Experiments 1 to 35 — Line-by-line Python + OpenCV Code

Each experiment is a complete, runnable Python snippet.
Update file paths (e.g., 'input.jpg', 'input.mp4') as needed.

Experiment 1

```
# Experiment 1 - Read image and convert to grayscale
import cv2

img = cv2.imread('input.jpg')              # read image (BGR)
if img is None:
   raise FileNotFoundError('input.jpg not found')
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY) # convert to grayscale
cv2.imshow('Gray', gray)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Experiment 2

```
# Experiment 2 - Read image and apply Gaussian Blur
import cv2

img = cv2.imread('input.jpg')
if img is None:
   raise FileNotFoundError('input.jpg not found')
blur = cv2.GaussianBlur(img, (7,7), 1.5)     # kernel 7x7, sigma 1.5
cv2.imshow('Blur', blur)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Experiment 3

```
# Experiment 3 - Read image and detect edges with Canny
import cv2

img = cv2.imread('input.jpg')
if img is None:
   raise FileNotFoundError('input.jpg not found')
```

```
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
edges = cv2.Canny(gray, 50, 150)          # low and high thresholds
cv2.imshow('Edges', edges)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Experiment 4

```
# Experiment 4 - Dilate an image
import cv2
import numpy as np

img = cv2.imread('input.jpg')
if img is None:
    raise FileNotFoundError('input.jpg not found')
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
kernel = np.ones((5,5), dtype=np.uint8)
dilated = cv2.dilate(gray, kernel, iterations=1)
cv2.imshow('Dilated', dilated)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Experiment 5

```
# Experiment 5 - Erode an image
import cv2
import numpy as np

img = cv2.imread('input.jpg')
if img is None:
    raise FileNotFoundError('input.jpg not found')
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
kernel = np.ones((5,5), dtype=np.uint8)
eroded = cv2.erode(gray, kernel, iterations=1)
cv2.imshow('Eroded', eroded)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Experiment 6

```
# Experiment 6 - Play a video file (normal, slow, fast)
```

```
import cv2
cap = cv2.VideoCapture('input.mp4')
if not cap.isOpened():
    raise FileNotFoundError('input.mp4 not found')
# normal playback
while True:
    ret, frame = cap.read()
    if not ret:
        break
    cv2.imshow('Video - Normal', frame)
    if cv2.waitKey(30) & 0xFF == ord('q'):
        break
cap.release()
cv2.destroyAllWindows()

# For slow: reduce waitKey to larger value (e.g., 100)
# For fast: reduce waitKey to smaller value (e.g., 10)
```

Experiment 7

```
# Experiment 7 - Capture from webcam and display (slow and fast)
import cv2
cap = cv2.VideoCapture(0)
if not cap.isOpened():
    raise RuntimeError('Cannot open webcam')
while True:
    ret, frame = cap.read()
    if not ret:
        break
    cv2.imshow('Webcam', frame)
    if cv2.waitKey(30) & 0xFF == ord('q'):
        break
cap.release()
cv2.destroyAllWindows()
```

Experiment 8

```
# Experiment 8 - Scale image up and down (resize)
import cv2

img = cv2.imread('input.jpg')
```

```
if img is None:
    raise FileNotFoundError('input.jpg not found')
h,w = img.shape[:2]
up = cv2.resize(img, (w*2, h*2), interpolation=cv2.INTER_LINEAR)
down = cv2.resize(img, (w//2, h//2), interpolation=cv2.INTER_AREA)
cv2.imshow('Up', up)
cv2.imshow('Down', down)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Experiment 9

```
# Experiment 9 - Rotate image clockwise and counter-clockwise
import cv2

img = cv2.imread('input.jpg')
if img is None:
    raise FileNotFoundError('input.jpg not found')
h,w = img.shape[:2]
M_cw = cv2.getRotationMatrix2D((w/2, h/2), -90, 1.0)  # clockwise 90
rot_cw = cv2.warpAffine(img, M_cw, (w, h))
M_ccw = cv2.getRotationMatrix2D((w/2, h/2), 90, 1.0)  # counter-clockwise 90
rot_ccw = cv2.warpAffine(img, M_ccw, (w, h))
cv2.imshow('CW', rot_cw)
cv2.imshow('CCW', rot_ccw)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Experiment 10

```
# Experiment 10 - Translate (move) image
import cv2
import numpy as np

img = cv2.imread('input.jpg')
if img is None:
    raise FileNotFoundError('input.jpg not found')
h,w = img.shape[:2]
M = np.float32([[1, 0, 50], [0, 1, 100]])   # shift x=50, y=100
translated = cv2.warpAffine(img, M, (w, h))
cv2.imshow('Translated', translated)
```

```
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Experiment 11

```
# Experiment 11 - Affine transformation
import cv2
import numpy as np

img = cv2.imread('input.jpg')
if img is None:
    raise FileNotFoundError('input.jpg not found')
h,w = img.shape[:2]
pts1 = np.float32([[50,50],[200,50],[50,200]])
pts2 = np.float32([[10,100],[200,50],[100,250]])
M = cv2.getAffineTransform(pts1, pts2)
affine = cv2.warpAffine(img, M, (w, h))
cv2.imshow('Affine', affine)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Experiment 12

```
# Experiment 12 - Perspective transform (image)
import cv2
import numpy as np

img = cv2.imread('input.jpg')
if img is None:
    raise FileNotFoundError('input.jpg not found')
h,w = img.shape[:2]
src = np.float32([[0,0],[w-1,0],[w-1,h-1],[0,h-1]])
dst = np.float32([[0,0],[w-1,0],[int(0.6*w),h-1],[int(0.4*w),h-1]])
M = cv2.getPerspectiveTransform(src, dst)
pers = cv2.warpPerspective(img, M, (w, h))
cv2.imshow('Perspective', pers)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Experiment 13

```
# Experiment 13 - Perspective transform on video frames
import cv2
import numpy as np

cap = cv2.VideoCapture('input.mp4')
if not cap.isOpened():
    raise FileNotFoundError('input.mp4 not found')
ret, frame = cap.read()
if not ret:
    raise RuntimeError('Empty video')
h,w = frame.shape[:2]
src = np.float32([[0,0],[w-1,0],[w-1,h-1],[0,h-1]])
dst = np.float32([[0,0],[w-1,0],[int(0.8*w),h-1],[int(0.2*w),h-1]])
M = cv2.getPerspectiveTransform(src, dst)
cap.set(cv2.CAP_PROP_POS_FRAMES, 0)
while True:
    ret, frame = cap.read()
    if not ret:
        break
    out = cv2.warpPerspective(frame, M, (w,h))
    cv2.imshow('Persp Video', out)
    if cv2.waitKey(25) & 0xFF == ord('q'):
        break
cap.release()
cv2.destroyAllWindows()
```

Experiment 14

```
# Experiment 14 - Homography (findHomography + warp)
import cv2
import numpy as np

img = cv2.imread('input.jpg')
img2 = cv2.imread('input2.jpg')
if img is None or img2 is None:
    raise FileNotFoundError('input images not found')
# Manually selected corresponding points (example)
pts1 = np.float32([[10,10],[200,50],[50,200],[220,220]])
pts2 = np.float32([[0,0],[300,30],[30,300],[300,300]])
H, mask = cv2.findHomography(pts1, pts2, cv2.RANSAC, 5.0)
```

```
h,w = img2.shape[:2]
warped = cv2.warpPerspective(img, H, (w, h))
cv2.imshow('Warped to image2', warped)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Experiment 15

```
# Experiment 15 - Direct Linear Transform (DLT) example (same as homography)
# This demonstrates solving for H using cv2.findHomography which uses DLT
internally.
import cv2
import numpy as np

# (See Experiment 14 for a runnable example)
```

Experiment 16

```
# Experiment 16 - Edge detection using Sobel X
import cv2
import numpy as np

img = cv2.imread('input.jpg', cv2.IMREAD_GRAYSCALE)
if img is None:
    raise FileNotFoundError('input.jpg not found')
sobelx = cv2.Sobel(img, cv2.CV_64F, 1, 0, ksize=3)
abs_sobelx = cv2.convertScaleAbs(sobelx)
cv2.imshow('Sobel X', abs_sobelx)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Experiment 17

```
# Experiment 17 - Edge detection using Sobel Y
import cv2
import numpy as np

img = cv2.imread('input.jpg', cv2.IMREAD_GRAYSCALE)
if img is None:
    raise FileNotFoundError('input.jpg not found')
sobely = cv2.Sobel(img, cv2.CV_64F, 0, 1, ksize=3)
```

```
abs_sobely = cv2.convertScaleAbs(sobely)
cv2.imshow('Sobel Y', abs_sobely)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Experiment 18

```
# Experiment 18 - Edge detection using Sobel XY
import cv2
import numpy as np

img = cv2.imread('input.jpg', cv2.IMREAD_GRAYSCALE)
if img is None:
    raise FileNotFoundError('input.jpg not found')
sobelxy = cv2.Sobel(img, cv2.CV_64F, 1, 1, ksize=3)
abs_sobelxy = cv2.convertScaleAbs(sobelxy)
cv2.imshow('Sobel XY', abs_sobelxy)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Experiment 19

```
# Experiment 19 - Sharpen using Laplacian mask (negative center)
import cv2
import numpy as np

img = cv2.imread('input.jpg')
if img is None:
    raise FileNotFoundError('input.jpg not found')
kernel = np.array([[ -1, -1, -1],
                   [ -1,  9, -1],
                   [ -1, -1, -1]])
sharpened = cv2.filter2D(img, -1, kernel)
cv2.imshow('Sharpened', sharpened)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Experiment 20

```
# Experiment 20 - Sharpen using unsharp masking
import cv2
```

```
import numpy as np

img = cv2.imread('input.jpg').astype('float32')
if img is None:
    raise FileNotFoundError('input.jpg not found')
blur = cv2.GaussianBlur(img, (9,9), 10.0)
unsharp = cv2.addWeighted(img, 1.5, blur, -0.5, 0)
unsharp = cv2.convertScaleAbs(unsharp)
cv2.imshow('Unsharp', unsharp)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Experiment 21

```
# Experiment 21 - High-boost filtering
import cv2
import numpy as np

img = cv2.imread('input.jpg').astype('float32')
if img is None:
    raise FileNotFoundError('input.jpg not found')
blur = cv2.GaussianBlur(img, (9,9), 10.0)
A = 1.8
high_boost = cv2.addWeighted(img, A, blur, -(A-1), 0)
high_boost = cv2.convertScaleAbs(high_boost)
cv2.imshow('High Boost', high_boost)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Experiment 22

```
# Experiment 22 - Insert watermark (logo) with transparency
import cv2
import numpy as np

img = cv2.imread('input.jpg')
logo = cv2.imread('logo.png', cv2.IMREAD_UNCHANGED)  # assume RGBA
if img is None or logo is None:
    raise FileNotFoundError('input.jpg or logo.png not found')
# resize logo if larger
lh, lw = logo.shape[:2]
```

```
if lw > img.shape[1]//5:
    scale = (img.shape[1]//5) / lw
    logo = cv2.resize(logo, (0,0), fx=scale, fy=scale)
# split alpha if present
if logo.shape[2] == 4:
    b,g,r,a = cv2.split(logo)
    overlay = cv2.merge((b,g,r))
    mask = a
else:
    overlay = logo
    mask = 255 * np.ones(overlay.shape[:2], dtype=np.uint8)
# place at bottom-right
x = img.shape[1] - overlay.shape[1] - 10
y = img.shape[0] - overlay.shape[0] - 10
roi = img[y:y+overlay.shape[0], x:x+overlay.shape[1]]
mask_inv = cv2.bitwise_not(mask)
bg = cv2.bitwise_and(roi, roi, mask=mask_inv)
fg = cv2.bitwise_and(overlay, overlay, mask=mask)
dst = cv2.add(bg, fg)
img[y:y+overlay.shape[0], x:x+overlay.shape[1]] = dst
cv2.imshow('Watermarked', img)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Experiment 23

```
# Experiment 23 - Cropping, copying and pasting image region
import cv2

img = cv2.imread('input.jpg')
if img is None:
    raise FileNotFoundError('input.jpg not found')
# crop region
crop = img[50:200, 100:300].copy()
# paste into new location
img[10:10+crop.shape[0], 10:10+crop.shape[1]] = crop
cv2.imshow('Copied', img)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Experiment 24

```
# Experiment 24 - Boundary detection using simple convolution kernel (edge/
outline)
import cv2
import numpy as np

img = cv2.imread('input.jpg', cv2.IMREAD_GRAYSCALE)
if img is None:
   raise FileNotFoundError('input.jpg not found')
kernel = np.array([[ -1, -1, -1],
                   [ -1,  8, -1],
                   [ -1, -1, -1]])
boundary = cv2.filter2D(img, -1, kernel)
cv2.imshow('Boundary', boundary)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Experiment 25

```
# Experiment 25 - Morphological Erosion (binary example)
import cv2
import numpy as np

img = cv2.imread('input.jpg', cv2.IMREAD_GRAYSCALE)
if img is None:
   raise FileNotFoundError('input.jpg not found')
_, thresh = cv2.threshold(img, 128, 255, cv2.THRESH_BINARY)
kernel = np.ones((5,5), np.uint8)
erosion = cv2.erode(thresh, kernel, iterations=1)
cv2.imshow('Erosion', erosion)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Experiment 26

```
# Experiment 26 - Morphological Dilation (binary example)
import cv2
import numpy as np

img = cv2.imread('input.jpg', cv2.IMREAD_GRAYSCALE)
if img is None:
```

```
    raise FileNotFoundError('input.jpg not found')
_, thresh = cv2.threshold(img, 128, 255, cv2.THRESH_BINARY)
kernel = np.ones((5,5), np.uint8)
dilation = cv2.dilate(thresh, kernel, iterations=1)
cv2.imshow('Dilation', dilation)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Experiment 27

```
# Experiment 27 - Morphological Opening
import cv2
import numpy as np

img = cv2.imread('input.jpg', cv2.IMREAD_GRAYSCALE)
if img is None:
    raise FileNotFoundError('input.jpg not found')
_, thresh = cv2.threshold(img, 128, 255, cv2.THRESH_BINARY)
kernel = np.ones((5,5), np.uint8)
opening = cv2.morphologyEx(thresh, cv2.MORPH_OPEN, kernel)
cv2.imshow('Opening', opening)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Experiment 28

```
# Experiment 28 - Morphological Closing
import cv2
import numpy as np

img = cv2.imread('input.jpg', cv2.IMREAD_GRAYSCALE)
if img is None:
    raise FileNotFoundError('input.jpg not found')
_, thresh = cv2.threshold(img, 128, 255, cv2.THRESH_BINARY)
kernel = np.ones((5,5), np.uint8)
closing = cv2.morphologyEx(thresh, cv2.MORPH_CLOSE, kernel)
cv2.imshow('Closing', closing)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Experiment 29

```
# Experiment 29 - Morphological Gradient
import cv2
import numpy as np

img = cv2.imread('input.jpg', cv2.IMREAD_GRAYSCALE)
if img is None:
    raise FileNotFoundError('input.jpg not found')
kernel = np.ones((5,5), np.uint8)
mg = cv2.morphologyEx(img, cv2.MORPH_GRADIENT, kernel)
cv2.imshow('Morphological Gradient', mg)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Experiment 30

```
# Experiment 30 - Top-hat Transform
import cv2
import numpy as np

img = cv2.imread('input.jpg', cv2.IMREAD_GRAYSCALE)
if img is None:
    raise FileNotFoundError('input.jpg not found')
kernel = np.ones((15,15), np.uint8)
tophat = cv2.morphologyEx(img, cv2.MORPH_TOPHAT, kernel)
cv2.imshow('Top-hat', tophat)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Experiment 31

```
# Experiment 31 - Recognize a watch using template matching (simple)
import cv2
import numpy as np

img = cv2.imread('scene.jpg')
template = cv2.imread('watch_template.jpg')
if img is None or template is None:
    raise FileNotFoundError('scene.jpg or watch_template.jpg not found')
res = cv2.matchTemplate(img, template, cv2.TM_CCOEFF_NORMED)
min_val, max_val, min_loc, max_loc = cv2.minMaxLoc(res)
```

```
top_left = max_loc
h, w = template.shape[:2]
bottom_right = (top_left[0] + w, top_left[1] + h)
cv2.rectangle(img, top_left, bottom_right, (0,255,0), 2)
cv2.imshow('Detected Watch', img)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Experiment 32

```
# Experiment 32 - Play video in reverse
import cv2

cap = cv2.VideoCapture('input.mp4')
if not cap.isOpened():
    raise FileNotFoundError('input.mp4 not found')
frames = []
while True:
    ret, frame = cap.read()
    if not ret:
        break
    frames.append(frame)
cap.release()
for frame in reversed(frames):
    cv2.imshow('Reverse', frame)
    if cv2.waitKey(25) & 0xFF == ord('q'):
        break
cv2.destroyAllWindows()
```

Experiment 33

```
# Experiment 33 - Face detection using Haar cascade
import cv2

face_cascade = cv2.CascadeClassifier(cv2.data.haarcascades +
'haarcascade_frontalface_default.xml')
img = cv2.imread('input.jpg')
if img is None:
    raise FileNotFoundError('input.jpg not found')
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
faces = face_cascade.detectMultiScale(gray, scaleFactor=1.1, minNeighbors=5)
```

```
for (x,y,w,h) in faces:
    cv2.rectangle(img, (x,y), (x+w, y+h), (255,0,0), 2)
cv2.imshow('Faces', img)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Experiment 34

```
# Experiment 34 - Vehicle detection using pre-trained cascade (example)
import cv2

car_cascade = cv2.CascadeClassifier('cars.xml')  # provide a cars.xml cascade file
cap = cv2.VideoCapture('traffic.mp4')
if not cap.isOpened():
    raise FileNotFoundError('traffic.mp4 not found')
while True:
    ret, frame = cap.read()
    if not ret:
        break
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
    cars = car_cascade.detectMultiScale(gray, 1.1, 3)
    for (x,y,w,h) in cars:
        cv2.rectangle(frame,(x,y),(x+w,y+h),(0,255,0),2)
    cv2.imshow('Cars', frame)
    if cv2.waitKey(25) & 0xFF == ord('q'):
        break
cap.release()
cv2.destroyAllWindows()
```

Experiment 35

```
# Experiment 35 - Draw rectangular shape and extract object
import cv2

img = cv2.imread('input.jpg')
if img is None:
    raise FileNotFoundError('input.jpg not found')
# draw rectangle and extract ROI
x, y, w, h = 100, 100, 150, 150
cv2.rectangle(img, (x,y), (x+w, y+h), (0,255,0), 2)
roi = img[y:y+h, x:x+w].copy()
```

```
cv2.imshow('Image with Rectangle', img)
cv2.imshow('Extracted ROI', roi)
cv2.waitKey(0)
cv2.destroyAllWindows()
```


Eroded Image


Grayscale Image


Dilated Image


Moved Image


Smaller Image


Original

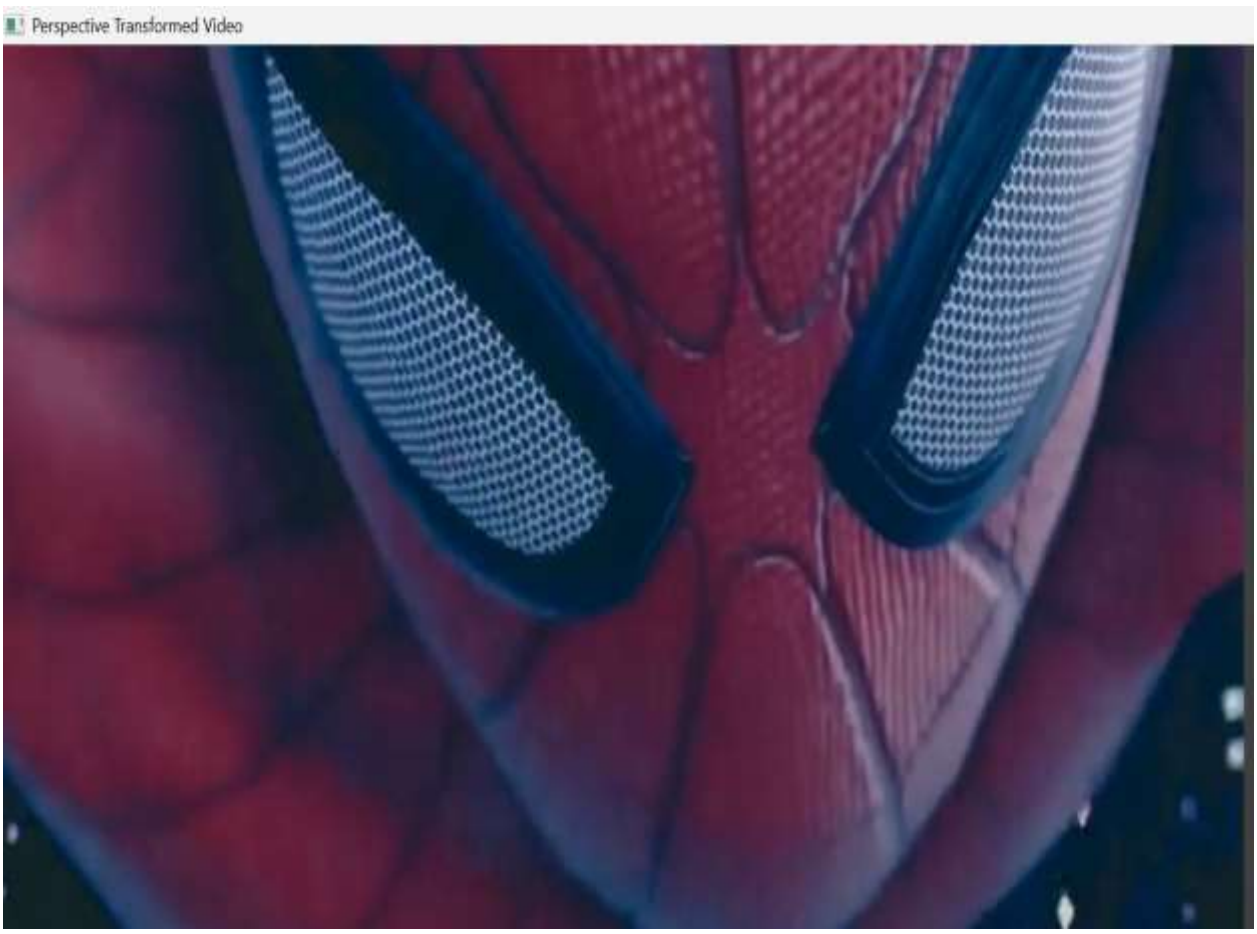

Counterclockwise Rotation


Video - Slow Motion

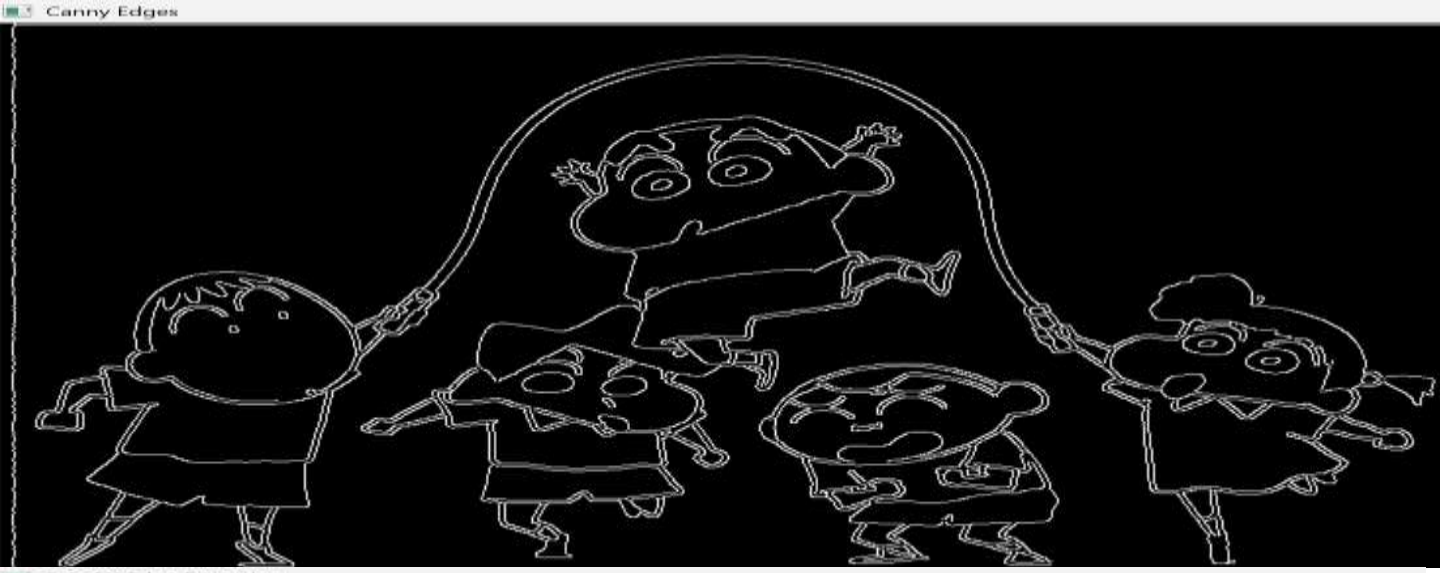

Average Blurred Image


Affine Transform


Perspective Transform


Perspective Transformed Video

Canny Edges

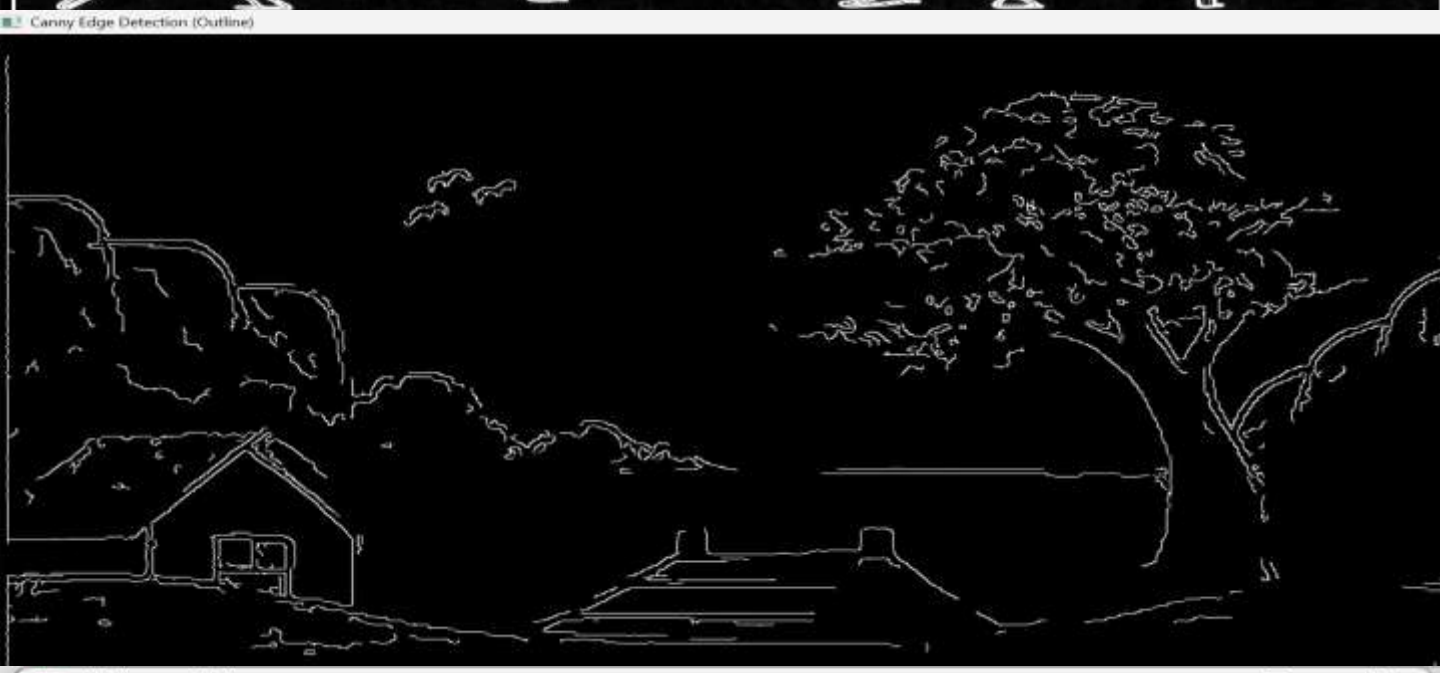


Sobel X Edge Detection

Sobel Y Edge Detection

DLT Transformed Video


Sobel X + Y Edge Detection

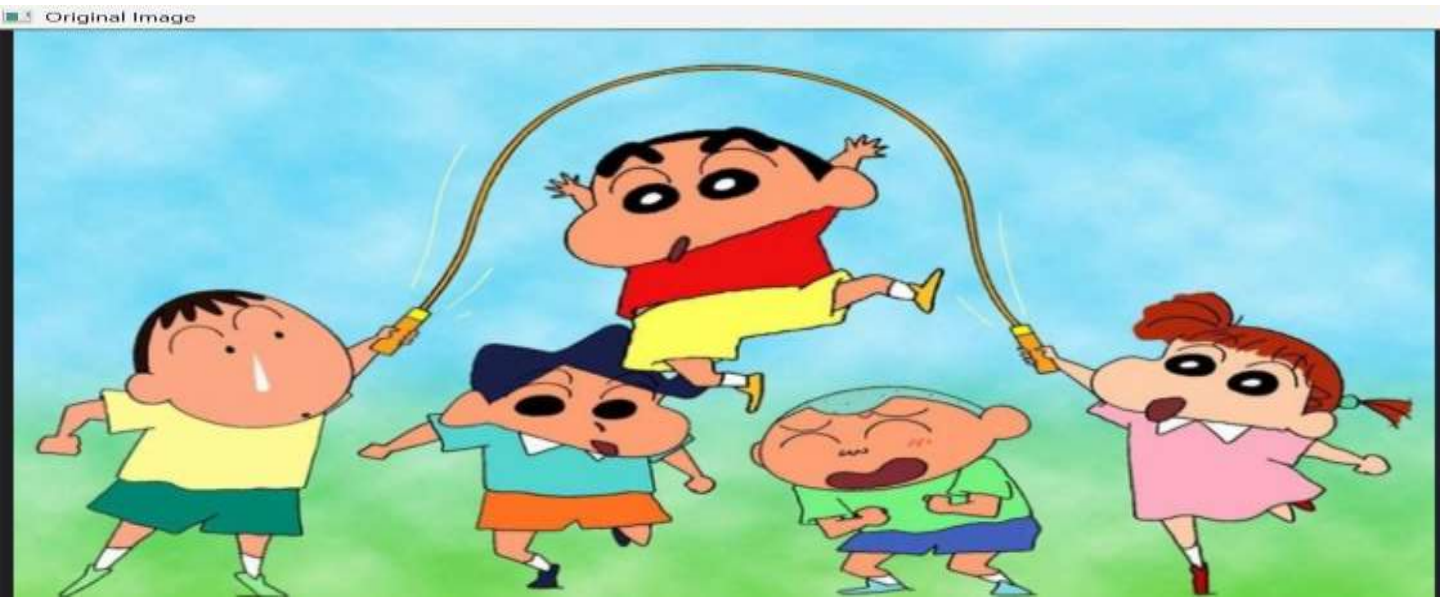

Canny Edge Detection (Outline)

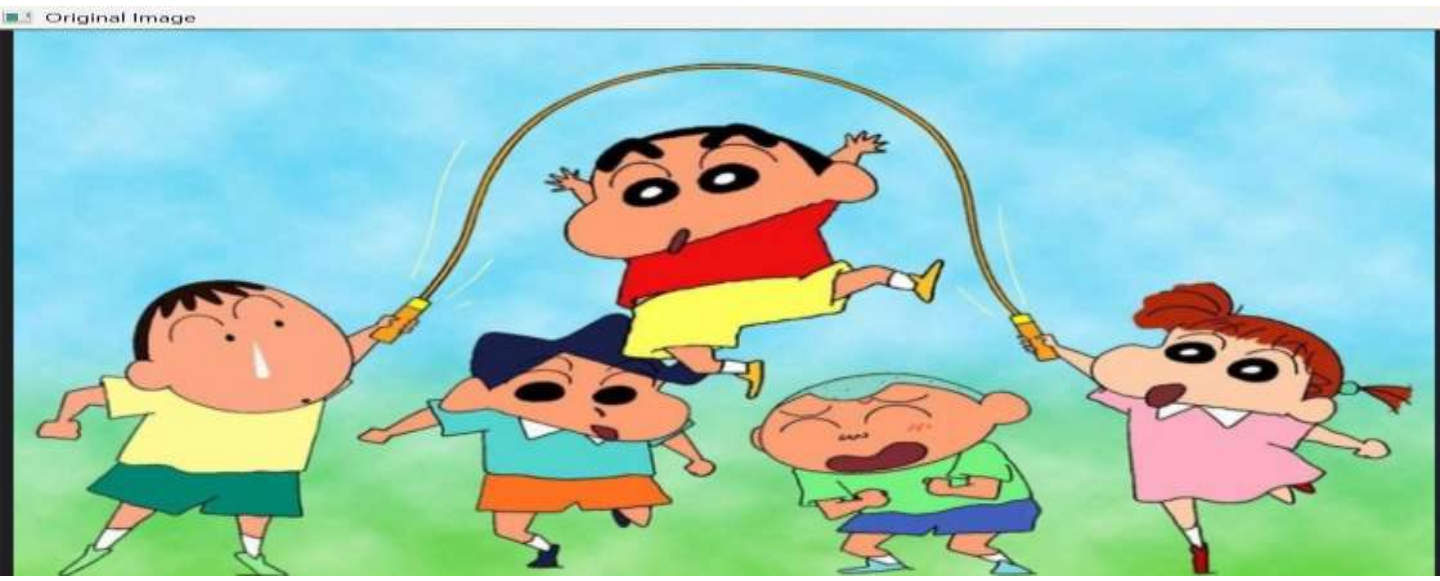

Webcam Video
edu
GAMES


Homography Transformed Video

Original Image

Original Image

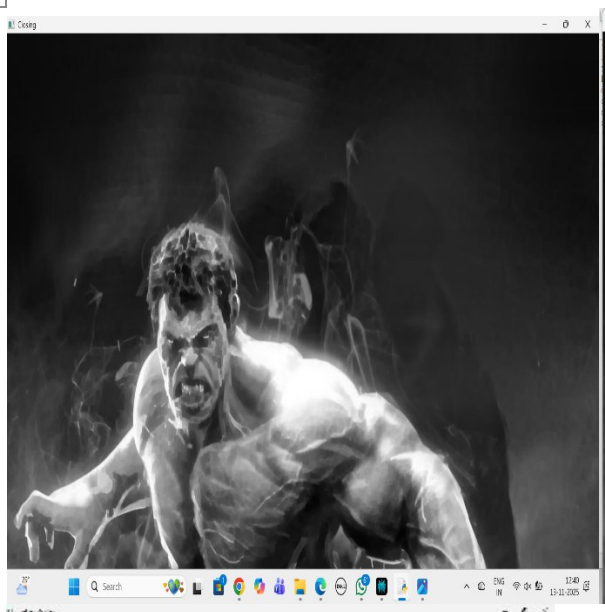

Original Image

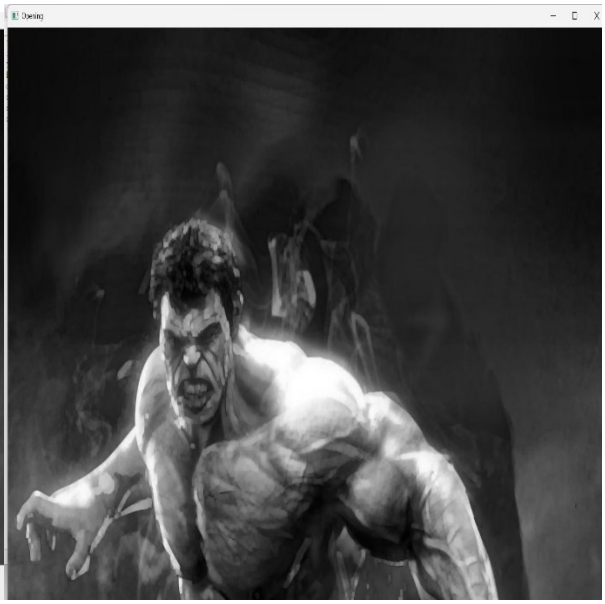

Original Image

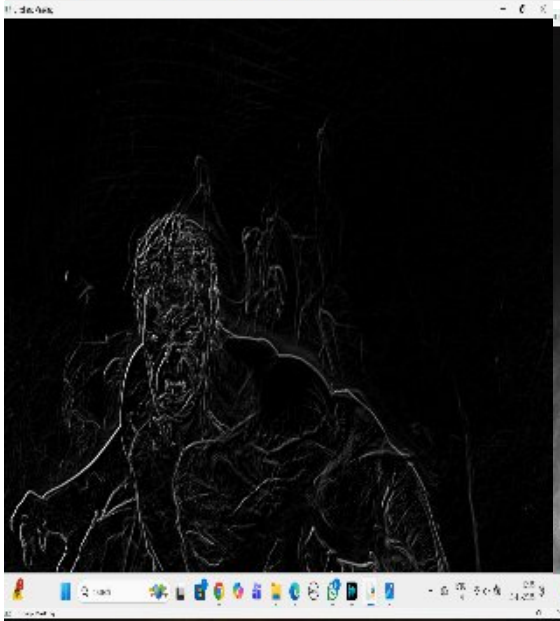

Original Image

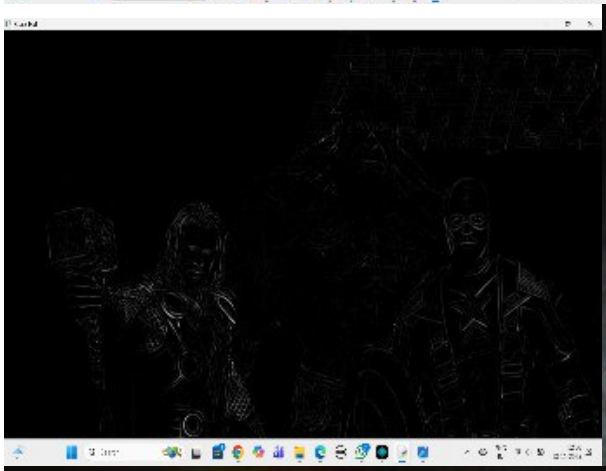

Original Image

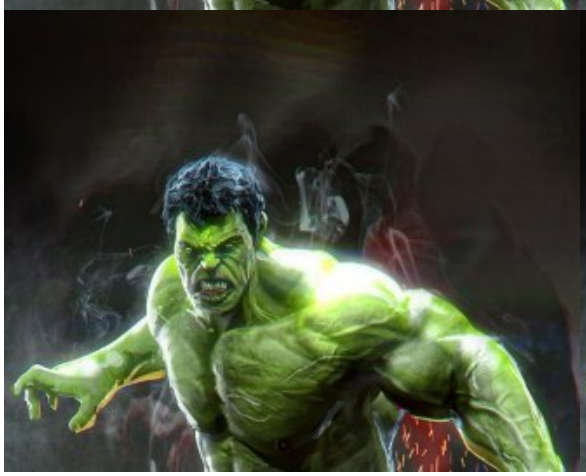

AVENGERS

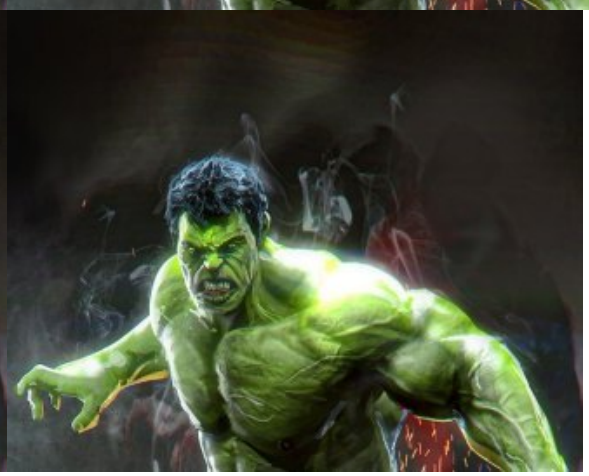




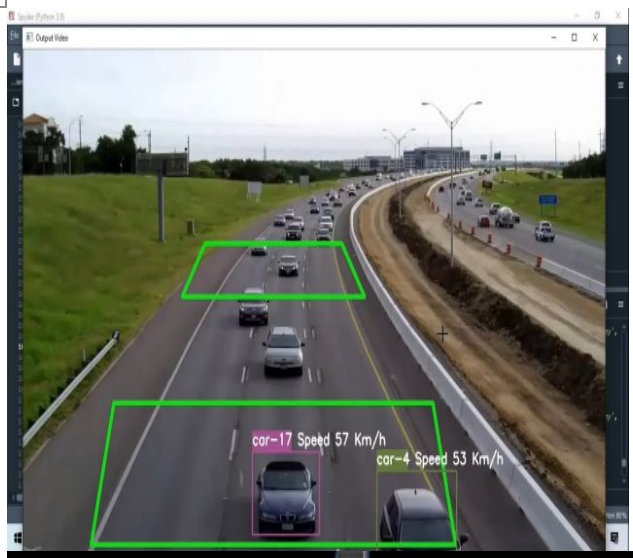




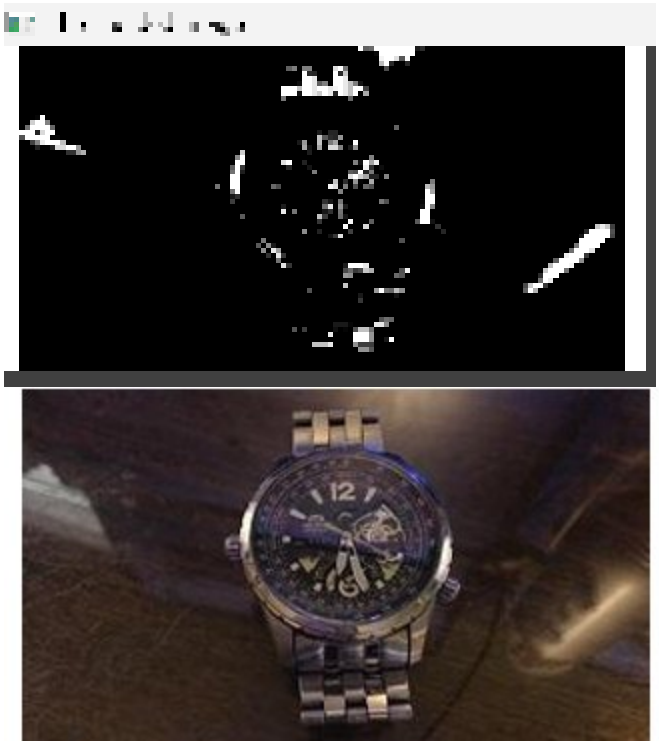


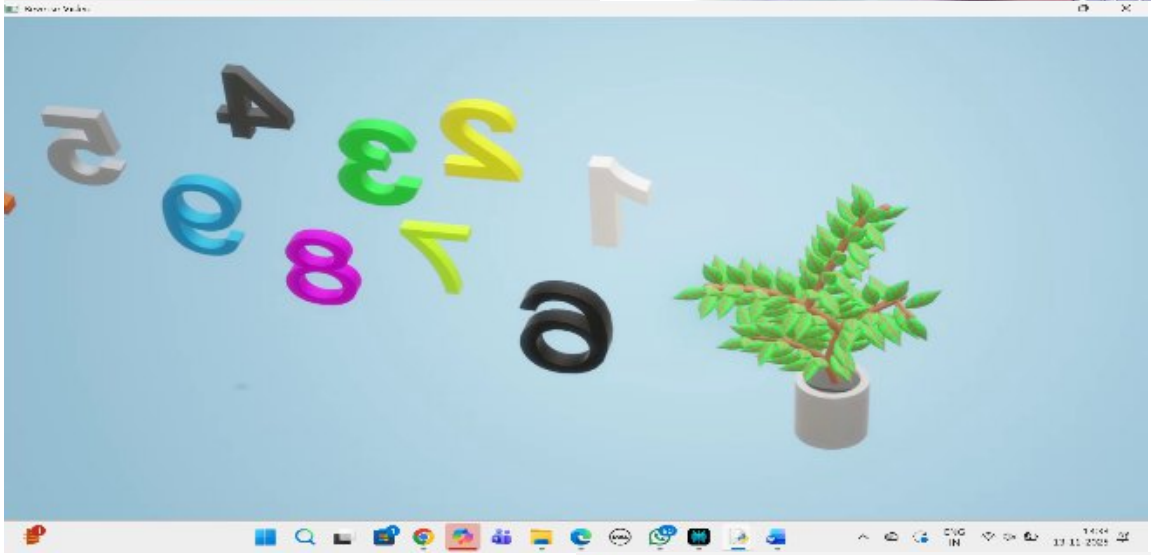



AVENGER


Output Video
car-17 Speed 57 Km/h
car-4 Speed 53 Km/h


AVENGER






Rohit Sharma
Virat kohli


Detected Objects
Rohit Sharma
Virat kohli