**Name :** Akshat Sharma

**Aim :** Write a program to implement Background subtraction method.

**Library Used :** cv2,numpy, matplotlib

**Theory :**

**Background Subtraction (BGS)** is a technique used to extract moving objects (foreground) from a static background in video sequences.

It works by building a **background model** and comparing each incoming frame with this model to detect changes.

Pixels that differ significantly from the background are marked as **foreground**, while others remain as **background**.

The most common methods include :

1. **Gaussian Mixture Model (GMM)** : Models each pixel as a mixture of Gaussians and updates the model over time.
2. **MOG2** : An improved version of GMM that handles illumination changes and shadows.
3. **Frame Detcetion** : After generating the foreground mask, **contour detection** is used to find the boundaries of moving objects.  
   cv2.boundingRect() is applied to draw a rectangle around the largest moving object(s).

### **Applications of Background Subtraction :**

* **Object Detection and Tracking:** Identifies and tracks moving objects in surveillance videos.
* **Human Activity Recognition:** Monitors human activities for security and monitoring purposes.
* **Traffic Monitoring:** Detects vehicles and counts traffic flow.
* **Video Compression:** Removes redundant information by focusing on the moving parts.

**Code :**

**import cv2**

**import numpy as np**

**import matplotlib.pyplot as plt**

**# Read the image and resize**

**image = cv2.imread("blue\_bottle.jpg")**

**image = cv2.resize(image, (500, 500))**

**image\_rgb = cv2.cvtColor(image, cv2.COLOR\_BGR2RGB)**

**# Create an initial mask**

**mask = np.zeros(image.shape[:2], dtype="uint8")**

**# Define model for background and foreground**

**bgdModel = np.zeros((1, 65), np.float64)**

**fgdModel = np.zeros((1, 65), np.float64)**

**# Define a rectangle around the foreground object**

**height, width = image.shape[:2]**

**rect = (15, 15, width - 30, height - 30)**

**# Apply grabCut**

**cv2.grabCut(image, mask, rect, bgdModel, fgdModel, 5, cv2.GC\_INIT\_WITH\_RECT)**

**# Create a binary mask where 1 represents foreground and 0 represents background**

**mask\_binary = np.where((mask == 2) | (mask == 0), 0, 1).astype('uint8')**

**# Apply mask to keep only the foreground**

**foreground = image \* mask\_binary[:, :, np.newaxis]**

**# Convert to RGB for correct color display**

**foreground\_rgb = cv2.cvtColor(foreground, cv2.COLOR\_BGR2RGB)**

**# Display the result**

**fig, ax = plt.subplots(1, 2, figsize=(12, 6))**

**ax[0].imshow(image\_rgb)**

**ax[0].set\_title("Original Image")**

**ax[0].axis("off")**

**ax[1].imshow(foreground\_rgb)**

**ax[1].set\_title("Background Removed Image")**

**ax[1].axis("off")**

**plt.show()**

**Output:**

****

**Code : with trying different parameters.**

**#change image**

**import cv2**

**import numpy as np**

**import matplotlib.pyplot as plt**

**# Read the image and resize**

**image = cv2.imread("nature.jpg")**

**image = cv2.resize(image, (200, 200))**

**image\_rgb = cv2.cvtColor(image, cv2.COLOR\_BGR2RGB)**

**# Create an initial mask**

**mask = np.zeros(image.shape[:2], dtype="uint8")**

**# Define model for background and foreground**

**bgdModel = np.zeros((1, 65), np.float64)**

**fgdModel = np.zeros((1, 65), np.float64)**

**# Define a rectangle around the foreground object**

**height, width = image.shape[:2]**

**rect = (20,20, width - 30, height - 30)**

**# Apply grabCut**

**cv2.grabCut(image, mask, rect, bgdModel, fgdModel, 5, cv2.GC\_INIT\_WITH\_RECT)**

**# Create a binary mask where 1 represents foreground and 0 represents background**

**mask\_binary = np.where((mask == 2) | (mask == 0), 0, 1).astype('uint8')**

**# Apply mask to keep only the foreground**

**foreground = image \* mask\_binary[:, :, np.newaxis]**

**# Convert to RGB for correct color display**

**foreground\_rgb = cv2.cvtColor(foreground, cv2.COLOR\_BGR2RGB)**

**# Display the result**

**fig, ax = plt.subplots(1, 2, figsize=(15,15))**

**ax[0].imshow(image\_rgb)**

**ax[0].set\_title("Original Image")**

**ax[0].axis("off")**

**ax[1].imshow(foreground\_rgb)**

**ax[1].set\_title("Background Removed Image")**

**ax[1].axis("off")**

**plt.show()**

**Output :**

****

**Code with mixture of gaussian algorithm :**

**import cv2**

**import numpy as np**

**import matplotlib.pyplot as plt**

**# Read the image and resize**

**image = cv2.imread("blue\_bottle.jpg")**

**# Check if image is loaded correctly**

**if image is None:**

**print("Error: Unable to load image. Check the file path or format.")**

**exit()**

**# Resize the image to desired dimensions**

**image = cv2.resize(image, (500, 500))**

**image\_rgb = cv2.cvtColor(image, cv2.COLOR\_BGR2RGB)**

**# Create a Background Subtractor using MOG2 (Mixture of Gaussians)**

**bg\_subtractor = cv2.createBackgroundSubtractorMOG2(history=100, varThreshold=50, detectShadows=True)**

**# Apply the background subtractor to get the foreground mask**

**mask = bg\_subtractor.apply(image)**

**# Refine the mask using morphological operations to remove noise**

**kernel = cv2.getStructuringElement(cv2.MORPH\_ELLIPSE, (5, 5))**

**mask = cv2.morphologyEx(mask, cv2.MORPH\_CLOSE, kernel)**

**mask = cv2.morphologyEx(mask, cv2.MORPH\_OPEN, kernel)**

**# Create a 3-channel mask to apply to the original image**

**mask\_3channel = cv2.merge([mask, mask, mask])**

**# Apply the mask to extract the foreground**

**foreground = cv2.bitwise\_and(image, mask\_3channel)**

**# Convert the result to RGB for displaying with matplotlib**

**foreground\_rgb = cv2.cvtColor(foreground, cv2.COLOR\_BGR2RGB)**

**# Display the original and the extracted foreground side by side**

**fig, ax = plt.subplots(1, 2, figsize=(12, 6))**

**ax[0].imshow(image\_rgb)**

**ax[0].set\_title("Original Image")**

**ax[0].axis("off")**

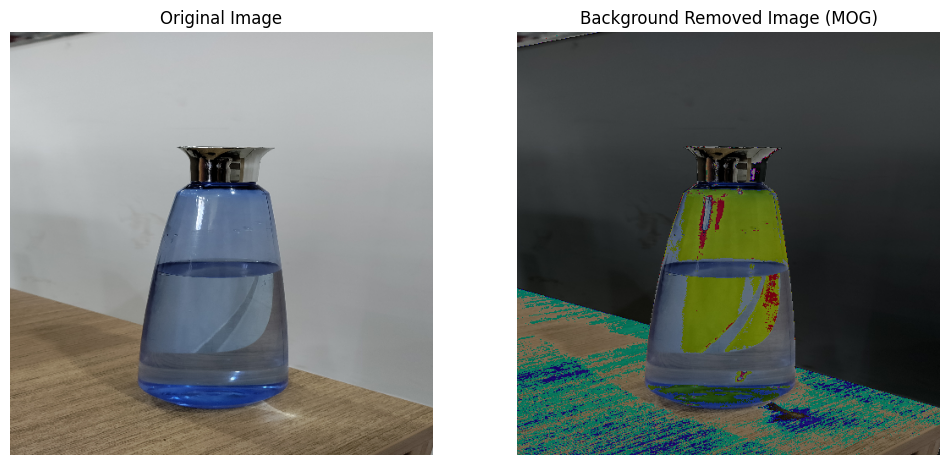
**ax[1].imshow(foreground\_rgb)**

**ax[1].set\_title("Background Removed Image (MOG)")**

**ax[1].axis("off")**

**plt.show()**

**Output :**

****

**Code with frame detection algorithm :**

**import cv2**

**import numpy as np**

**import matplotlib.pyplot as plt**

**# Read the image and resize**

**image = cv2.imread("Taj.jpeg")**

**image = cv2.resize(image, (500, 500))**

**image\_rgb = cv2.cvtColor(image, cv2.COLOR\_BGR2RGB)**

**# Create an initial mask**

**mask = np.zeros(image.shape[:2], dtype="uint8")**

**# Define model for background and foreground**

**bgdModel = np.zeros((1, 65), np.float64)**

**fgdModel = np.zeros((1, 65), np.float64)**

**# Define a rectangle around the foreground object**

**height, width = image.shape[:2]**

**rect = (15, 15, width - 30, height - 30)**

**# Apply grabCut to segment the foreground**

**cv2.grabCut(image, mask, rect, bgdModel, fgdModel, 5, cv2.GC\_INIT\_WITH\_RECT)**

**# Create a binary mask where 1 represents foreground and 0 represents background**

**mask\_binary = np.where((mask == 2) | (mask == 0), 0, 1).astype('uint8')**

**# Apply mask to keep only the foreground**

**foreground = image \* mask\_binary[:, :, np.newaxis]**

**# Convert the binary mask to grayscale for contour detection**

**mask\_gray = (mask\_binary \* 255).astype('uint8')**

**# Find contours in the binary mask**

**contours, \_ = cv2.findContours(mask\_gray, cv2.RETR\_EXTERNAL, cv2.CHAIN\_APPROX\_SIMPLE)**

**# Draw a bounding rectangle around the largest contour (foreground)**

**if contours:**

**largest\_contour = max(contours, key=cv2.contourArea)**

**x, y, w, h = cv2.boundingRect(largest\_contour)**

**cv2.rectangle(foreground, (x, y), (x + w, y + h), (0, 255, 0), 2) # Green rectangle with thickness 2**

**# Convert the result to RGB for correct color display**

**foreground\_rgb = cv2.cvtColor(foreground, cv2.COLOR\_BGR2RGB)**

**# Display the result**

**fig, ax = plt.subplots(1, 2, figsize=(12, 6))**

**ax[0].imshow(image\_rgb)**

**ax[0].set\_title("Original Image")**

**ax[0].axis("off")**

**ax[1].imshow(foreground\_rgb)**

**ax[1].set\_title("Foreground with Frame Detected")**

**ax[1].axis("off")**

**plt.show()**

**Code with video :**

**import cv2**

**import numpy as np**

**import matplotlib.pyplot as plt**

**# Open the video stream (for example, a video file or webcam)**

**cap = cv2.VideoCapture("video.mp4") # Replace with 0 for webcam (e.g., cap = cv2.VideoCapture(0))**

**# Check if video capture is successful**

**if not cap.isOpened():**

**print("Error: Could not open video stream.")**

**exit()**

**# Create a figure for displaying results**

**fig, ax = plt.subplots(1, 2, figsize=(12, 6))**

**# Create the MOG2 background subtractor object**

**fgbg = cv2.createBackgroundSubtractorMOG2()**

**while True:**

**ret, frame = cap.read()**

**if not ret:**

**break # Break if no frame is read (end of video)**

**# Resize the frame for faster processing (optional)**

**frame\_resized = cv2.resize(frame, (500, 500))**

**# Convert the image to RGB for display with matplotlib**

**frame\_rgb = cv2.cvtColor(frame\_resized, cv2.COLOR\_BGR2RGB)**

**# Apply MOG2 background subtraction**

**fgmask = fgbg.apply(frame\_resized)**

**# Apply the mask to the frame to extract the foreground**

**foreground = cv2.bitwise\_and(frame\_resized, frame\_resized, mask=fgmask)**

**# Convert the foreground image to RGB for display**

**foreground\_rgb = cv2.cvtColor(foreground, cv2.COLOR\_BGR2RGB)**

**# Display the original and foreground images in real-time**

**ax[0].imshow(frame\_rgb)**

**ax[0].set\_title("Original Frame")**

**ax[0].axis("off")**

**ax[1].imshow(foreground\_rgb)**

**ax[1].set\_title("Foreground (MOG2)")**

**ax[1].axis("off")**

**plt.draw() # Update the figure**

**plt.pause(0.01) # Pause to allow the plot to update**

**# Optional: Close the window if 'q' is pressed**

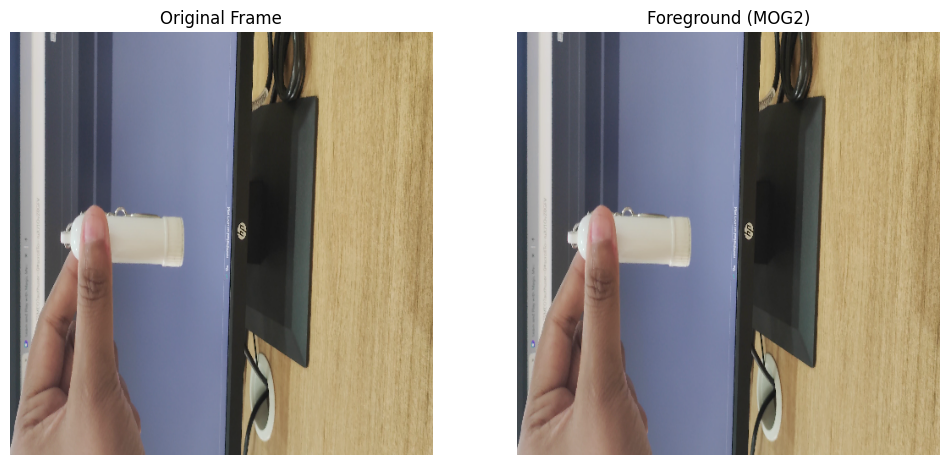
**if cv2.waitKey(1) & 0xFF == ord('q'):**

**break**

**# Release the video capture and close any windows**

**cap.release()**

**cv2.destroyAllWindows()**

****

### **Background Subtraction using OpenCV**

**Here's a simple program to implement background subtraction using OpenCV:**

**import cv2**

**import numpy as np**

**import matplotlib.pyplot as plt**

**# Initialize the background subtractor**

**fgbg = cv2.createBackgroundSubtractorMOG2()**

**# Open the video capture (0 for webcam, or provide a video file path)**

**cap = cv2.VideoCapture(0)**

**while cap.isOpened():**

**ret, frame = cap.read()**

**if not ret:**

**break**

**# Apply background subtraction**

**fgmask = fgbg.apply(frame)**

**# Display the original frame and the foreground mask**

**cv2.imshow("Original Frame", frame)**

**cv2.imshow("Foreground Mask", fgmask)**

**if cv2.waitKey(30) & 0xFF == 27: # Press 'ESC' to exit**

**break**

**cap.release()**

**cv2.destroyAllWindows()**

### **Conclusion:**

1. **Background subtraction is an effective technique for detecting moving objects in a video stream.**
2. **The cv2.createBackgroundSubtractorMOG2() method dynamically learns the background and adapts to changes.**
3. **It is widely used in applications such as surveillance, object tracking, and motion detection.**
4. **By tuning parameters, we can improve the accuracy of background subtraction and reduce noise.**