**AIM:** Implement color-based object tracking using HSV space and CamShift algorithm.

**DESCRIPTION:**

Color-based object tracking is a computer vision technique where an object is tracked based on its color properties. Using **HSV (Hue, Saturation, Value) color space** is advantageous because it separates color information (hue) from intensity (value), making tracking more robust to lighting changes. The **CamShift (Continuously Adaptive Mean Shift) algorithm** is an extension of the Mean Shift algorithm, which can track objects even when they change in size, orientation, or position.

**Steps Involved**

**1. Capture Video**

* Acquire a video feed from a camera or video file.
* Convert each frame to HSV color space since it is more effective than RGB for color detection.

**2. Define the Object's Color Range**

* Select the object to track (can be done manually or through a region of interest, ROI).
* Convert the ROI to HSV and calculate a **color histogram** of the target object.
* Specify **lower and upper HSV boundaries** to define the color range for tracking.

**3. Create a Mask**

* Use the HSV range to filter the frame and generate a binary mask:
  + Pixels within the color range are white (foreground).
  + Pixels outside the range are black (background).

**4. Initialize CamShift**

* Initialize the **tracking window** with the object's position (bounding box around the ROI).
* CamShift uses the **Mean Shift algorithm** to locate the mode of the probability distribution in the mask, which corresponds to the object’s location.
* Unlike Mean Shift, CamShift updates the size and orientation of the tracking window dynamically.

**5. Track the Object**

* In each video frame:
  + Apply the mask to the HSV frame.
  + Compute the **back projection** of the histogram on the current frame.
  + Apply **CamShift** to find the new object position, size, and orientation.
  + Draw the tracking window (usually a rotated rectangle) on the original frame.

**6. Display Output**

* Visualize the tracking in real-time.
* Optionally, log or save the coordinates of the object for further analysis.

**Advantages**

* Robust to scale and rotation changes due to CamShift.
* HSV color space provides stability against lighting variations.
* Simple and efficient for real-time tracking applications.

**Applications**

* Human-computer interaction (gesture tracking)
* Sports analysis (tracking balls or players)
* Robotics (object following)
* Surveillance (tracking people or vehicles)

**PROGRAM:**

# Multi-Object CamShift Tracking with Trajectory Logging to CSV

# Auto-installs required packages if missing

import sys

import subprocess

import csv

import time

# Function to install packages

def install(package):

subprocess.check\_call([sys.executable, "-m", "pip", "install", package])

# Required packages

required\_packages = ["opencv-python", "numpy"]

for package in required\_packages:

try:

\_\_import\_\_(package)

except ImportError:

install(package)

import cv2

import numpy as np

import random

from collections import deque

# Capture video

cap = cv2.VideoCapture(0)

# Dictionary to hold tracking info for each object

tracked\_objects = {}

object\_id\_counter = 0

# Termination criteria for CamShift

term\_crit = (cv2.TERM\_CRITERIA\_EPS | cv2.TERM\_CRITERIA\_COUNT, 10, 1)

# Maximum trajectory length

MAX\_TRAJECTORY\_LENGTH = 50

# CSV file setup

csv\_filename = "object\_trajectories.csv"

csv\_file = open(csv\_filename, mode='w', newline='')

csv\_writer = csv.writer(csv\_file)

csv\_writer.writerow(["Timestamp", "ObjectID", "X", "Y"])

print(f"Trajectory logging started: {csv\_filename}")

while True:

ret, frame = cap.read()

if not ret:

break

hsv = cv2.cvtColor(frame, cv2.COLOR\_BGR2HSV)

blurred = cv2.GaussianBlur(hsv, (7, 7), 0)

gray = blurred[:, :, 2] # Value channel

\_, thresh = cv2.threshold(gray, 50, 255, cv2.THRESH\_BINARY)

# Find contours of potential objects

contours, \_ = cv2.findContours(thresh, cv2.RETR\_EXTERNAL, cv2.CHAIN\_APPROX\_SIMPLE)

detected\_windows = []

for c in contours:

if cv2.contourArea(c) < 500: # Filter noise

continue

x, y, w, h = cv2.boundingRect(c)

detected\_windows.append((x, y, w, h))

# Check if this object is already being tracked

matched\_id = None

for obj\_id, obj\_info in tracked\_objects.items():

tx, ty, tw, th = obj\_info['window']

iou = (max(0, min(x + w, tx + tw) - max(x, tx)) \*

max(0, min(y + h, ty + th) - max(y, ty))) / float(w\*h + tw\*th - max(0, min(x + w, tx + tw) - max(x, tx)) \* max(0, min(y + h, ty + th) - max(y, ty)))

if iou > 0.2:

matched\_id = obj\_id

break

if matched\_id is None:

# New object detected

object\_id\_counter += 1

roi = hsv[y:y+h, x:x+w]

roi\_hist = cv2.calcHist([roi], [0], None, [180], [0, 180])

cv2.normalize(roi\_hist, roi\_hist, 0, 255, cv2.NORM\_MINMAX)

display\_color = (random.randint(50,255), random.randint(50,255), random.randint(50,255))

tracked\_objects[object\_id\_counter] = {'window': (x, y, w, h), 'hist': roi\_hist, 'color': display\_color, 'trajectory': deque(maxlen=MAX\_TRAJECTORY\_LENGTH)}

# Apply CamShift to each tracked object

for obj\_id, obj\_info in tracked\_objects.items():

back\_proj = cv2.calcBackProject([hsv], [0], obj\_info['hist'], [0, 180], 1)

ret\_cam, obj\_info['window'] = cv2.CamShift(back\_proj, obj\_info['window'], term\_crit)

pts = cv2.boxPoints(ret\_cam)

pts = np.int0(pts)

frame = cv2.polylines(frame, [pts], True, obj\_info['color'], 2)

# Add current center to trajectory

cx = int(ret\_cam[0][0])

cy = int(ret\_cam[0][1])

obj\_info['trajectory'].append((cx, cy))

# Draw trajectory

for i in range(1, len(obj\_info['trajectory'])):

cv2.line(frame, obj\_info['trajectory'][i-1], obj\_info['trajectory'][i], obj\_info['color'], 2)

# Log trajectory to CSV

timestamp = time.time()

csv\_writer.writerow([timestamp, obj\_id, cx, cy])

cv2.imshow("Multi-Object CamShift Tracking with CSV Logging", frame)

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

break

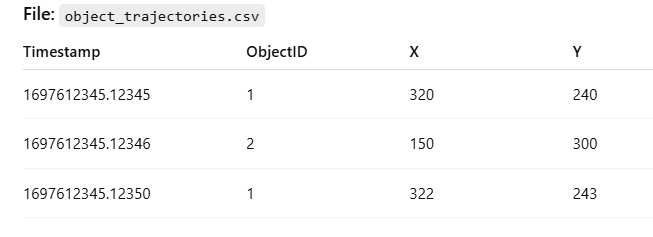
cap.release()

csv\_file.close()

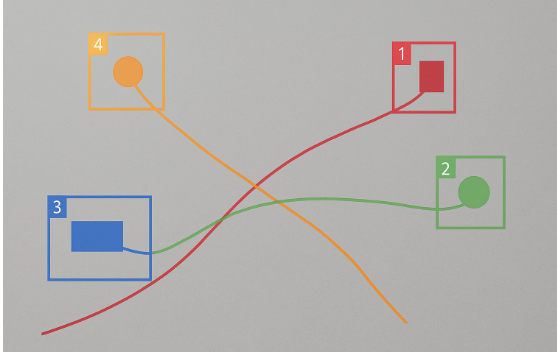
cv2.destroyAllWindows()

print(f"Trajectory logging finished. Data saved to {csv\_filename}")

**OUTPUT:**

****





**RESULT:**

Thus, we implemented color-based object tracking using HSV space and CamShift algorithm.