Ai

**Background subtraction and scene detection.**

**Ex No : 11**

**Date : 27.3.2025**

**AIM :**

To perform background subtraction (BGS), BGS using averaging method and scene detection using BGS.

**Background subtraction:**

1. Import necessary libraries (cv2, numpy).
2. Initialize video capture from the given video file.
3. Create background subtractors (MOG and MOG2).
4. Start a loop to process each video frame until the video ends.
5. Read a frame from the video and check if it exists.
6. Apply background subtraction using both MOG and MOG2.
7. Resize the original and processed frames for consistent display.
8. Display the original video and the two subtracted frames using cv2.imshow().
9. Wait for a key press and check if the user presses ESC to exit.
10. Release video resources and close all OpenCV windows after exiting.

**CODE** :

import cv2

import numpy as np

import matplotlib.pyplot as plt

fgbg1 = cv2.bgsegm.createBackgroundSubtractorMOG();

fgbg2 = cv2.createBackgroundSubtractorMOG2();

cap = cv2.VideoCapture(r"C:\Users\haris\Downloads\video.mp4");

while(1):

# read frames

ret, img = cap.read();

# apply mask for background subtraction

fgmask1 = cv2.resize(fgbg1.apply(img),(852,480));

fgmask2 = cv2.resize(fgbg2.apply(img),(852,480));

cv2.imshow('Original', cv2.resize(img,(852,480)));

cv2.imshow('MOG', fgmask1);

cv2.imshow('MOG2', fgmask2);

k = cv2.waitKey(30) & 0xff;

if k == 27:

break;

cap.release();

cv2.destroyAllWindows();

**Output:**



**Background subtraction using averaging:**

1. Initialize video capture using cv2.VideoCapture().
2. Read the first frame and check if the video is accessible.
3. Convert the first frame to grayscale and initialize the background model.
4. Loop through video frames until the video ends:

Convert each frame to grayscale.

Update the background model using cv2.accumulateWeighted().

Compute foreground mask using cv2.absdiff().

Apply thresholding to extract moving objects.

Resize and display the background model and foreground mask.

1. Exit on 'ESC' key press and release resources.

**CODE** :

cap = cv2.VideoCapture(r"C:\Users\haris\Downloads\video.mp4")

ret, frame = cap.read()

if not ret:

print("Error: Couldn't read video")

cap.release()

cv2.destroyAllWindows()

exit()

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

avg\_bg = np.float32(gray)

while True:

ret, frame = cap.read()

if not ret:

break

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

cv2.accumulateWeighted(gray, avg\_bg, 0.01)

bg\_model = cv2.convertScaleAbs(avg\_bg)

fg\_mask = cv2.absdiff(gray, bg\_model)

\_, fg\_mask = cv2.threshold(fg\_mask, 40, 255, cv2.THRESH\_BINARY)

small\_bg\_model = cv2.resize(bg\_model, (600, 480)) # Adjust width & height as needed

small\_fg\_mask = cv2.resize(fg\_mask, (600, 480))

cv2.imshow("Background Model", small\_bg\_model)

cv2.imshow("Foreground Mask", small\_fg\_mask)

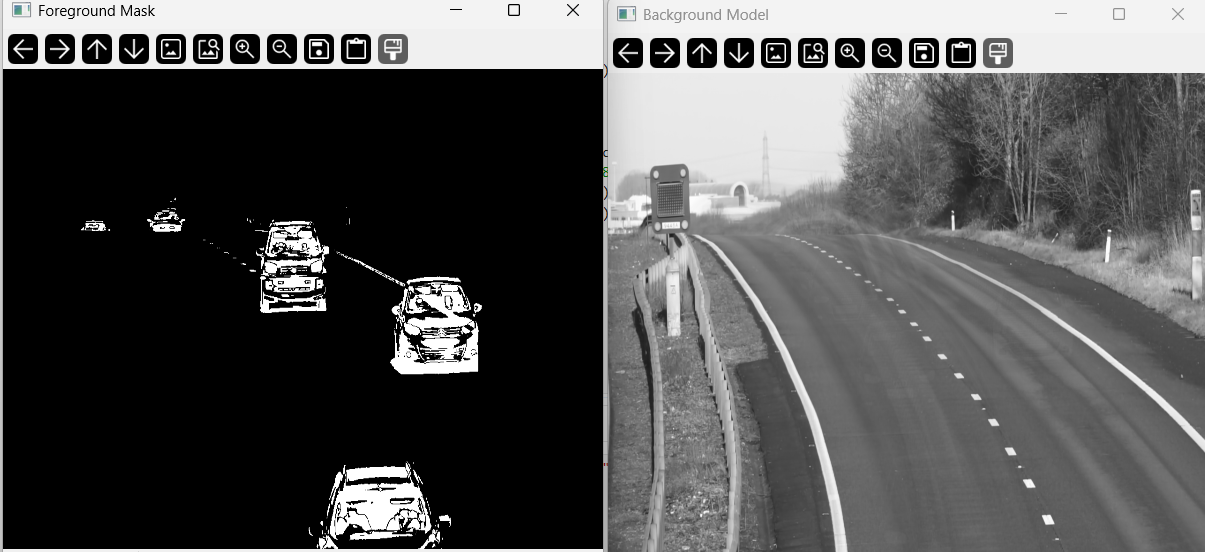
if cv2.waitKey(30) & 0xFF == 27:

break

cap.release()

cv2.destroyAllWindows()

**Output:**



**Scene detection using Background subtraction:**

1. Initialize video capture using cv2.VideoCapture().
2. Create Background Subtractor using cv2.createBackgroundSubtractorMOG2().
3. Read the first frame and check if the video is accessible.
4. Store the first frame as the initial scene.
5. Convert the first frame to grayscale and compute its histogram for comparison.
6. Loop through the video frames:

Convert the frame to grayscale and compute its histogram.

Compare histograms using cv2.compareHist() to detect scene changes.

Apply background subtraction to detect motion and compute change ratio.

If histogram similarity is low & motion is high, mark scene change detection.

If histogram similarity becomes high, mark scene stabilization and save the frame.

Update the previous histogram for the next iteration.

1. Release video resources after processing.
2. Display all detected scene frames using matplotlib.pyplot.
3. Print total detected scenes and complete processing.

**CODE** :

video\_path = r"C:\Users\haris\Downloads\mixed.mp4"

cap = cv2.VideoCapture(video\_path)

fgbg = cv2.createBackgroundSubtractorMOG2()

ret, frame = cap.read()

if not ret:

print("Error: Couldn't read video")

cap.release()

exit()

scenes = [frame.copy()]

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

prev\_hist = cv2.calcHist([prev\_gray], [0], None, [256], [0, 256])

cv2.normalize(prev\_hist, prev\_hist, 0, 1, cv2.NORM\_MINMAX)

scene\_change\_detected = False

while True:

ret, frame = cap.read()

if not ret:

break

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

curr\_hist = cv2.calcHist([gray], [0], None, [256], [0, 256])

cv2.normalize(curr\_hist, curr\_hist, 0, 1, cv2.NORM\_MINMAX)

similarity = cv2.compareHist(prev\_hist, curr\_hist, cv2.HISTCMP\_CORREL)

fg\_mask = fgbg.apply(frame)

change\_ratio = np.count\_nonzero(fg\_mask) / fg\_mask.size

if similarity < 0.5 and change\_ratio > 0.3 and not scene\_change\_detected:

print(f"Scene {len(scenes) + 1} Changed! Waiting for stabilization...")

scene\_change\_detected = True

elif scene\_change\_detected and similarity > 0.9:

print(f"Scene {len(scenes) + 1} Stabilized! Saving frame...")

scenes.append(frame.copy())

scene\_change\_detected = False # Reset flag

prev\_hist = curr\_hist.copy()

cap.release()

print(f"Total Scenes Detected: {len(scenes)}")

plt.figure(figsize=(15, 5))

for i, scene in enumerate(scenes):

plt.subplot(1, len(scenes), i + 1)

plt.imshow(cv2.cvtColor(scene, cv2.COLOR\_BGR2RGB))

plt.title(f"Scene {i + 1}")

plt.axis("off")

plt.show()

print("Processing Complete. All detected scenes displayed.")

**Output:**



**RESULT :**

Hence , Background subtraction, BDS with averaging and scene detecion has been implemented.