EXERCISE NO. 5
MULTIPLE OBJECT TRACKING USING OPENCV

AIM:
To implement multiple object tracking using OpenCV.

ALGORITHM:
1. Import the necessary libraries.
2. Download the video.
3. Read the frames and get dimensions.
4. Set bounding boxes based on the frame dimensions.
5. Create a multi object tracking object.
6. Initialise the object.
7. Process the key frames and draw tracked objects.
8. Display the frame with tracking boxes.

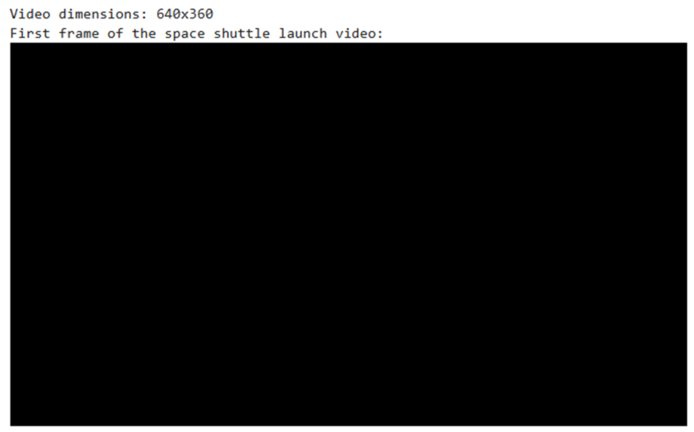
PROGRAM:
```
import cv2
import yt_dlp
import numpy as np
from google.colab.patches import cv2_imshow
import time

def download_video(url, output_path='/downloaded_video.mp4'):
    ydl_opts = {
        'format': 'best[height<=720]',
        'outtmpl': output_path
    }

    with yt_dlp.YoutubeDL(ydl_opts) as ydl:
        ydl.download([url])
    print(f"Video downloaded successfully: {output_path}")
    return output_path

video_url = 'https://www.youtube.com/watch?v=Vfn_u768UoQ'

print("Downloading space shuttle launch video. This may take a moment...")
```

```
video_path = download_video(video_url, '/downloaded_video.mp4')

def track_space_shuttle_launch(video_path, tracker_type="KCF"):
    cap = cv2.VideoCapture(video_path)

    if not cap.isOpened():
        print("Error: Could not open video.")
        return

        ret, frame = cap.read()
    if not ret:
        print("Error: Couldn't read the first video frame.")
        return

    frame_height, frame_width = frame.shape[:2]
    print(f"Video dimensions: {frame_width}x{frame_height}")

    print("First frame of the space shuttle launch video:")
    cv2_imshow(frame)

    bboxes = [
        (int(frame_width * 0.4), int(frame_height * 0.3), int(frame_width * 0.2), int(frame_height * 0.4)),

        (int(frame_width * 0.3), int(frame_height * 0.6), int(frame_width * 0.4), int(frame_height * 0.3)),

        (int(frame_width * 0.1), int(frame_height * 0.1), int(frame_width * 0.2), int(frame_height * 0.1))
    ]

    colors = [
        (0, 0, 255),
        (0, 255, 0),
        (255, 0, 0)
    ]

    object_labels = ["Shuttle/Rocket", "Launch Pad", "Timer/Logo"]

   multi_tracker = cv2.legacy.MultiTracker_create()
```

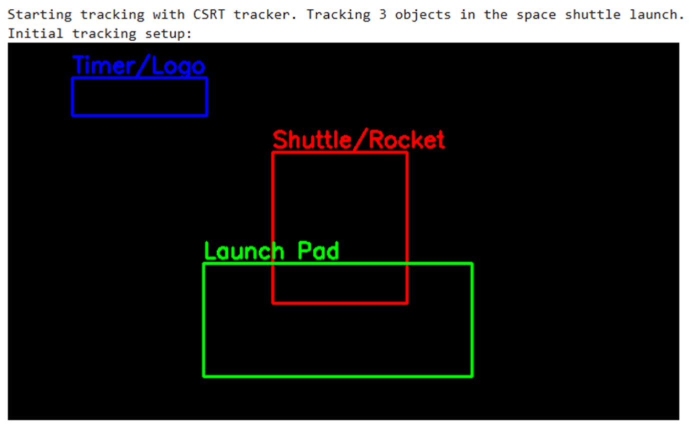
```
    for bbox in bboxes:
        if tracker_type == "CSRT":
            tracker = cv2.legacy.TrackerCSRT_create()
        elif tracker_type == "KCF":
            tracker = cv2.legacy.TrackerKCF_create()
        elif tracker_type == "MOSSE":
            tracker = cv2.legacy.TrackerMOSSE_create()
        else:
            tracker = cv2.legacy.TrackerKCF_create()

        multi_tracker.add(tracker, frame, bbox)

    print(f"Starting tracking with {tracker_type} tracker. Tracking {len(bboxes)} objects in the space
shuttle launch.")

    success, boxes = multi_tracker.update(frame)
    first_frame_with_boxes = frame.copy()

    if success:
        for i, box in enumerate(boxes):
            x, y, w, h = [int(v) for v in box]
            cv2.rectangle(first_frame_with_boxes, (x, y), (x + w, y + h), colors[i], 2)
            cv2.putText(first_frame_with_boxes, object_labels[i], (x, y - 5),
                        cv2.FONT_HERSHEY_SIMPLEX, 0.7, colors[i], 2)

    print("Initial tracking setup:")
    cv2_imshow(first_frame_with_boxes)

    print("Processing video frames...")

    key_frame_numbers = [
        1,
        300,
        600,
        900,
        1200,
        1500,
        1800,
        2100
    ]

    frame_count = 0
    shown_frames = 0

    while cap.isOpened() and shown_frames < len(key_frame_numbers):
        ret, frame = cap.read()

        if not ret:
            print("End of video")
            break

        frame_count += 1

        if frame_count not in key_frame_numbers:
            continue

        success, boxes = multi_tracker.update(frame)

       if success:
            for i, box in enumerate(boxes):
                x, y, w, h = [int(v) for v in box]
                cv2.rectangle(frame, (x, y), (x + w, y + h), colors[i], 2)
                cv2.putText(frame, object_labels[i], (x, y - 5),
                            cv2.FONT_HERSHEY_SIMPLEX, 0.7, colors[i], 2)

        print(f"Frame {frame_count} (approx. {frame_count/30:.1f} seconds into video):")
        cv2_imshow(frame)
        shown_frames += 1

        time.sleep(0.5)

    cap.release()
    print("Tracking complete.")
```

```
    print("Note: For more precise tracking of the rocket during launch, specialized detection algorithms would typically be used.")

track_space_shuttle_launch(video_path, "CSRT")  # CSRT tracker is more accurate for this type of video
```

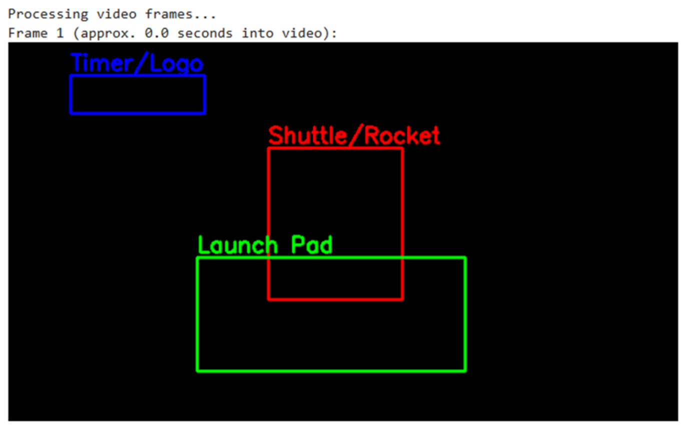
OUTPUT:



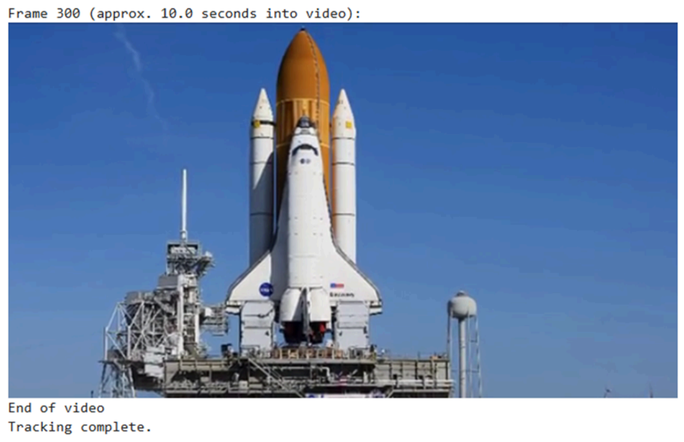
```
Video dimensions: 640x360
First frame of the space shuttle launch video:
```



```
Starting tracking with CSRT tracker. Tracking 3 objects in the space shuttle launch.
Initial tracking setup:
```



```
Processing video frames...
Frame 1 (approx. 0.0 seconds into video):
```



```
Frame 300 (approx. 10.0 seconds into video):
End of video
Tracking complete.
```

RESULT:
Thus the program has been successfully implemented and verified.