**Aim:- Write a program to use Lucas-Kanade method for optical flow detection**

**Theory:-** The Lucas-Kanade method is a differential-based optical flow estimation technique that assumes motion between two consecutive frames is small and relatively constant within a neighborhood of pixels. It was introduced by Bruce D. Lucas and Takeo Kanade in 1981.

#### Principle

The method is based on the following assumptions:

1. Brightness Constancy Assumption: The intensity of a pixel remains constant between two consecutive frames.  
     
    If I(x,y,t)I(x, y, t)I(x,y,t) is the intensity of a pixel at location (x,y)(x, y)(x,y) at time ttt,  
    I(x,y,t)=I(x+δx,y+δy,t+δt)I(x, y, t) = I(x + \delta x, y + \delta y, t + \delta t)I(x,y,t)=I(x+δx,y+δy,t+δt)
2. Small Motion Assumption: The pixel displacements δx\delta xδx and δy\delta yδy between consecutive frames are small.
3. Spatial Coherence Assumption: The motion of neighboring pixels is similar within a small window, usually a square or rectangular region.

**CODE:-**

import cv2

import numpy as np

# Load the video

video\_path = r"C:\Users\aksha\Downloads\vid6.mp4"

video = cv2.VideoCapture(video\_path)

ret, prev\_frame = video.read()

if not ret:

print("Error reading video file")

exit()

# Convert first frame to grayscale

prev\_gray = cv2.cvtColor(prev\_frame, cv2.COLOR\_BGR2GRAY)

# Lucas-Kanade Optical Flow parameters

lk\_params = dict(winSize=(15, 15),

maxLevel=2,

criteria=(cv2.TERM\_CRITERIA\_EPS | cv2.TERM\_CRITERIA\_COUNT, 10, 0.03))

# Detect good features to track

prev\_corners = cv2.goodFeaturesToTrack(prev\_gray, maxCorners=100, qualityLevel=0.3, minDistance=7, blockSize=7)

# Create a mask for drawing motion vectors

mask = np.zeros\_like(prev\_frame)

while True:

ret, curr\_frame = video.read()

if not ret:

break

# Convert current frame to grayscale

curr\_gray = cv2.cvtColor(curr\_frame, cv2.COLOR\_BGR2GRAY)

# Compute Optical Flow

curr\_corners, status, err = cv2.calcOpticalFlowPyrLK(prev\_gray, curr\_gray, prev\_corners, None, \*\*lk\_params)

if curr\_corners is not None and status is not None:

# Filter valid points

good\_old = prev\_corners[status == 1]

good\_new = curr\_corners[status == 1]

for (old, new) in zip(good\_old, good\_new):

x\_old, y\_old = old.ravel()

x\_new, y\_new = new.ravel()

mask = cv2.line(mask, (int(x\_new), int(y\_new)), (int(x\_old), int(y\_old)), (0, 255, 0), 2)

curr\_frame = cv2.circle(curr\_frame, (int(x\_new), int(y\_new)), 5, (0, 0, 255), -1)

# Combine mask with the frame

output = cv2.add(curr\_frame, mask)

# Show the frame in a new window

cv2.imshow("Optical Flow", output)

# Update previous frame and corners

prev\_corners = good\_new.reshape(-1, 1, 2)

prev\_gray = curr\_gray.copy()

# Break if 'q' is pressed

if cv2.waitKey(1) & 0xFF == ord('q'):

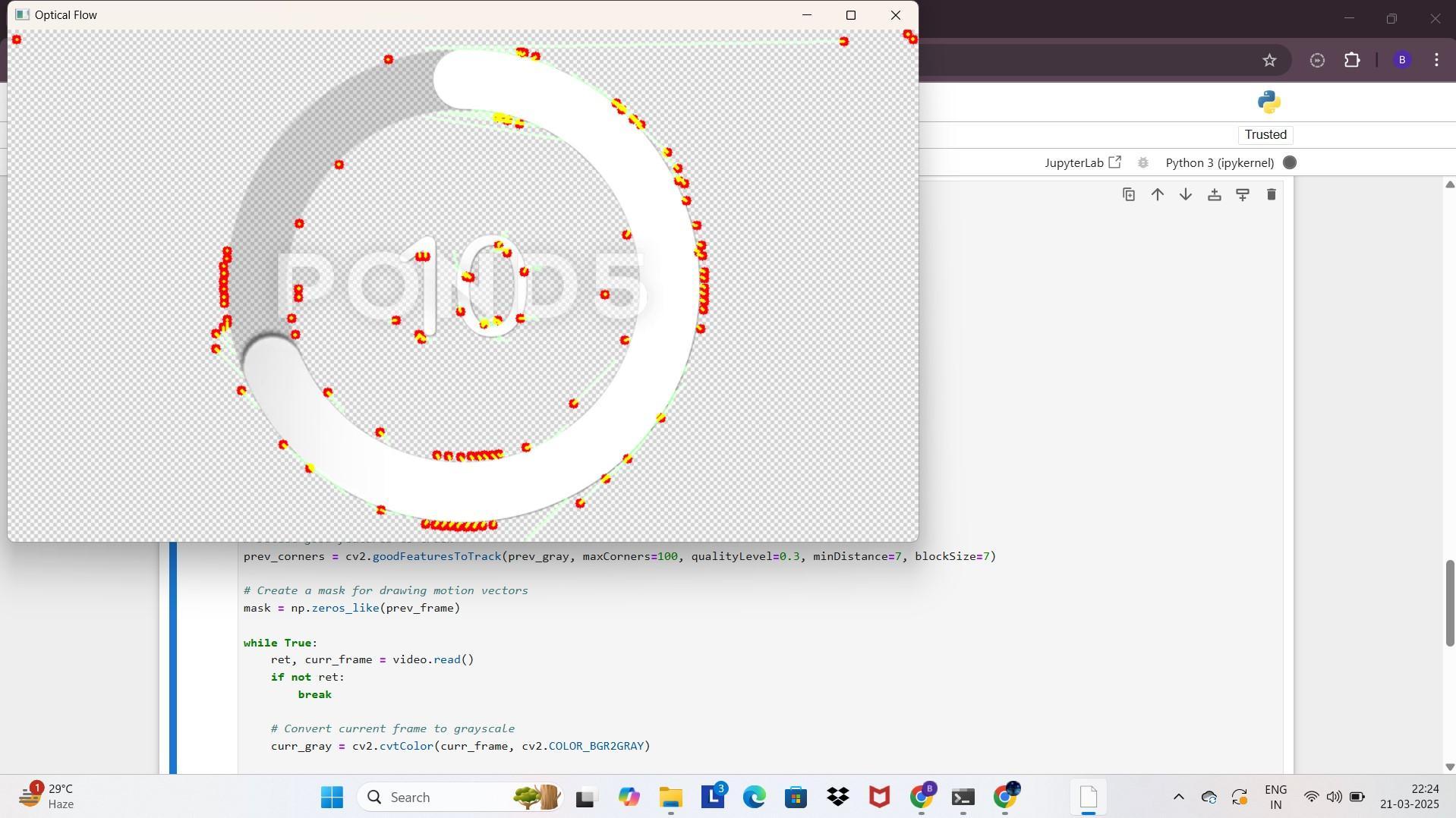
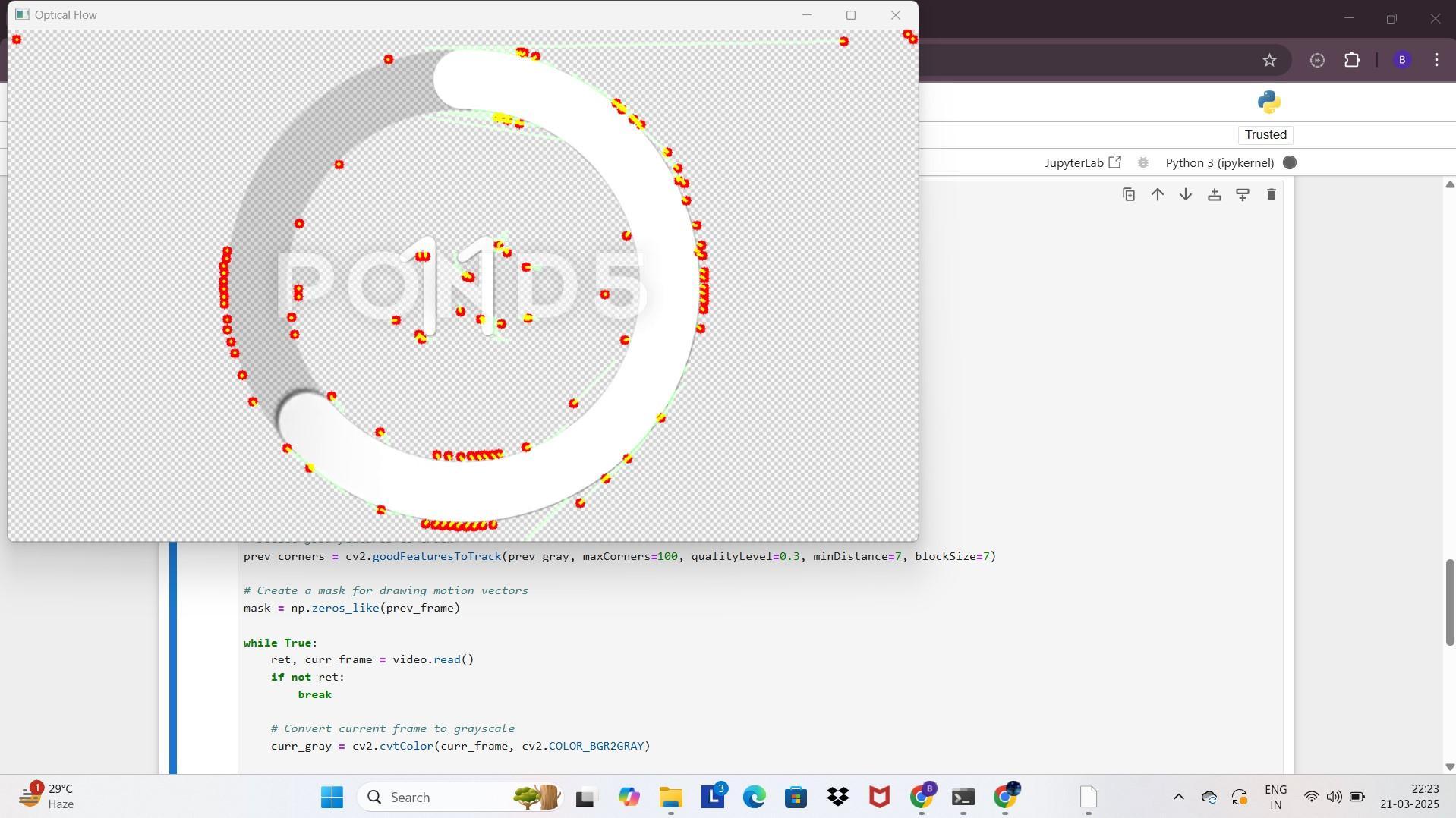
break

# Release resources

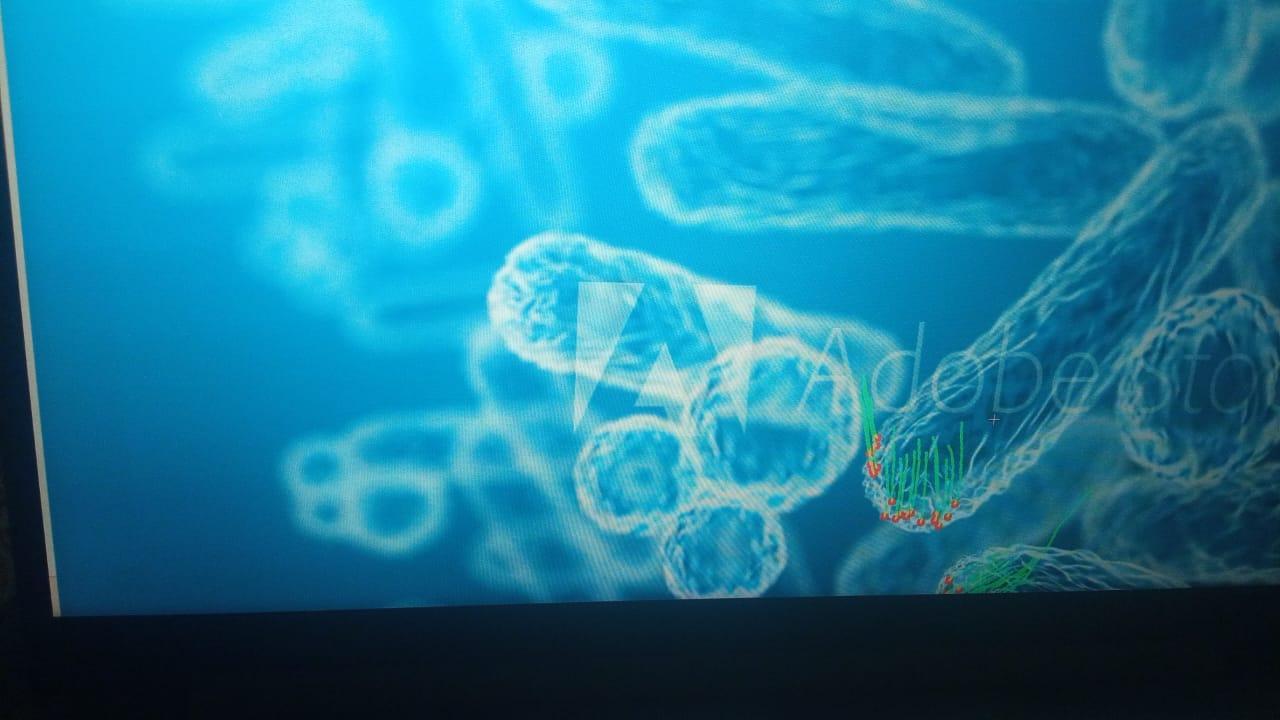
video.release()

cv2.destroyAllWindows()

**NOTE:- Code is Same for both the videos only change is of video and its location.**

**OUTPUT:- **

**OUTPUT 2:-**

****

**Conclusion:-**

The **Lucas-Kanade method** is a powerful and computationally efficient technique for **estimating optical flow** in scenarios where motion is **small and continuous**. It works well for applications such as **video stabilization, motion detection, tracking, and gesture recognition**.

However, it has limitations:

* **Fails for large displacements** between frames (can be mitigated using image pyramids - the **Pyramidal Lucas-Kanade method**).
* Assumes **constant motion within a neighborhood**, which may not hold true for highly textured or complex moving regions.
* Sensitive to **illumination changes** and **noise**.