IMAGE PROCESSING

Introduction

-> An image can be represented as a matrix of pixels

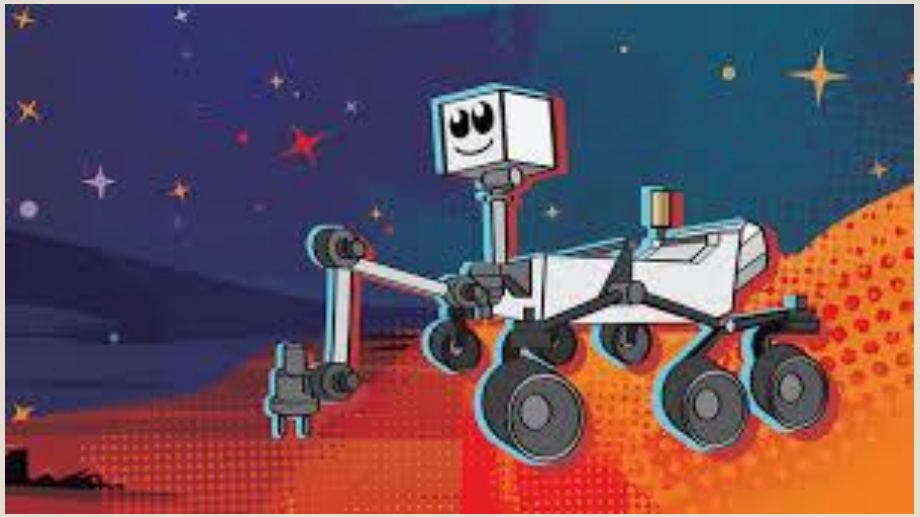
-> Each pixel is expressed as a 3D vector

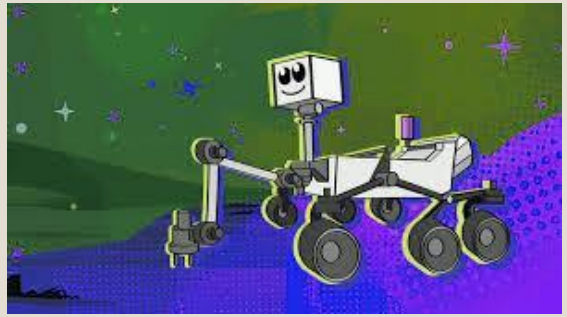
-> Composed by the amount of red, green and blue of the color.

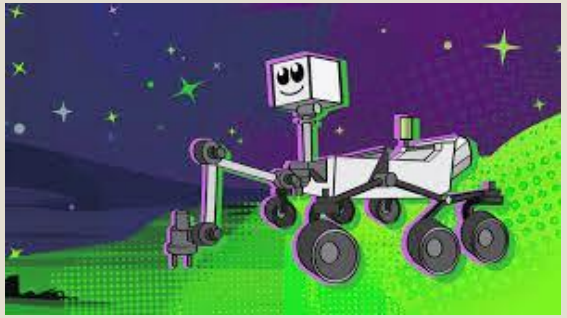
SHADES TINTS AND HUES

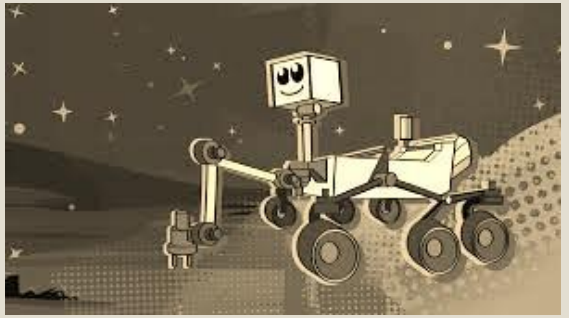
-> Filters are applied to each pixel of the matrix using the filter function
-> Filter function - basically matrix addition, subtraction and multiplication
-> The input of this function can be just a pixel
-> Each pixel has values from 0–255
-> Different combination of RGB produce different Colors - colour filters
-> Photography filters

BEFORE IMAGE









GRAYSCALING

->The components of each new pixel- obtained by calculating the average of the three components
-> Value of each pixel represents only the intensity information of the light.
-> Such images typically display only the darkest black to the brightest white.
-> Image contains only black, white, and gray colors, in which gray has multiple levels.

BEFORE AND AFTER IMAGE

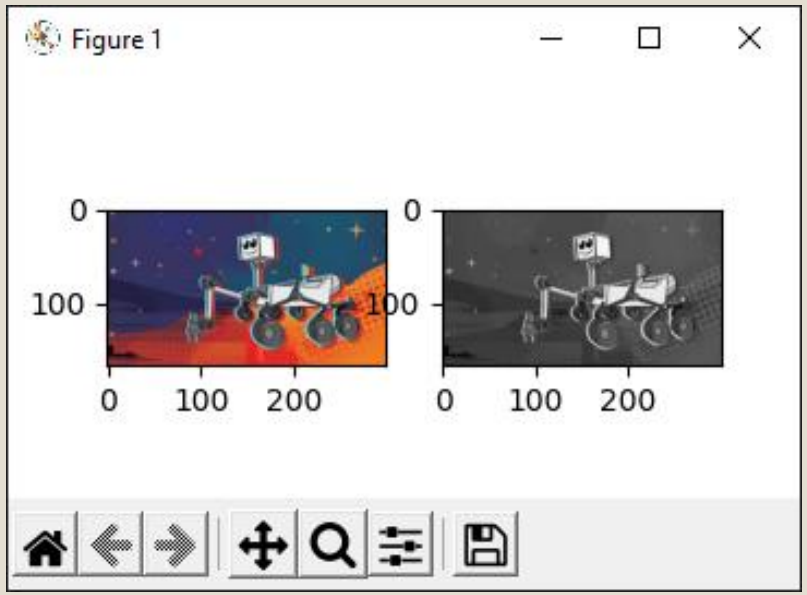


IMAGE RECONSTRUCTION USING SVD

->Image reconstruction is a technique deployed to reconstruct the image approximately towards the original image

->It uses the concept of Singular value decomposition i.e we detect the top 2 singular vectors and use them for reconstruction

->We can reconstruct SVD of an image by using **linalg.svd()** method of NumPy module.

->With the help of singular values you can reconstruct a image

Data_Matrix = U * s * V^T,

- Data_Matrix is matrix of size m x n
- **U** is an orthogonal matrix of size m*p
- **V** is an orthogonal matrix of size p*p
- **s** is an diagonal matrix of size n*p


Original
Reconstructed n = 100

SHEARING

->Shear mapping is a linear map that displaces each point in fixed direction, it substitutes every point horizontally or vertically by a specific value in proportional to its x or y coordinates.

->There are two types of shearing effects

->Shearing in x-direction

->When shearing is done in the x-axis direction, the boundaries of the image that are parallel to the x-axis keep their location, and the edges parallel to y-axis changes their place depending on the shearing factor.

->Shearing in y-direction

->When shearing is done in the y-axis direction, the boundaries of the image that are parallel to the y-axis keep their location, and the edges parallel to x-axis changes their place depending on the shearing factor.

ORIGINAL IMAGE

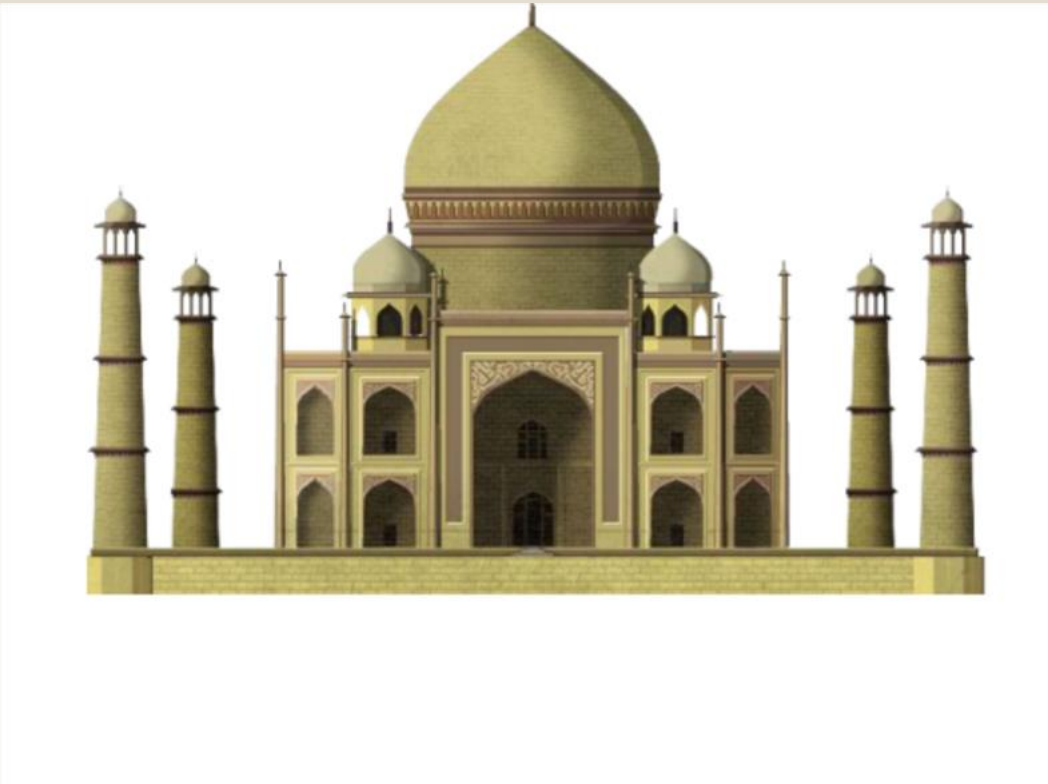
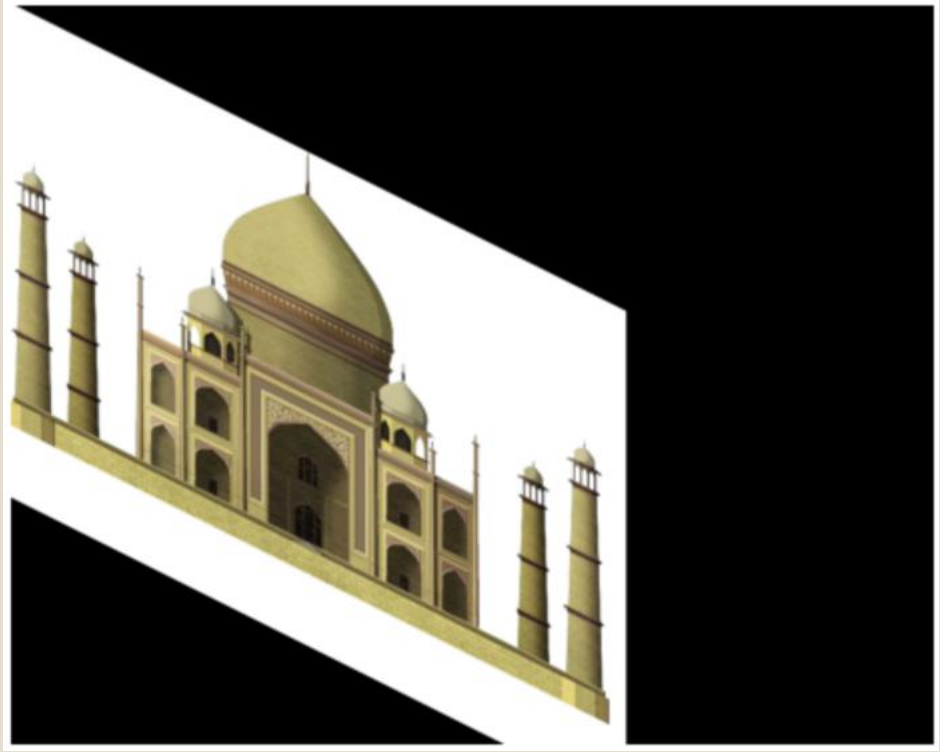


IMAGE AFTER TRANSFORMATION



LINEAR FILTER

Linear Filter: Introduction

•These process time-varying input signals to produce output signals, subject to the constraint of linearity.

•It is a filtering in which the value of an output pixel is a linear combination of the values of the input pixel's neighborhood.

•The concept used here is convolution in 2 dimension.[ie,It uses a 2D matrix]

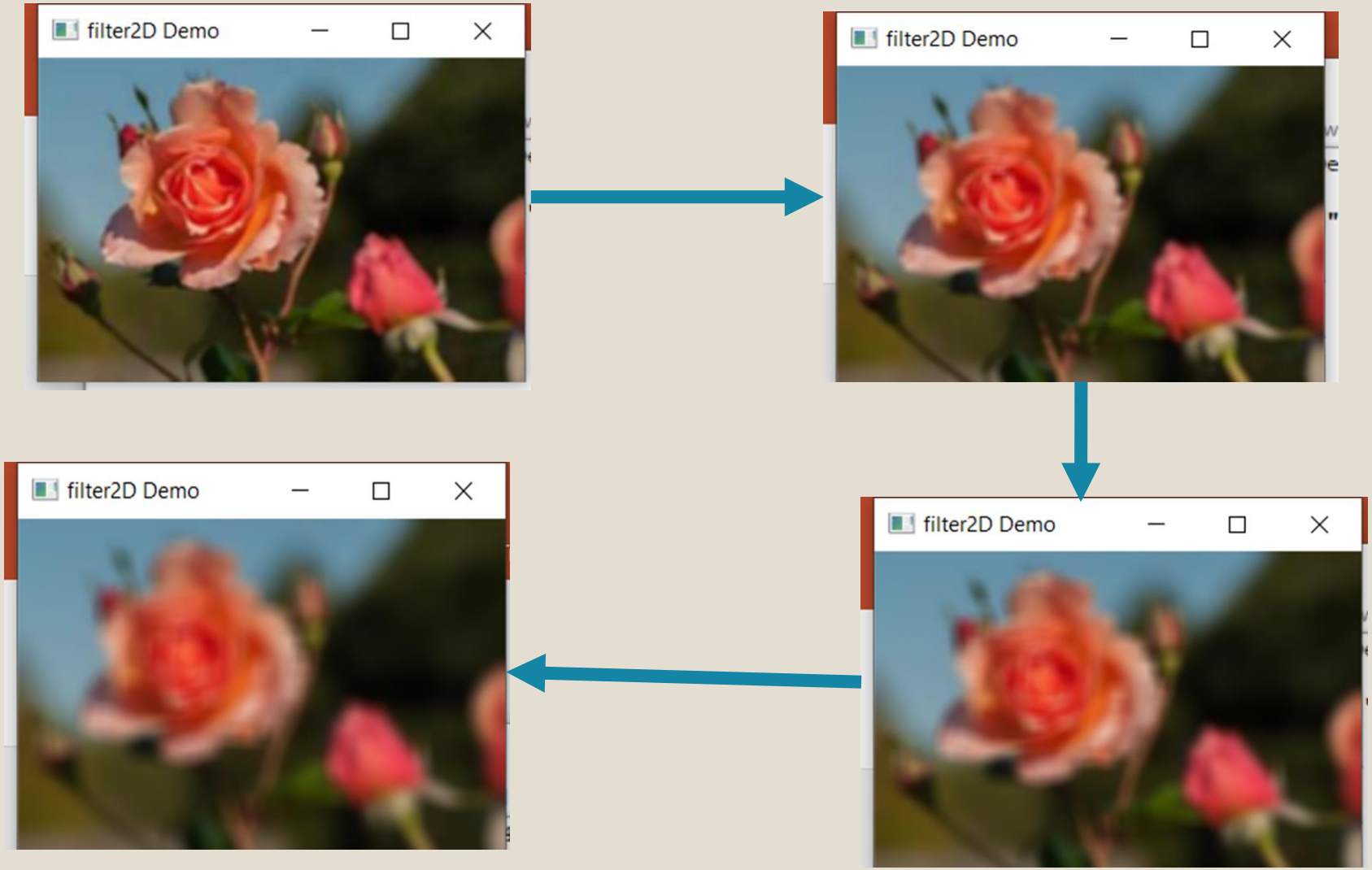
•Applications: Generic tasks such as image/video contrast improvement, denoising, sharpening, and feature enhancement.

Linear Filter: Working

The code works in the following way:

1. It first loads the original image and initializes the variable (ddepth).
2. Then we define the kernel .
3. This can be done by defining Kernel size.
4. Filling the kernel with 1’s and normalizing it.
5. Then we use the source image and Kernel to get output.
6. The output is that every 1 second the kernel size of this filter will be updated.

Linear Filter: Input and Output



RESIZING

Resizing an Image: Introduction

- It is the process of changing the scale of a discrete or continuous image by way of resolution. The whole image is present after applying resizing.

- The concept used here is, if we want to resize an image, we shall have to change its ‘pixel information’. [This is done with the help of matrices]

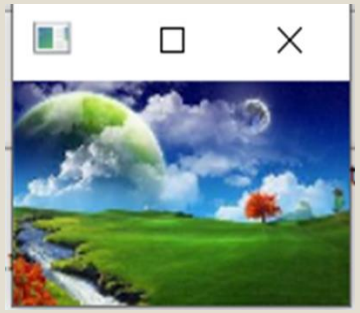
- Applications: transliteration of the image, correcting for lens distortion, changing perspective, and rotating a picture.

Resizing an Image: Working

The code works in the following way:

1. It first loads the original image and initializes the variables (x and y).
2. Then we assign 3 variables[blue plane, green plane, red plane.
3. Each variable has the specific plane value, which is got from that specific plane value of the blue-green-red image.
4. It take 1/x pixels of the row & 1/y pixels of the column for each value.
5. Then we create a zero matrix for the resized image variable.
6. Assigning the new ‘blue plane’, ‘red plane’ and ‘green plane’ values to get the resized image and displaying the same.

Resizing an Image: Input and Output



ROTATION

Rotation : Introduction

->Implementation of a rotation filter on an image that rotates the image around its center by the specified degrees .

->Uses the concept of linear transformation particularly rotation to fill in intensity values for the coordinates required

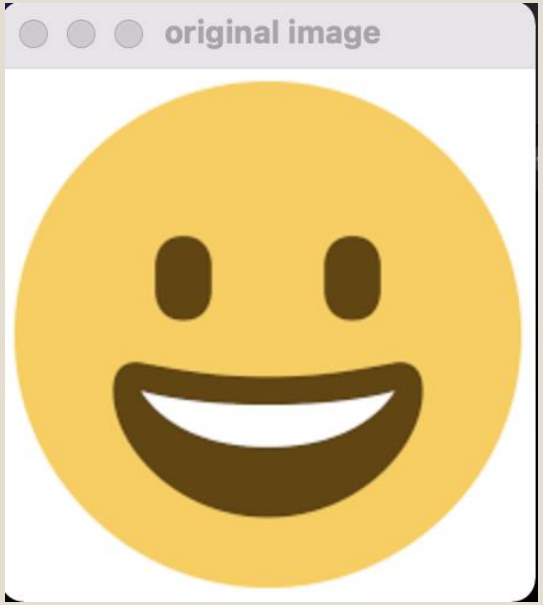
->We first find the height and width of the rotated image followed by finding the center point coordinates of both original as well as rotated image. Then we create a rotated image with appropriate height and width

Linear transformation used

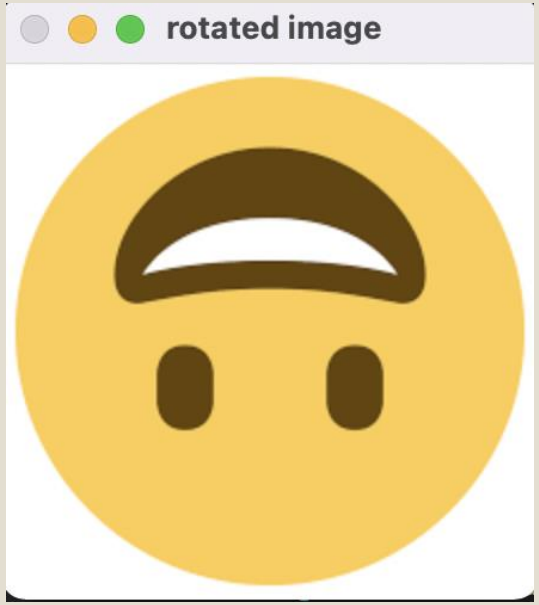
$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos(\theta) & \sin(\theta) \\ -\sin(\theta) & \cos(\theta) \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

Where theta is the angle to be rotated

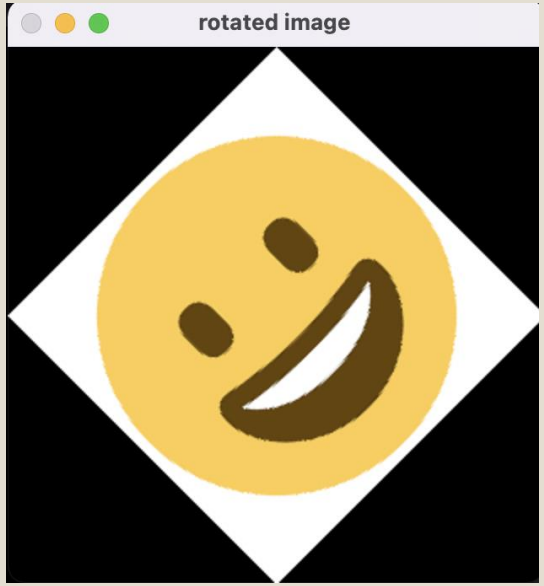
ORIGINAL IMAGE



ROTATED IMAGE(180)



ROTATED IMAGE(45)



GAUSSIAN FILTER

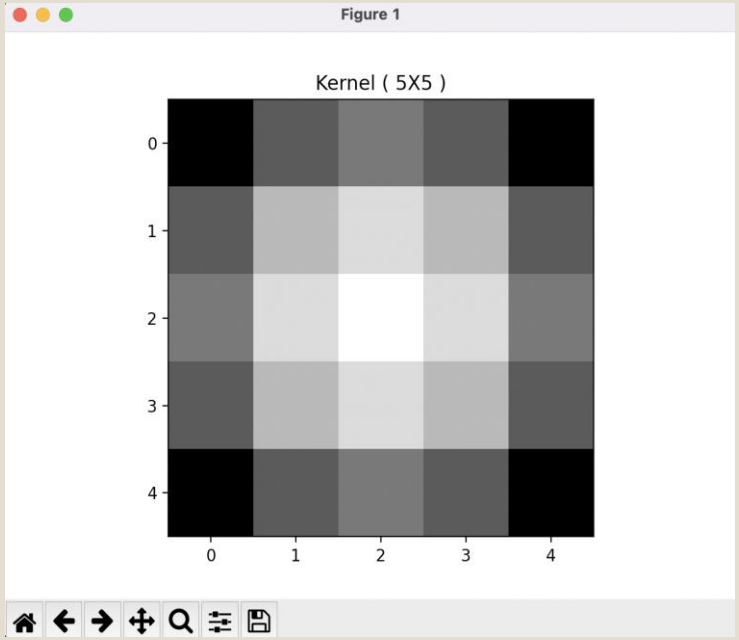
->Implementation of a Gaussian filter/kernel to smooth/blur an image .

->Typically to reduce image noise and reduce detail using the Gaussian function.

->We implement it in 2 parts:

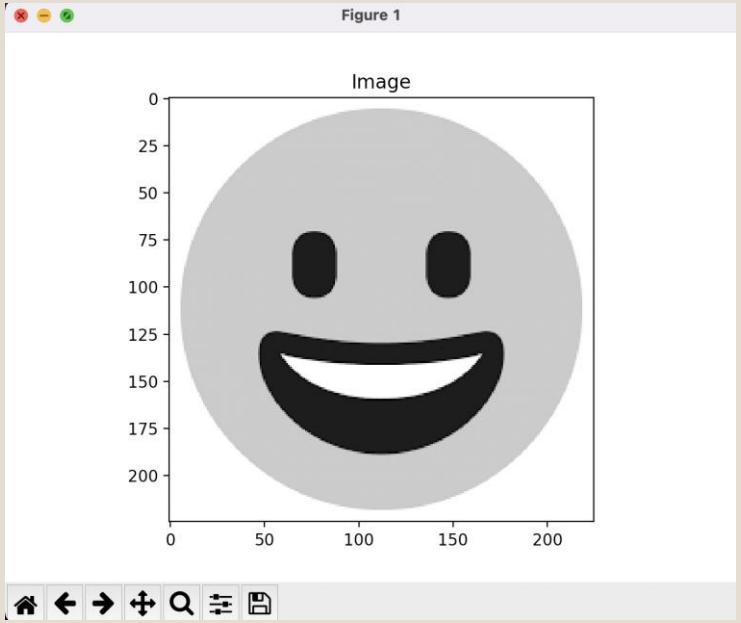
- Creating a Gaussian Kernel that is a mask or a small matrix for performing operations such as blurring in this case.
- Performing Convolution i.e. transforming an image by applying a kernel over each pixel and its local neighbors across the entire image.

KERNEL
GRAYSCALE



(KERNEL 5X5)

IMAGE

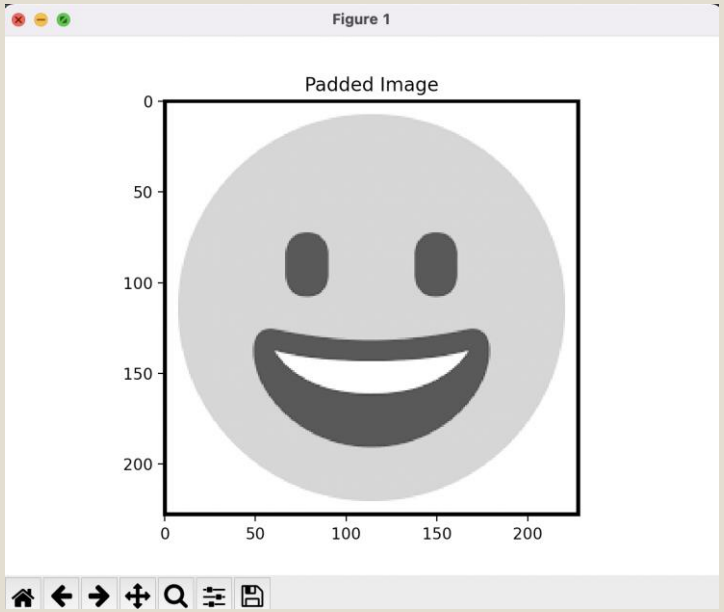


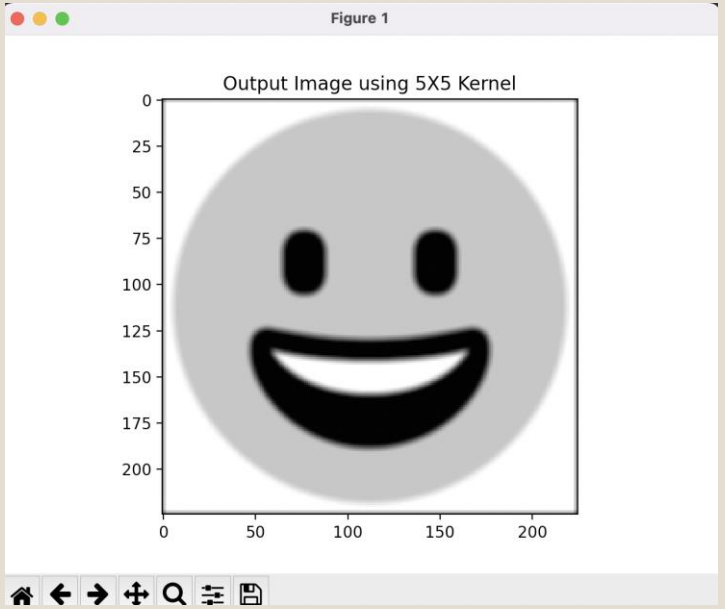
(BGR TO GRAY AS CONVOLUTION TAKES ONLY ONE CHANNEL)

IMAGE

PADDED IMAGE

FINAL





(USES ZERO PADDING:adding rectangular strips of zeros outside the rectangular edge

(USING 5X5 KERNEL)

MEDIAN FILTER

MEDIAN FILTER

The **Median Filter in Image Processing** is normally used to reduce noise in an image, somewhat like the mean filter. However, it often does a better job than the mean filter of preserving useful detail in the image.

Median Filter is a simple and powerful non-linear filter.

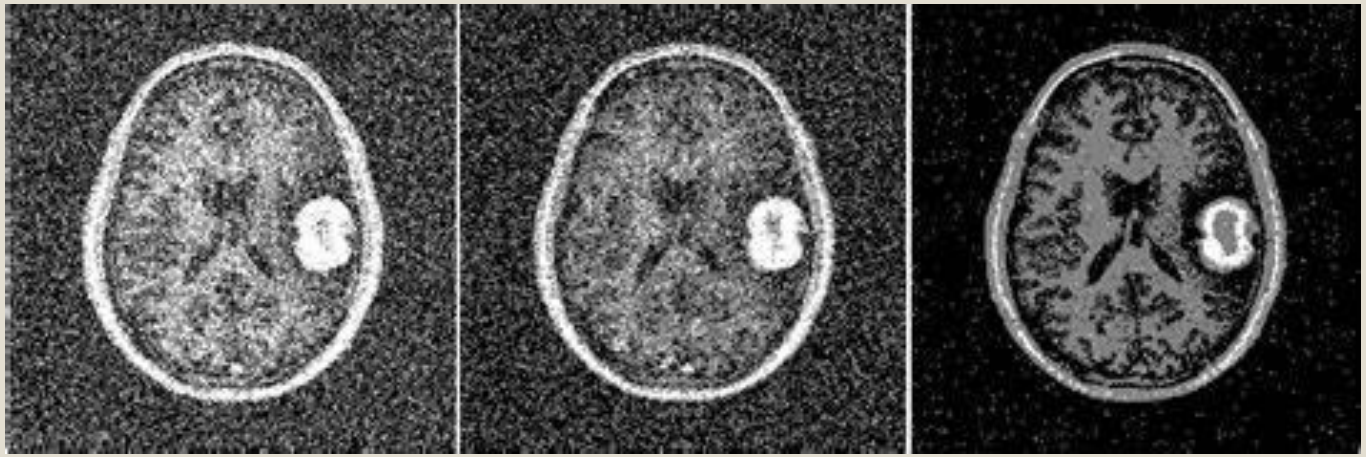
- It is used for reducing the amount of intensity variation between one pixel and the other pixel.
- In this filter, we replace pixel value with the median value.
- The median is calculated by first sorting all the pixel values into ascending order and then replace the pixel being calculated with the middle pixel value
- Reduces salt and pepper noise

Working of Median Filter:

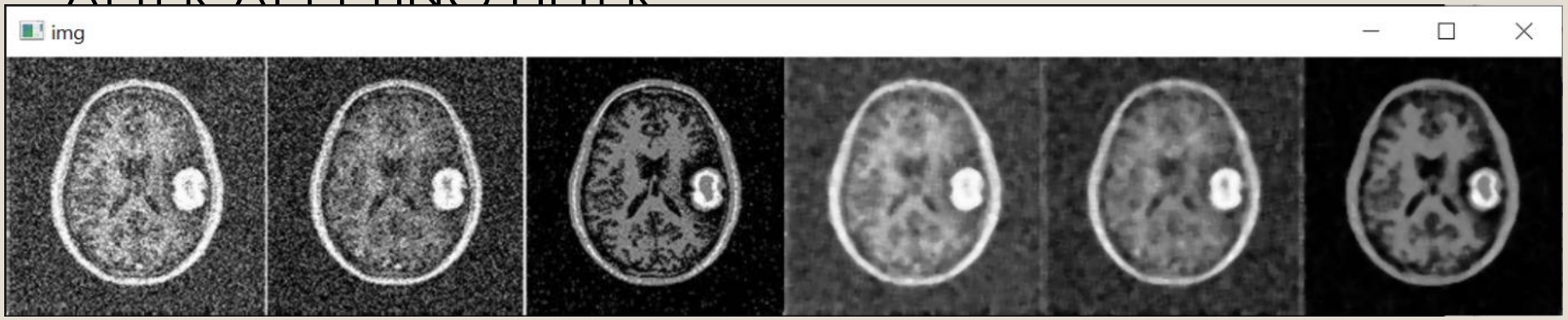
The median filter considers each pixel in the image in turn and looks at its nearby neighbour to decide whether or not it is representative of its surroundings. Replacing the pixel value with the ***median*** of those values.

The median is calculated by first sorting all the pixel values from the surrounding neighborhood into numerical order and then replacing the pixel being considered with the middle pixel value. (If the neighborhood under consideration contains an even number of pixels, the average of the two middle pixel values is used)

BEFORE APPLYING FILTER:



AFTER APPLYING FILTER:



BEFORE

AFTER

Scaling Transformations:

The scale operator performs a geometric transformation which can be used to shrink or zoom the size of an image (or part of an image). Image reduction, commonly known as *subsampling*, is performed by replacement (of a group of by pixel values one arbitrarily chosen pixel value from within this group) or by *interpolating* between pixel values in a local neighborhoods.

Image zooming is achieved by *pixel replication* or by interpolation. Scaling is used to change the visual appearance of an image, to alter the quantity of information stored in a scene representation, or as a low-level preprocessor in multi-stage image processing chain which operates on features of a particular scale. Scaling is a special case of **affine transformation.**

BEFORE

AFTER

THANK YOU