Task: Enhance the created video by adding text on each frame

```
import cv2
import numpy as np

# Input video file path
video_path = "/content/WhatsApp Video 2025-03-14 at 12.12.08 PM.mp4"
output_video_path = "/content/final_output.mp4"  # Final output video

# Load video
cap = cv2.VideoCapture(video_path)

# Check if video is opened correctly
if not cap.isOpened():
    print(f"Error: Could not open video file '{video_path}'.")
    exit()

# Get video properties
frame_width = int(cap.get(cv2.CAP_PROP_FRAME_WIDTH))
frame_height = int(cap.get(cv2.CAP_PROP_FRAME_HEIGHT))
fps = int(cap.get(cv2.CAP_PROP_FPS))

# Define VideoWriter to save the processed video
fourcc = cv2.VideoWriter_fourcc(*'mp4v')  # Codec for MP4 format
out = cv2.VideoWriter(output_video_path, fourcc, fps, (frame_width * 2, frame_height))

print("Processing video...")

frame_count = 0  # Frame counter

while True:
    # Read frame
    ret, frame = cap.read()
    if not ret:
        print("End of video reached.")
        break

    frame_count += 1  # Increment frame count

    # Processing Techniques
    # 1. Convert to Grayscale
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)

    # 2. Apply Gaussian Blurring
    blurred = cv2.GaussianBlur(gray, (15, 15), 0)

    # 3. Edge Detection
    edges = cv2.Canny(blurred, 50, 150)
    edges_bgr = cv2.cvtColor(edges, cv2.COLOR_GRAY2BGR)  # Convert edges to BGR for stacking

    # 4. Bitwise AND Operation (Blended Image)
    bitwise_and = cv2.bitwise_and(frame, frame, mask=edges)

    # Combine Original & Processed Frames Side by Side
    processed_frame = np.hstack((frame, edges_bgr))
```

```
    # Add Text Overlay
    text = f"Frame: {frame_count}"
    cv2.putText(processed_frame, text, (50, 50), cv2.FONT_HERSHEY_SIMPLEX, 1, (0, 255, 0), 2)
    cv2.putText(processed_frame, "Processing Video", (50, 100), cv2.FONT_HERSHEY_SIMPLEX, 1, (0, 0, 255),

    # Write the processed frame to the output video
    out.write(processed_frame)

# Release resources
cap.release()
out.release()

print(f"✅ Video processing completed! Output saved as '{output_video_path}'.")
```

```
Processing video...
End of video reached.
✅ Video processing completed! Output saved as '/content/final_output.mp4'.
```

Task: Create a new video by writing processed frames to a video file using OpenCV.

```
import cv2
import numpy as np

# Input video file path
video_path = "/content/WhatsApp Video 2025-03-14 at 12.12.08 PM.mp4"
output_video_path = "/content/output_video.mp4"  # Processed video output

# Load video
cap = cv2.VideoCapture(video_path)

# Check if video is opened correctly
if not cap.isOpened():
    print(f"Error: Could not open video file '{video_path}'.")
    exit()

# Get video properties
frame_width = int(cap.get(cv2.CAP_PROP_FRAME_WIDTH))
frame_height = int(cap.get(cv2.CAP_PROP_FRAME_HEIGHT))
fps = int(cap.get(cv2.CAP_PROP_FPS))

# Define VideoWriter to save the processed video
fourcc = cv2.VideoWriter_fourcc(*'mp4v')  # Codec for MP4 format
out = cv2.VideoWriter(output_video_path, fourcc, fps, (frame_width * 2, frame_height))

print("Processing video...")

while True:
    # Read frame
    ret, frame = cap.read()
    if not ret:
        print("End of video reached.")
        break
```

```
    # Processing Techniques
    # 1. Convert to Grayscale
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)

    # 2. Apply Gaussian Blurring
    blurred = cv2.GaussianBlur(gray, (15, 15), 0)

    # 3. Edge Detection
    edges = cv2.Canny(blurred, 50, 150)
    edges_bgr = cv2.cvtColor(edges, cv2.COLOR_GRAY2BGR)  # Convert edges to BGR for stacking

    # 4. Bitwise AND Operation (Blended Image)
    bitwise_and = cv2.bitwise_and(frame, frame, mask=edges)

    # Combine Original & Processed Frames Side by Side
    processed_frame = np.hstack((frame, edges_bgr))

    # Write the processed frame to the output video
    out.write(processed_frame)

# Release resources
cap.release()
out.release()

print(f"✅ Processing completed! Output saved as '{output_video_path}'.")
```

```
Processing video...
End of video reached.
✅ Processing completed! Output saved as '/content/output_video.mp4'.
```

Process images of a video using OpenCV

Implement image processing techniques on frames extracted from a video using OpenCV

Adaptive Threshold

Smoothing

Edge Detection

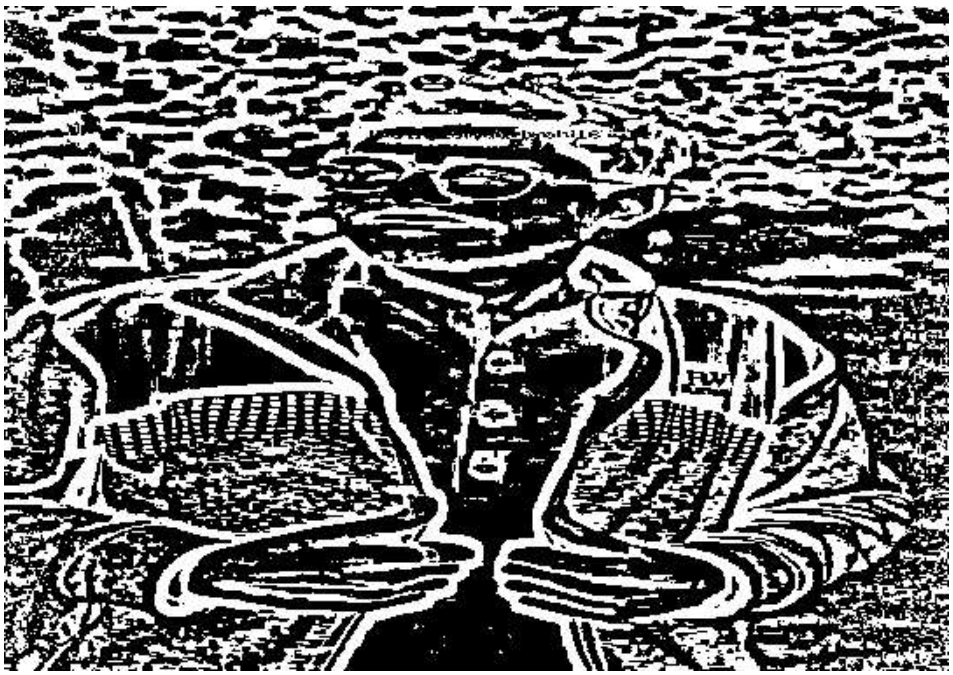
Bitwise Operations

```
# Importing the necessary libraries
import cv2
import numpy as np
from google.colab.patches import cv2_imshow  # Use this for displaying images in Colab

# Creating a VideoCapture object to read the video
video_path = "/content/WhatsApp Video 2025-03-14 at 12.12.08 PM.mp4"
cap = cv2.VideoCapture(video_path)

# Check if video file opened successfully
if not cap.isOpened():
    print(f"Error: Could not open video file '{video_path}'")
else:
    print("Video loaded successfully. Processing first frame...")
```

```
# Read only the first frame
ret, frame = cap.read()

if ret:
    # Resize frame
    frame = cv2.resize(frame, (540, 380), interpolation=cv2.INTER_CUBIC)

    # Convert BGR to grayscale
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)

    # Apply adaptive thresholding
    thresholded = cv2.adaptiveThreshold(gray, 255, cv2.ADAPTIVE_THRESH_MEAN_C,
                                        cv2.THRESH_BINARY_INV, 11, 2)

    # Apply Gaussian Blur for smoothing
    blurred = cv2.GaussianBlur(frame, (15, 15), 0)

    # Apply Edge Detection (Canny)
    edges = cv2.Canny(gray, 100, 200)
    edges_colored = cv2.cvtColor(edges, cv2.COLOR_GRAY2BGR)  # Convert to BGR for display

    # Apply Bitwise NOT operation
    bitwise_not = cv2.bitwise_not(frame)

    # Display the frames using cv2_imshow
    print("Original Frame:")
    cv2_imshow(frame)

    print("Thresholded Frame:")
    cv2_imshow(thresholded)

    print("Smoothing (Gaussian Blur):")
    cv2_imshow(blurred)

    print("Edge Detection (Canny):")
    cv2_imshow(edges_colored)

    print("Bitwise NOT Operation:")
    cv2_imshow(bitwise_not)

else:
    print("Error: Could not read the first frame.")

# Release video resource
cap.release()
print("Processing completed.")
```

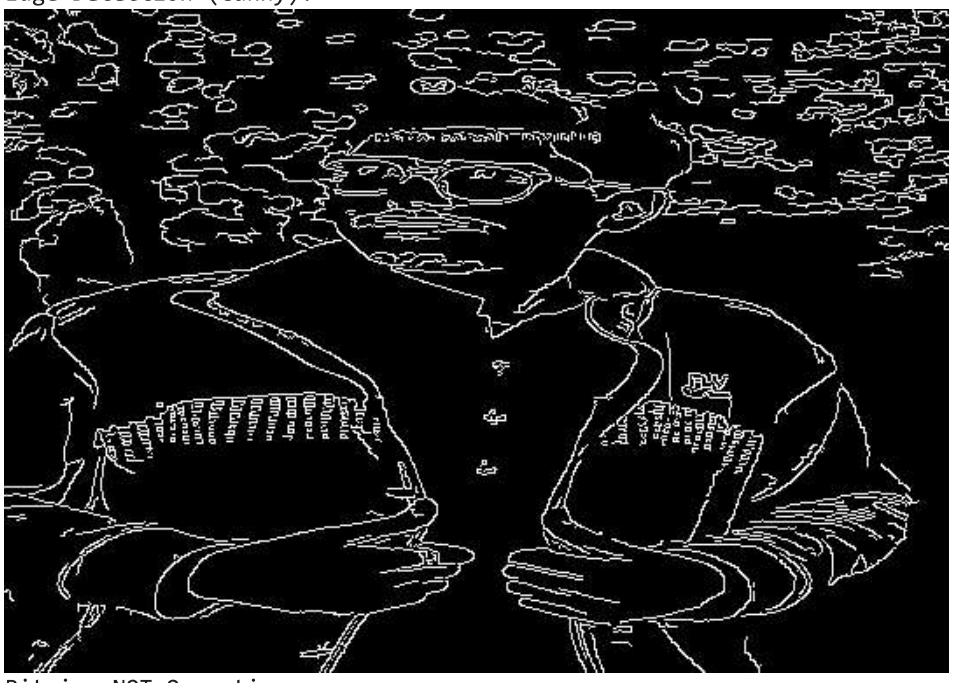
Video loaded successfully. Processing first frame...
Original Frame:

Thresholded Frame:



Smoothing (Gaussian Blur):

Edge Detection (Canny):



Bitwise NOT Operation: