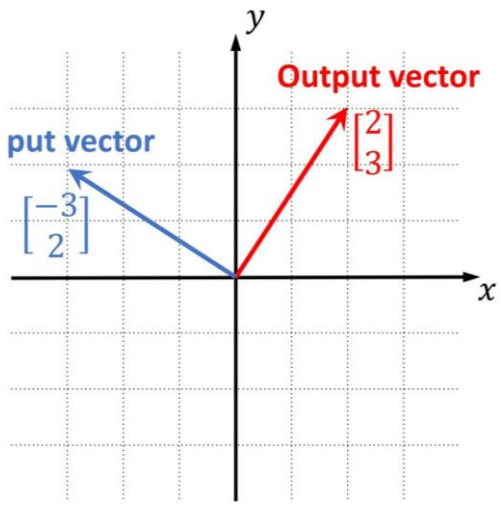
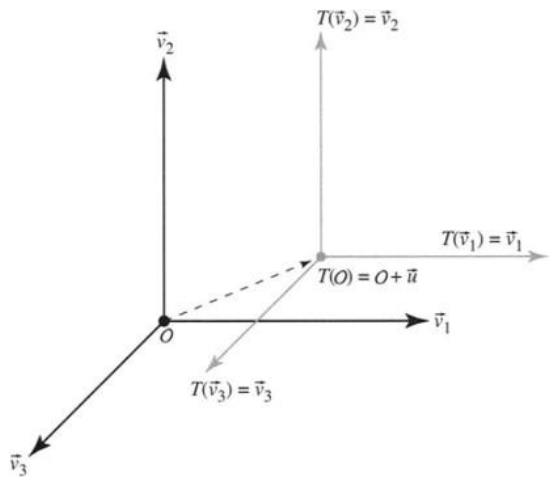
Computer Vision

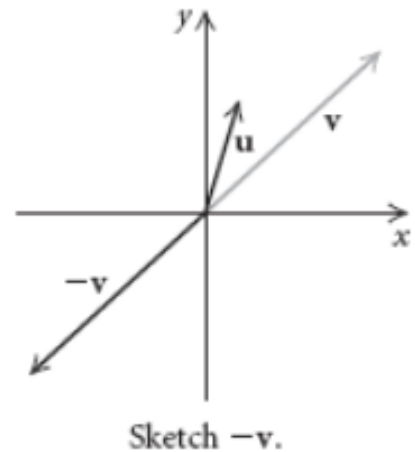
CVI620

Session 7
01/2025



$$Ax = \begin{bmatrix} 6 & 2 & 4 \\ -1 & 4 & 3 \\ -2 & 9 & 3 \end{bmatrix} \times \begin{bmatrix} 4 \\ -2 \\ 1 \end{bmatrix} = \begin{bmatrix} 24 \\ -9 \\ 23 \end{bmatrix}$$





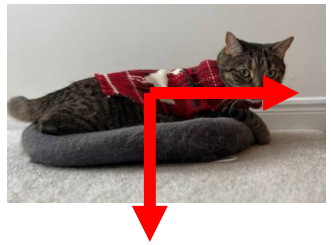
Sketch $-\mathbf{v}$.

2D Transformations

Translation

pixels move in the same direction

$$u = x + tx$$
$$v = y + ty$$



Scale or resize

$$u = x * sx$$
$$v = y * sy$$

Rotation

$$u = x * \cos\theta - y * \sin\theta$$
$$v = y * \sin\theta + x * \cos\theta$$

Rotate

- Rotating an image around a specific point (usually the center) by a given angle
- Rotation in OpenCV is achieved using transformation matrices
- cv2.getRotationMatrix2D(center, angle, scale) generates the rotation matrix

Rotate

- The standard 2D rotation matrix for counterclockwise rotation is:

$$R = \begin{bmatrix} \cos\theta & -\sin\theta & (1-\cos\theta \cdot s) \cdot cx + \sin\theta \cdot s \cdot cy \text{ or } 0 \\ \sin\theta & \cos\theta & (1-\cos\theta \cdot s) \cdot cy - \sin\theta \cdot s \cdot cx \text{ or } 0 \end{bmatrix}$$

- OpenCV uses a clockwise rotation by default, so it flips the sign of

$$R = \begin{bmatrix} \cos\theta & \sin\theta & (1-\cos\theta \cdot s) \cdot cx - \sin\theta \cdot s \cdot cy \\ -\sin\theta & \cos\theta & \sin\theta \cdot cx + (1-\cos\theta \cdot s) \cdot s \cdot cy \end{bmatrix}$$

- The third column in OpenCV's 2D rotation matrix represents translation and adjusts the rotated image's position to keep it properly aligned within the output frame.

Rotate

```python
import cv2
import numpy as np

image = cv2.imread("Lucy.jpg")

# dimensions of the image
(h, w) = image.shape[:2]

# center of the image
center = (w // 2, h // 2)

# rotation angle in degrees
angle = 45

# the scale (1.0 means no scaling)
scale = 1.0

# rotation matrix
rotation_matrix = cv2.getRotationMatrix2D(center, angle, scale)

# rotate
rotated_image = cv2.warpAffine(image, rotation_matrix, (w, h))

cv2.imshow("Rotated Image", rotated_image)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Shear

```python
import cv2
import numpy as np

image = cv2.imread("image.jpg")

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

# shear factor
shear_factor_x = 0.5  # horizontal shear
shear_factor_y = 0.2  # vertical shear

# shear matrix
shear_matrix = np.array([
    [1, shear_factor_x, 0],
    [shear_factor_y, 1, 0]
], dtype=np.float32)

# the new width and height after shearing
new_w = int(w + abs(shear_factor_x * h))
new_h = int(h + abs(shear_factor_y * w))

# shear
sheared_image = cv2.warpAffine(image, shear_matrix, (new_w, new_h))

cv2.imshow("Sheared Image", sheared_image)
cv2.waitKey(0)
cv2.destroyAllWindows()
```