Exception Handling

```
import cv2
def process_video(video_file):
    try:
        # Open the video file
        cap = cv2.VideoCapture(video_file)

        # Check if the video file opened successfully
        if not cap.isOpened():
            raise Exception("Error: Could not open the video file.")

        while True:
            # Read a frame from the video
            ret, frame = cap.read()

            # Check if the video has ended
            if not ret:
                break

            # Process the frame (you can add your video processing logic here)
            processed_frame = frame

            # Display the processed frame
            cv2.imshow('Processed Video', processed_frame)

            # Press 'q' to exit the video
            if cv2.waitKey(25) & 0xFF == ord('q'):
```

```
            break

        # Release the video capture object and close the display window
        cap.release()
        cv2.destroyAllWindows()

    except Exception as e:
        # Handle and log the exception
        print(f"Error: {e}")

if __name__ == "__main__":
    video_file = 'your_video.mp4'
    try:
        process_video(video_file)
    except KeyboardInterrupt:
        # Handle user interruption (e.g., when pressing Ctrl+C)
        print("Video processing interrupted by the user.")
```

In this program:

The process_video function opens a video file using OpenCV and processes the frames. It catches exceptions such as failure to open the video file or errors during processing.

If an exception occurs, it is caught in the except block, and an error message is displayed. You can customize the exception handling logic as needed.

The program also handles user interruption (e.g., pressing Ctrl+C) using a Keyboard Interrupt exception, allowing you to exit the video processing gracefully.

Replace ‘your_video.mp4’ with the actual path to your video file. You can add your video processing logic inside the loop where frames are processed.

This is a basic example of exception handling in a video-related program. Depending on the complexity of your video processing tasks, you may need more robust error handling and logging mechanisms to address specific issues that can arise during video processing.