|  |  |
| --- | --- |
| **Exercise No**:12  **Date**:03.04.25 | **Lucas Kanade and KLT Tracking** |

## AIM:

To apply Lucas Kanade algorithm on two frames and for video using KLT Tracking for tracking the

features.

**CODE:**

# Lucas Kanade and KLT Tracking

**import** cv2

**import** numpy **as** np

**import** gdown

**def** read\_video(url):

file\_id = url.split("/d/")[1].split("/")[0]

download\_url = f'https://drive.google.com/uc?id={file\_id}' gdown.download(download\_url,"video.mp4", quiet=False) **return** cv2.VideoCapture("video.mp4")

**def** play\_video(cap):

**while** cap.isOpened():

ret, frame = cap.read()

**if not** ret:

### break

cv2.imshow("Video Playback", frame)

*# Press 'q' to quit early*

**if** cv2.waitKey(25) & 0xFF == ord('q'):

### break

cv2.destroyAllWindows() cap =

read\_video("https://drive.google.com/file/d/16czHsa364vmD5lnAHgJRXSTAUFDWLrWH/view?us p=sharing")

Downloading...

From: https://drive.google.com/uc?id=16czHsa364vmD5lnAHgJRXSTAUFDWLrWH To: C:\Users\sanka\PYTHON\COMPUTER VISION\video.mp4

100%|█████████████████████████████████████████████████████████████████████████████|

8.13M/8.13M [00:07<00:00, 1.13MB/s]

play\_video(cap)

# Lucas Kanade for Two Frames

**def** lucas\_kanade\_two\_frames(cap): cap.set(cv2.CAP\_PROP\_POS\_FRAMES,0) ret, old\_frame = cap.read()

**if not** ret:

print("Failed to read the first frame")

### return

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

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

lk\_params = dict(winSize=(15, 15), maxLevel=2,

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

0.03))

mask = np.zeros\_like(old\_frame) ret, new\_frame = cap.read()

**if not** ret:

print("Failed to read the second frame")

### return

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

next\_points, status, \_ = cv2.calcOpticalFlowPyrLK(old\_gray, new\_gray, features, None, \*\*lk\_params)

good\_new = next\_points[status == 1] good\_old = features[status == 1]

**for** i, (new, old) **in** enumerate(zip(good\_new, good\_old)): x\_new, y\_new = new.ravel()

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

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

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

1)

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

cv2.imshow('Lucas-Kanade Optical Flow (First Two Frames)', output)

cv2.waitKey(0) cv2.destroyAllWindows()

lucas\_kanade\_two\_frames(cap)

# OUTPUT:

****

**Lucas Kanade and KLT for Video**

ret, old\_frame = cap.read()

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

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

lk\_params = dict(winSize=(15, 15), maxLevel=2,

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

0.03))

mask = np.zeros\_like(old\_frame) cap.set(cv2.CAP\_PROP\_POS\_FRAMES,0)

**while** True:

ret, frame = cap.read()

**if not** ret:

print('Video Finishes')

### break

frame\_gray = cv2.cvtColor(frame, cv2.COLOR\_BGR2GRAY)

next\_points, status, \_ = cv2.calcOpticalFlowPyrLK(old\_gray, frame\_gray, features, None, \*\*lk\_params)

good\_new = next\_points[status == 1] good\_old = features[status == 1]

**for** i, (new, old) **in** enumerate(zip(good\_new, good\_old)): x\_new, y\_new = new.ravel()

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

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

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

output = cv2.add(frame, mask) cv2.imshow('KLT Tracking', output)

old\_gray = frame\_gray.copy()

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

**if** cv2.waitKey(30) & 0xFF == ord('q'):

### break

cv2.destroyAllWindows()

## RESULT:

Thus, Lucas Kanade algorithm on two frames and for video using KLT Tracking for tracking the features was written and executed successfully.