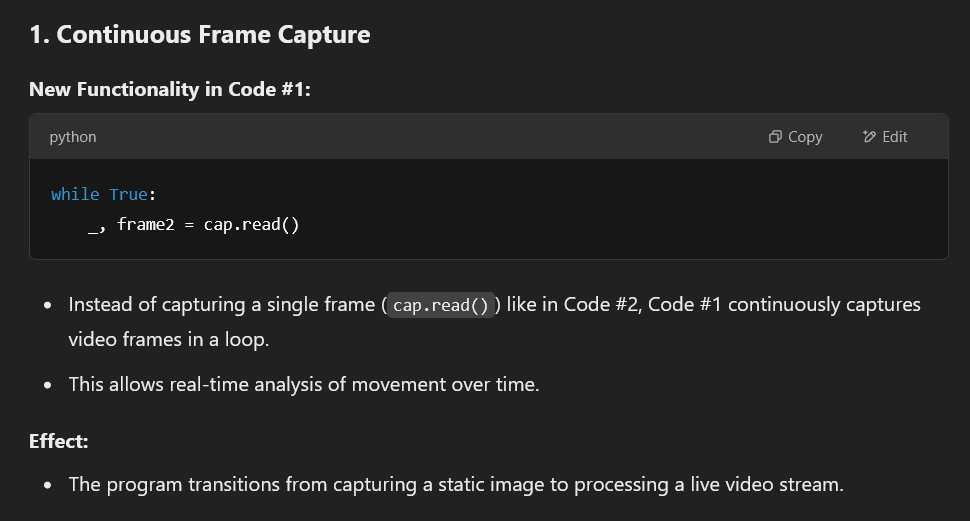
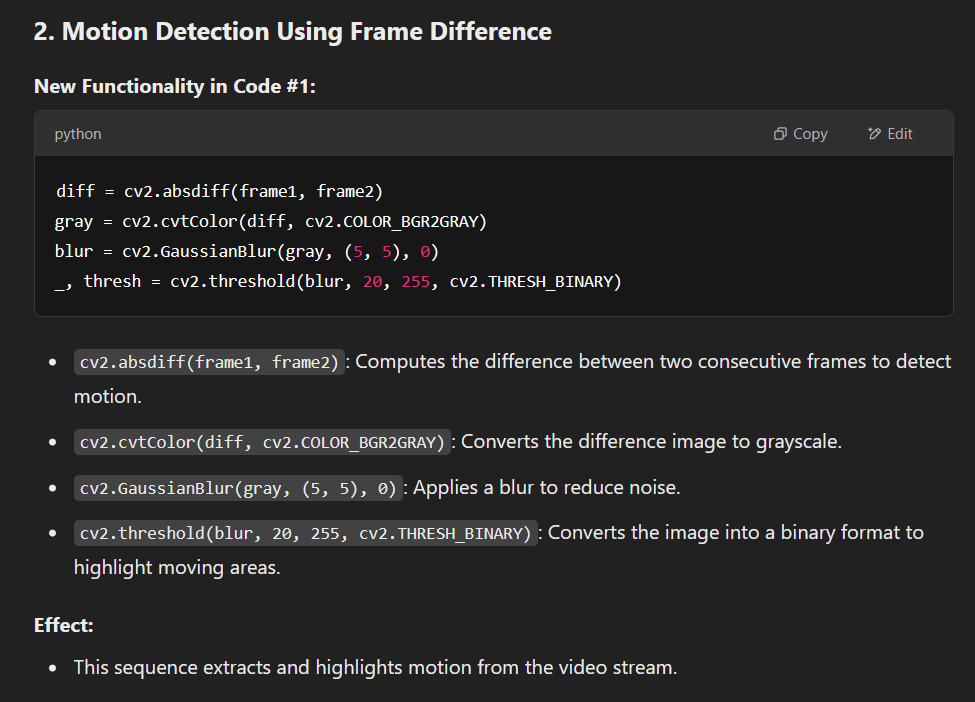
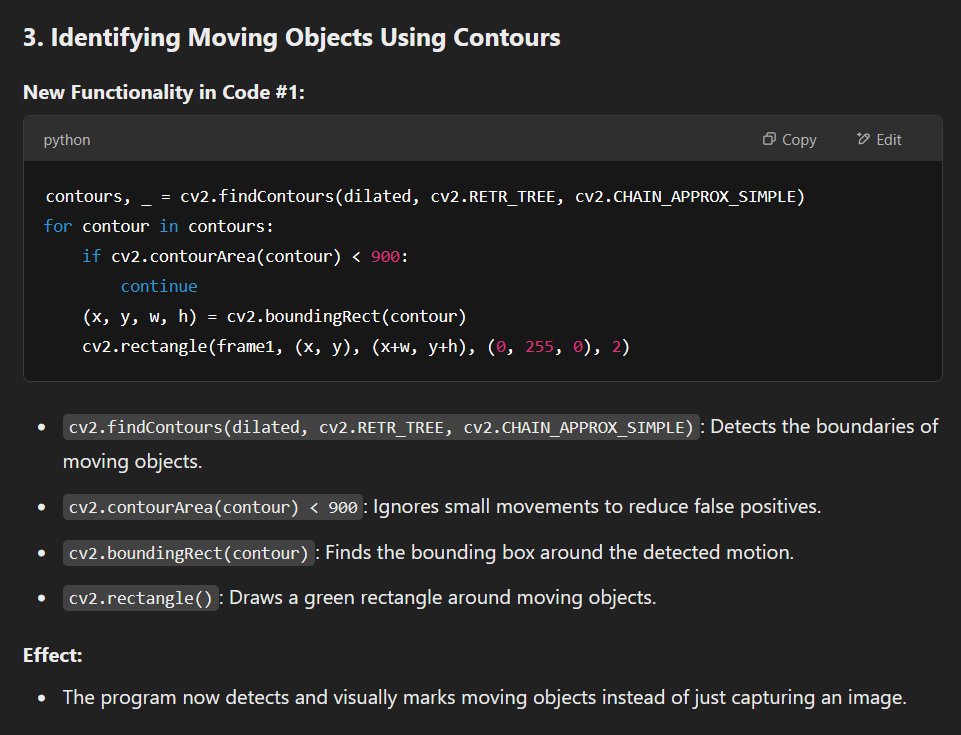
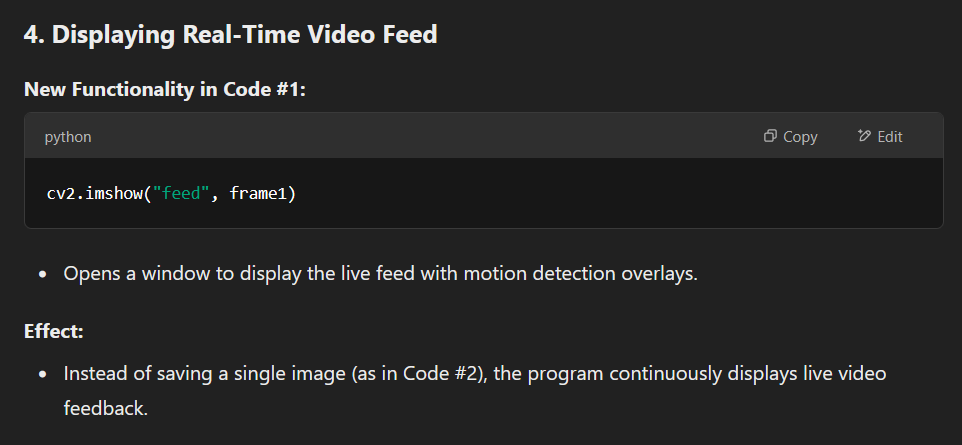
1. Identify and explain the additional functionalities introduced in Code #2. How do these changes transform the program from a simple image capture to a movement detection system?

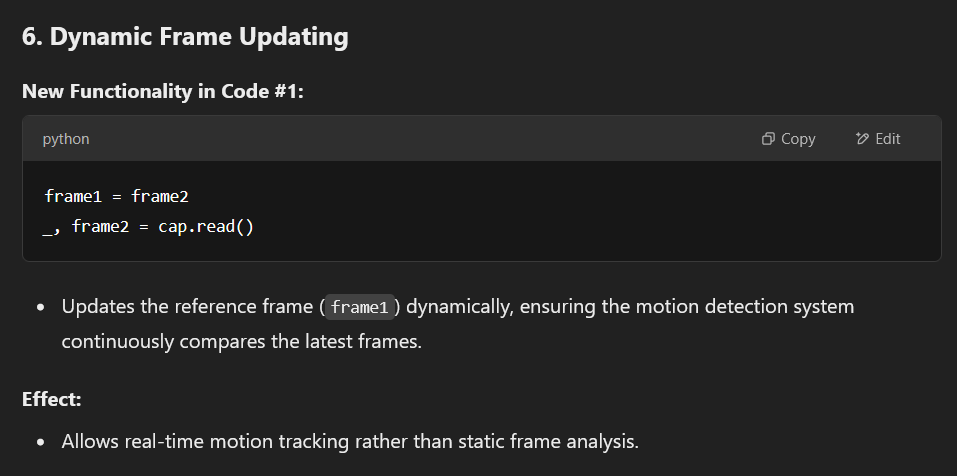


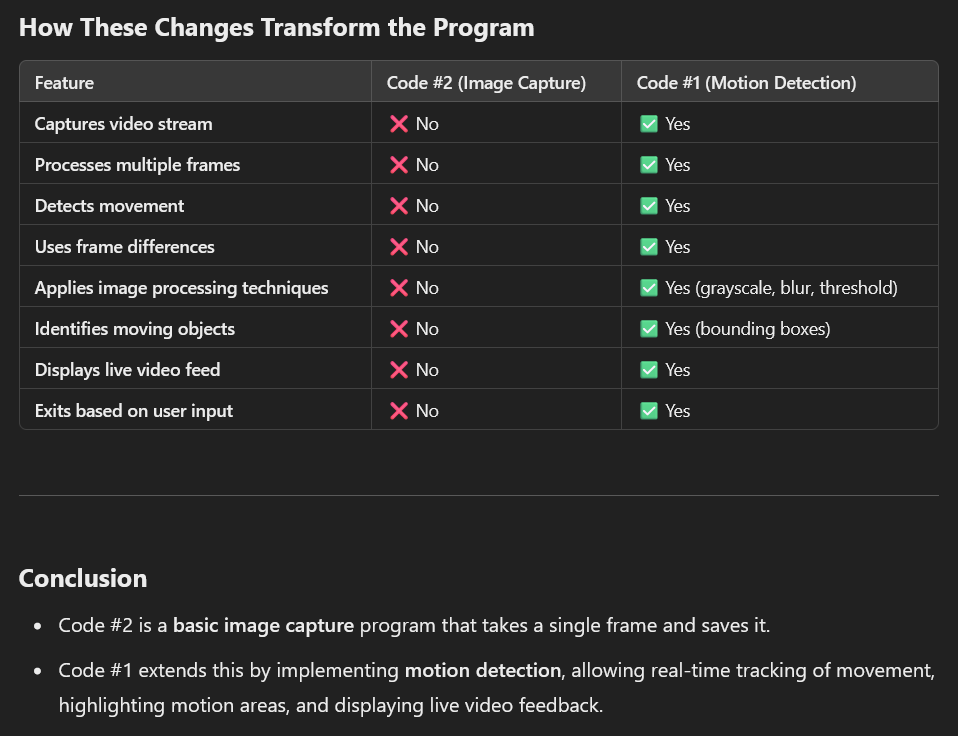




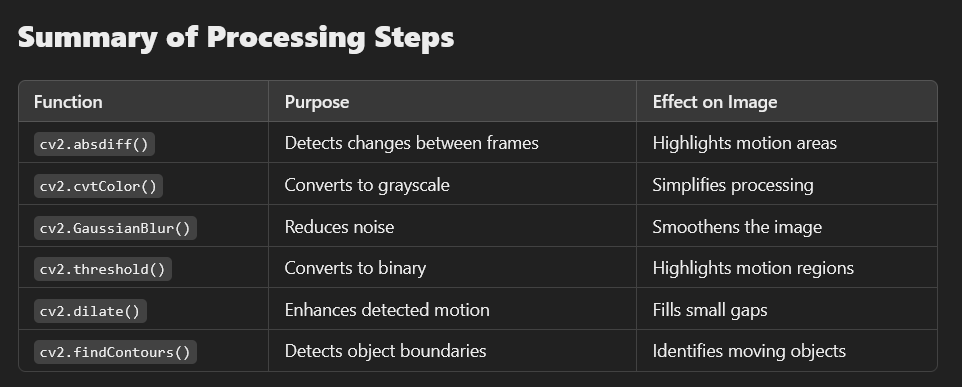




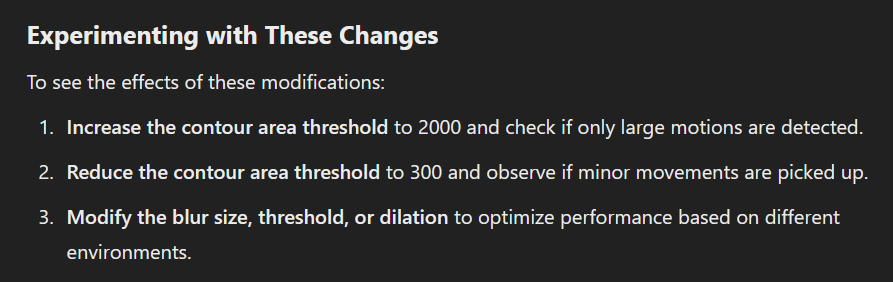




1. Several new OpenCV functions are used (like cv2.absdiff, cv2.cvtColor, cv2.GaussianBlur, cv2.threshold, cv2.dilate, and cv2.findContours). Research each of these functions and understand their role in processing the video frames for movement detection.



1. The program uses specific conditions (such as contour area) to decide when to draw rectangles and indicate movement. Experiment with these parameters to see how they affect the accuracy and sensitivity of movement detection.



1. Loop Mechanics and Video Processing: Analyze the role of the while loop in the 2nd Code for continuous video capture and processing. How does this looping mechanism differ from the single capture approach in the 1st Code, especially in terms of real-time processing and movement detection?

while Loop in the First Code (Motion Detection)

**How the Loop Enables Continuous Video Processing**

1. **Reads frames continuously** using cap.read(), ensuring new data is processed in real time.
2. **Compares two consecutive frames (frame1 and frame2)** to detect movement.
3. **Processes each frame dynamically**, applying image processing techniques (grayscale conversion, blurring, thresholding, dilation, and contour detection).
4. **Draws rectangles on moving objects**, updating the displayed frame dynamically.
5. **Waits for the user to press 'q' to stop**, allowing the loop to run indefinitely.



1. Consider aspects like improving the accuracy of movement detection, optimizing performance, or adding new features (like recording video when movement is detected).

