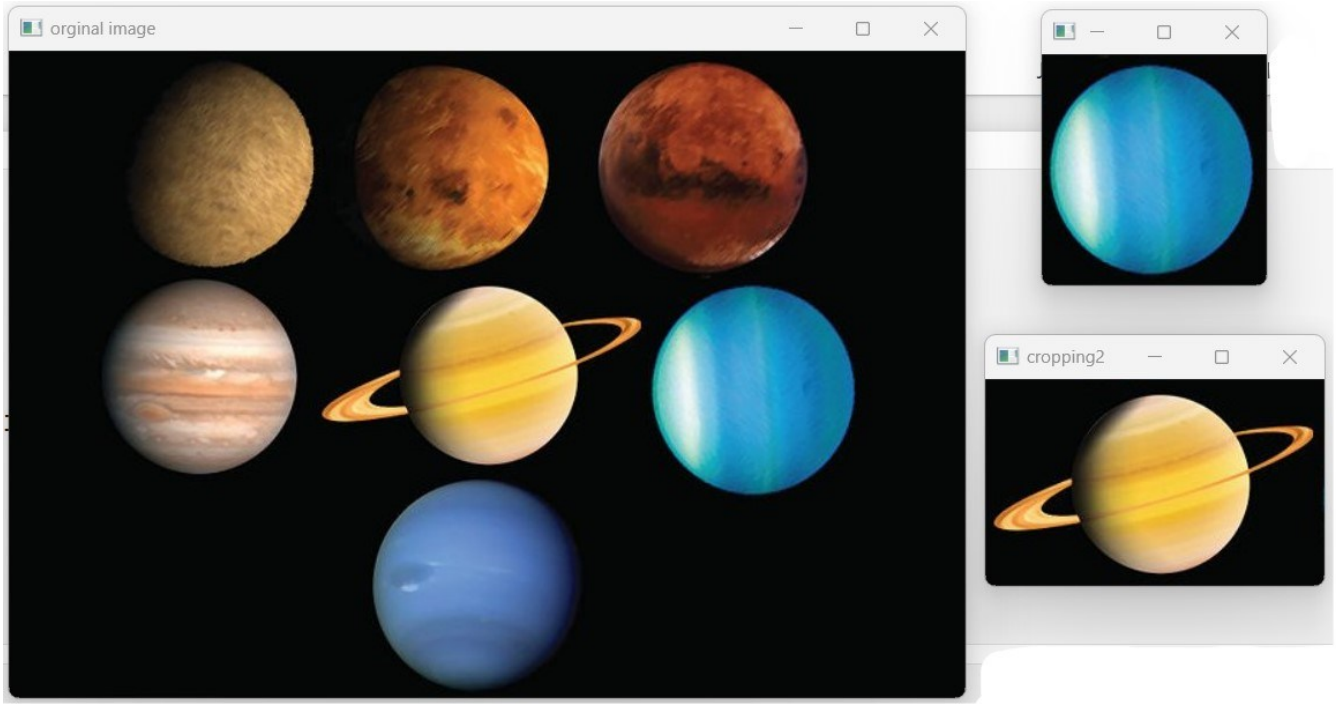
Region of Interest Image Geometry (ROI)

1. Cropping

```
import cv2 as c
#Crop an image
img=c.imread('images/planet_glow.jpg')
startRow=155
endRow=315
startCol=440
endCol=596
ROI=img[startRow:endRow,startCol:endCol]
ROI2=img[152:295,211:446]
c.imshow('orginal image',img)
c.imshow('cropping',ROI)
c.imshow('cropping2',ROI2)
c.waitKey()
c.destroyAllWindows()
```



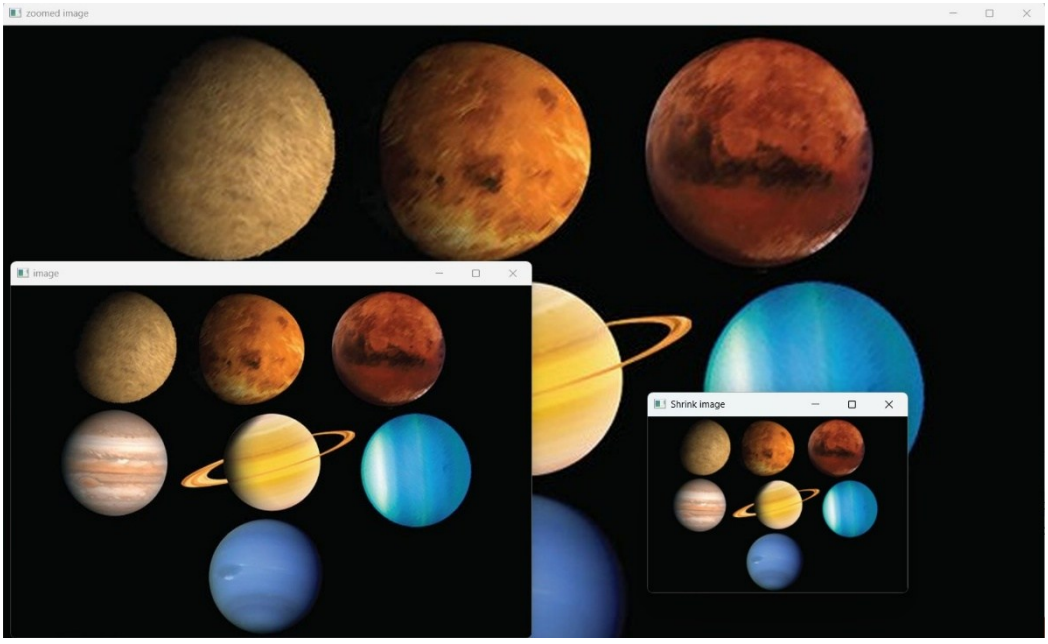
2. Image Resizing (Enlarge, Shrink)

Scaling operations increase or reduce the size of an image.

- **The cv2.resize() function** is used to resize a python image in OpenCV. It takes the following arguments:

```
cv2.resize(src, dsize,interpolation)
Here,
src          :The image to be resized.
dsize        :The desired width and height of the resized image.
interpolation:The interpolation method to be used.
```

- When the python image is resized, the **interpolation** method defines how the new pixels are computed. There are several interpolation techniques, each of which has its own quality vs. speed trade-offs.
- **It is important to note that resizing an image can reduce its quality**. This is because the new pixels are calculated by interpolating between the existing pixels, and this can introduce some blurring.



```python
import cv2
image = cv2.imread('images/planet_glow.jpg')
# Define the scale factor
scale_factor_1 = 2.0  # Increase the size by 2 times
scale_factor_2 = 1/2.0 # Decrease the size by 2 times
# Get the original image dimensions
height, width = image.shape[:2]
# Calculate the new image dimensions
new_height = int(height * scale_factor_1)
new_width = int(width * scale_factor_1)
# Resize the image (Enlarge)
zoomed_image = cv2.resize(src =image,dsize=(new_width, new_height),
interpolation=cv2.INTER_CUBIC)
# Calculate the new image dimensions
new_height1 = int(height * scale_factor_2)
new_width1 = int(width * scale_factor_2)
# Shrink image(zooming-out)
Shrink_image = cv2.resize(src= image,dsize =(new_width1, new_height1),
interpolation=cv2.INTER_AREA)
cv2.imshow('image',image)
cv2.imshow('zoomed image',zoomed_image)
cv2.imshow('Shrink image',Shrink_image)
cv2.waitKey()
cv2.destroyAllWindows()
```

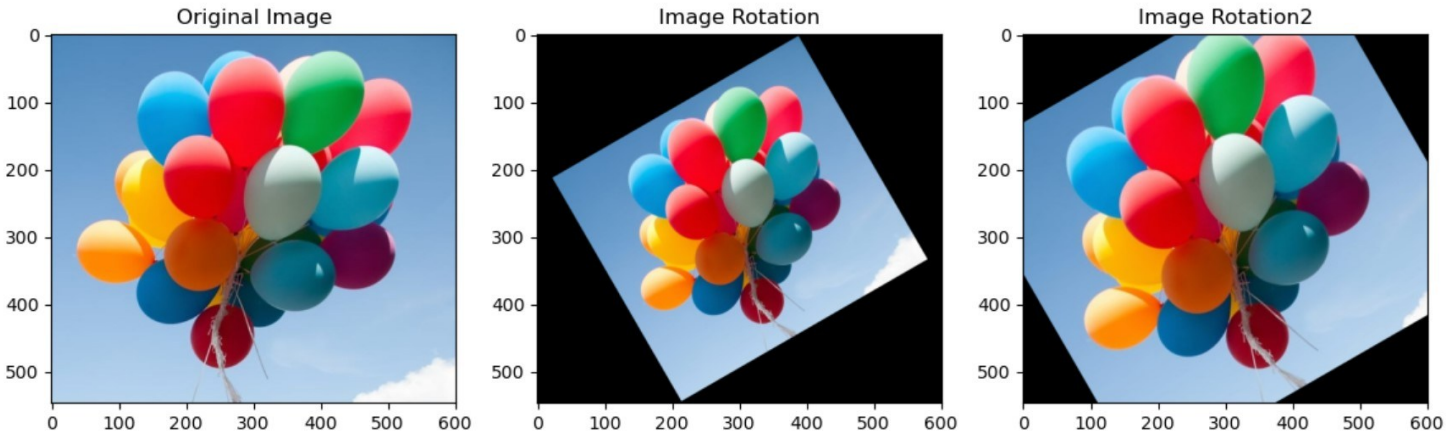
3. Image Rotation

Images can be rotated to any degree clockwise or otherwise. We just need to define rotation matrix listing rotation point, degree of rotation and the scaling factor.

- **The cv2.getRotationMatrix2D() function** is used to create a rotation matrix for an image. It takes the following arguments:
    - The center of rotation for the image.
    - The angle of rotation in degrees.
    - The scale factor.

- **The cv2.warpAffine() function** is used to apply a transformation matrix to an image. It takes the following arguments:
    - The python image to be transformed.
    - The transformation matrix.
    - The output image size.
- **The rotation angle can be positive or negative**. A positive angle rotates the image clockwise, while a negative angle rotates the image counterclockwise.
- **The scale factor can be used to scale the image up or down**. A scale factor of 1 will keep the image the same size, while a scale factor of 2 will double the size of the python image.

```
import cv2
import matplotlib.pyplot as plt
img = cv2.imread('images/balloons.jpg')
# Convert BGR image to RGB
image_rgb = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
# Image rotation parameter
center = (image_rgb.shape[1] // 2, image_rgb.shape[0] // 2)
# getRotationMatrix2D creates a matrix needed for transformation.
rotation_matrix = cv2.getRotationMatrix2D(center, 30, 0.7)
rotation_matrix2 = cv2.getRotationMatrix2D(center, 30, 1)
# We want matrix for rotation w.r.t center to 30 degree without scaling.
rotated_image = cv2.warpAffine(image_rgb, rotation_matrix, (img.shape[1], img.shape[0]))
rotated_image2 = cv2.warpAffine(image_rgb, rotation_matrix2, (img.shape[1], img.shape[0]))
# Create subplots
fig, axs = plt.subplots(1,3, figsize=(14,4))
axs[0].imshow(image_rgb)# Plot the original image
axs[0].set_title('Original Image')
axs[1].imshow(rotated_image)# Plot the Rotated image
axs[1].set_title('Image Rotation')
axs[2].imshow(rotated_image2)
axs[2].set_title('Image Rotation2')
plt.show()
```



4. Image Translation

Translating an image means shifting it within a given frame of reference that can be along the x-axis and y-axis.

- **To translate an image using OpenCV**, we need to create a transformation matrix. This matrix is a 2×3 matrix that specifies the amount of translation in each direction.
- **The cv2.warpAffine() function** is used to apply a transformation matrix to an image. It takes the following arguments:
    - The image to be transformed.
    - The transformation matrix.
    - The output image size.
- **The translation parameters** are specified in the transformation matrix as the tx and ty elements. The tx element specifies the amount of translation in the x-axis, while the ty element specifies the amount of translation in the y-axis.

```python
import cv2
import matplotlib.pyplot as plt
img = cv2.imread('images/balloons.jpg')
# Convert BGR image to RGB
image_rgb = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
width = image_rgb.shape[1]
height = image_rgb.shape[0]
tx = 100
ty = 70
# Translation matrix
translation_matrix = np.array([[1, 0, tx], [0, 1, ty]], dtype=np.float32)
# warpAffine does appropriate shifting given the Translation matrix.
translated_image = cv2.warpAffine(image_rgb, translation_matrix, (width, height))
fig, axs = plt.subplots(1, 2, figsize=(7, 4))# Create subplots
axs[0].imshow(image_rgb)
axs[0].set_title('Original Image')
# Plot the transalted image
axs[1].imshow(translated_image)
axs[1].set_title('Image Translation')
plt.show()
```