```python
import cv2
import numpy as np

def create_background_model(video_path, num_frames_for_bg):
    cap = cv2.VideoCapture(video_path)
    frames = []

    while len(frames) < num_frames_for_bg:
        ret, frame = cap.read()
        if not ret:
            break
        frames.append(frame)

    cap.release()

    if len(frames) < num_frames_for_bg:
        raise ValueError("Not enough frames to create the background model.")

    bg_model = np.mean(frames, axis=0).astype(dtype=np.uint8)
    return bg_model

def process_video(video_path, bg_model, TL, AL, AH):
    cap = cv2.VideoCapture(video_path)
    frame_width = int(cap.get(cv2.CAP_PROP_FRAME_WIDTH))
    frame_height = int(cap.get(cv2.CAP_PROP_FRAME_HEIGHT))
    fps = cap.get(cv2.CAP_PROP_FPS)

    fourcc = cv2.VideoWriter_fourcc(*'MJPG')
    original_output =
cv2.VideoWriter("C:\\Users\\zezva\\Desktop\\HW#6\\original_output.avi", fourcc,
fps, (frame_width, frame_height))
    difference_output =
cv2.VideoWriter("C:\\Users\\zezva\\Desktop\\HW#6\\difference_output.avi", fourcc,
fps, (frame_width, frame_height))
    detection_output =
cv2.VideoWriter("C:\\Users\\zezva\\Desktop\\HW#6\\detection_output.avi", fourcc,
fps, (frame_width, frame_height))

    while cap.isOpened():
        ret, frame = cap.read()
        if not ret:
            break

        # Calculate the difference between the background model and the current frame
        diff = cv2.absdiff(frame, bg_model)
```

```
        gray_diff = cv2.cvtColor(diff, cv2.COLOR_BGR2GRAY)

        # Threshold the difference to get the foreground mask
        _, fg_mask = cv2.threshold(gray_diff, TL, 255, cv2.THRESH_BINARY)

        # Apply morphological operations to clean up the mask
        kernel = np.ones((5, 5), np.uint8)
        fg_mask = cv2.morphologyEx(fg_mask, cv2.MORPH_OPEN, kernel)
        fg_mask = cv2.morphologyEx(fg_mask, cv2.MORPH_CLOSE, kernel)

        # Find connected components to get the ROIs
        num_labels, labels, stats, centroids =
cv2.connectedComponentsWithStats(fg_mask, connectivity=8)

        # Filter ROIs based on area
        rois = []

        for i in range(1, num_labels):  # Start from 1 to skip the background
            area = stats[i, cv2.CC_STAT_AREA]
            if AL < area < AH:
                x, y, w, h = stats[i, cv2.CC_STAT_LEFT], stats[i, cv2.CC_STAT_TOP],
stats[i, cv2.CC_STAT_WIDTH], stats[i, cv2.CC_STAT_HEIGHT]
                rois.append((x, y, w, h))
                # Draw rectangle on the original frame
                cv2.rectangle(frame, (x, y), (x + w, y + h), (0, 255, 0), 2)

        # Write frames to output videos
        original_output.write(frame)
        difference_output.write(cv2.cvtColor(gray_diff, cv2.COLOR_GRAY2BGR))
        detection_output.write(cv2.cvtColor(fg_mask, cv2.COLOR_GRAY2BGR))

        # Display frames for debugging (optional)
        cv2.imshow('Original', frame)
        cv2.imshow('Difference', gray_diff)
        cv2.imshow('Detection', fg_mask)

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

    cap.release()
    original_output.release()
    difference_output.release()
    detection_output.release()
    cv2.destroyAllWindows()
    print("Processing completed and videos saved successfully.")
```

```
# Paths
video_path = "C:\\Users\\zezva\\Desktop\\HW#6\\1.mp4"

# Parameters
num_frames_for_bg = 32  # Number of frames to create the background model
TL = 80  # Threshold value for binarizing the difference image
AL = 3000  # Lower limit for the area of ROI
AH = 50000  # Upper limit for the area of ROI

# Create the background model
bg_model = create_background_model(video_path, num_frames_for_bg)

# Process the video with adjustable threshold and area limits
process_video(video_path, bg_model, TL, AL, AH)
```