Explanation**:
    - Many trigonometric functions in NumPy (like `cos`, `sin`, etc.) expect angles to be in radians, not degrees.
    - This function converts an angle from degrees (0–360) to radians (0–2π) using the formula: radians=degrees×π180\text{radians} = \text{degrees} \times \frac{\pi}{180}

`np.indices((h, w))`

- **Purpose**: Generates a grid of coordinates for every pixel in the image.
- **Explanation**:
    - This function creates two arrays, one for the row indices (`y`) and one for the column indices (`x`), covering the entire image.
    - If the image is `h` pixels high and `w` pixels wide, the output will be two 2D arrays of shape `(h, w)`—one holding the x-coordinates and the other holding the y-coordinates for each pixel.
    - These coordinates are later used to perform transformations like rotation or translation.

---

2. Pillow (PIL) Functions

`Image.fromarray()`

- **Purpose**: Converts a NumPy array into a `PIL.Image`.
- **Explanation**:
    - The image data is initially stored as a NumPy array, which is a general-purpose array structure.
    - `Image.fromarray()` takes this array and interprets it as an image. This allows us to use the extensive image manipulation features provided by the `PIL` library.

`rotate(angle, expand=True)`

- **Purpose**: Rotates an image by a specified angle.
- **Explanation**:
    - The `rotate()` function rotates the image counterclockwise by the specified `angle` in degrees.
    - `expand=True` ensures that the output image is large enough to hold the entire rotated image. Without this, the rotated image might be cropped to fit the original dimensions.
    - The rotation is smooth and handled internally by PIL's transformation algorithms, making it efficient and easy to use.

`np.array(rotated_image)`

- **Purpose**: Converts a `PIL.Image` back to a NumPy array.
- **Explanation**:
    - After performing the rotation in `PIL`, the image is converted back to a NumPy array so that it can be used in further NumPy-based operations or displayed alongside other processed images using Matplotlib.

---

3. `gaussian_filter` from `scipy.ndimage`

`gaussian_filter(image, sigma=sigma)`

- **Purpose**: Applies a Gaussian blur (smoothing) to the image.
- **Explanation**:
    - A Gaussian blur is a type of low-pass filter that smooths the image, reducing noise and detail.
    - The `sigma` parameter controls the standard deviation of the Gaussian kernel. A larger `sigma` means a stronger blur, where more neighboring pixels influence each pixel's new value.
    - Mathematically, a Gaussian function is applied to each pixel and its surrounding neighbors, weighting the nearby pixels based on their distance from the center (the closer a pixel is, the more it contributes to the final value).