**2. To perform Image Transformations using OpenCV in Python**

**Image Transformations** in computer vision refer to techniques used to **modify or manipulate an image** in terms of its **geometry, orientation, size, or pixel values** without altering its essential content.

These transformations are crucial for tasks like **image preprocessing, data augmentation, image registration, or computer vision model training**.

**🔄 Types of Image Transformations:**

**1. Geometric Transformations**

Change the spatial relationship of pixels.

* **Translation** – Moves the image in x/y direction.
* **Rotation** – Rotates the image by a certain angle.
* **Scaling** – Resizes the image (zoom in/out).
* **Shearing** – Tilts the image.
* **Flipping** – Mirror the image horizontally or vertically.

Here's a step-by-step guide with **examples** for each:

**✅ 1. Import Required Libraries**

import cv2

import numpy as np

**📷 2. Read the Image**

img = cv2.imread('your\_image.jpg')

cv2.imshow('Original Image', img)

cv2.waitKey(0)

cv2.destroyAllWindows()

**🔀 3. Translation (Moving the image)**

# Move image by 100px right and 50px down

rows, cols = img.shape[:2]

M = np.float32([[1, 0, 100], [0, 1, 50]])

translated\_img = cv2.warpAffine(img, M, (cols, rows))

cv2.imshow('Translated Image', translated\_img)

cv2.waitKey(0)

cv2.destroyAllWindows()

**🔄 4. Rotation**

# Rotate by 45 degrees around the center

center = (cols // 2, rows // 2)

M = cv2.getRotationMatrix2D(center, 45, 1.0)

rotated\_img = cv2.warpAffine(img, M, (cols, rows))

cv2.imshow('Rotated Image', rotated\_img)

cv2.waitKey(0)

cv2.destroyAllWindows()

**🔍 5. Resizing**

resized\_img = cv2.resize(img, (300, 300), interpolation=cv2.INTER\_LINEAR)

cv2.imshow('Resized Image', resized\_img)

cv2.waitKey(0)

cv2.destroyAllWindows()

**↔️ 6. Flipping**

# Flip vertically (0), horizontally (1), both (-1)

flipped\_img = cv2.flip(img, 1)

cv2.imshow('Flipped Image', flipped\_img)

cv2.waitKey(0)

cv2.destroyAllWindows()

**🧭 7. Affine Transformation**

pts1 = np.float32([[50, 50], [200, 50], [50, 200]])

pts2 = np.float32([[10, 100], [200, 50], [100, 250]])

M = cv2.getAffineTransform(pts1, pts2)

affine\_img = cv2.warpAffine(img, M, (cols, rows))

cv2.imshow('Affine Transform', affine\_img)

cv2.waitKey(0)

cv2.destroyAllWindows()

**🔳 8. Perspective Transformation**

pts1 = np.float32([[56,65], [368,52], [28,387], [389,390]])

pts2 = np.float32([[0,0], [300,0], [0,300], [300,300]])

M = cv2.getPerspectiveTransform(pts1, pts2)

perspective\_img = cv2.warpPerspective(img, M, (300, 300))

cv2.imshow('Perspective Transform', perspective\_img)

cv2.waitKey(0)

cv2.destroyAllWindows()

**💡 Notes:**

* Always call cv2.waitKey(0) and cv2.destroyAllWindows() after displaying an image.
* Replace 'your\_image.jpg' with the actual path to your image.

Would you like this in **.py file format** or need **plots using matplotlib** as well?

-----------------------------------------------------------------------------------------

1. **Translation (Moving Image) in OpenCV**

**Translation** means shifting an image along the X and/or Y axis. In computer vision, translation is commonly used for image augmentation or shifting focus within an image.

**🧠 Concept**

We use a **transformation matrix** to move the image:

Translation Matrix=[10tx01ty]\text{Translation Matrix} = \begin{bmatrix} 1 & 0 & t\_x \\ 0 & 1 & t\_y \end{bmatrix}

Where:

* t\_x = shift along X-axis
* t\_y = shift along Y-axis

**🧾 Python Code using OpenCV**

import cv2

import numpy as np

# Load the image

image = cv2.imread('your\_image.jpg')

# Define translation values

tx = 100 # Shift 100 pixels to the right

ty = 50 # Shift 50 pixels down

# Create the translation matrix

M = np.float32([[1, 0, tx],[0, 1, ty]])

# Apply the translation

translated\_image = cv2.warpAffine(image, M, (image.shape[1], image.shape[0]))

# Display the result

cv2.imshow('Original Image', image)

cv2.imshow('Translated Image', translated\_image)

cv2.waitKey(0)

cv2.destroyAllWindows()

**🔍 Important Functions Used**

* cv2.imread() – Reads an image.
* np.float32() – Defines the matrix with 32-bit float values.
* cv2.warpAffine() – Applies the affine transformation.
* cv2.imshow() – Displays the image.

2. **🔄 Rotation in OpenCV (Python) with Code**

In OpenCV, rotating an image involves using an **affine transformation matrix**. This is commonly done using cv2.getRotationMatrix2D() and cv2.warpAffine().

**✅ Code to Rotate an Image**

import cv2

import numpy as np

# Load the image

image = cv2.imread('your\_image.jpg')

# Get the image dimensions

(h, w) = image.shape[:2]

# Define the center of the image (around which rotation happens)

center = (w // 2, h // 2)

# Define the angle of rotation (positive = counterclockwise)

angle = 45 # rotate by 45 degrees

# Define the scale of the image

scale = 1.0 # keep original size

# Get the rotation matrix

rotation\_matrix = cv2.getRotationMatrix2D(center, angle, scale)

# Perform the rotation using warpAffine

rotated\_image = cv2.warpAffine(image, rotation\_matrix, (w, h))

# Display the rotated image

cv2.imshow('Rotated Image', rotated\_image)

cv2.waitKey(0)

cv2.destroyAllWindows()

**🧠 Explanation**

* cv2.getRotationMatrix2D(center, angle, scale):
  + Returns a 2x3 affine transform matrix for rotation.
* cv2.warpAffine(image, matrix, (w, h)):

Let me know if you want this in a Jupyter notebook, with sample images, or rotation via a GUI slider.

**3. Image Resizing-**

3. In OpenCV, **resizing** an image refers to changing its dimensions (width and height) using the cv2.resize() function.

**✅ Syntax:**

resized\_image = cv2.resize(src, dsize, interpolation)

* src: Original image.
* dsize: Desired size in the form (width, height).
* interpolation: (Optional) Interpolation method like cv2.INTER\_LINEAR, cv2.INTER\_AREA, etc.

**📌 Common Interpolation Methods:**

* cv2.INTER\_LINEAR: Default, good for upscaling.
* cv2.INTER\_AREA: Preferred for downscaling.
* cv2.INTER\_NEAREST: Fastest, lower quality.

**✅ Example Code:**

import cv2

# Load the original image

image = cv2.imread('example.jpg')

# Resize to a specific width and height

resized = cv2.resize(image, (300, 200)) # width=300, height=200

# Resize by scale (e.g., half the size)

resized\_half = cv2.resize(image, (0, 0), fx=0.5, fy=0.5)

# Show images

cv2.imshow('Original', image)

cv2.imshow('Resized (300x200)', resized)

cv2.imshow('Resized Half', resized\_half)

cv2.waitKey(0)

cv2.destroyAllWindows()

**📝 Notes:**

* If you're using **scaling factors** (fx, fy), set dsize to (0, 0).
* Always use cv2.waitKey() and cv2.destroyAllWindows() to display and then close image windows properly.

Let me know if you want to resize using aspect ratio or with GUI sliders.