OpenCV Project - RN(Android)

Developing an app with React Native(Android) and OpenCV for continuous image processing, as you described, involves several key steps. Here's a high-level overview of the process and some estimates on the complexity and time requirements.

Steps for Development:

1. Setup React Native Environment

- **Task**: Install React Native CLI, initialise a new React Native project, and set up the development environment for Android (and iOS if needed).
- Estimate: 1 day

2. Integrate OpenCV with React Native

- **Task**: Since OpenCV is a C++ library, you need to integrate it with your React Native project. This involves using native modules to bridge OpenCV code with React Native. You may need to write some Java/Kotlin (for Android) code to use OpenCV functions.
- **Resources**: Look for existing projects or tutorials on integrating OpenCV with React Native. You might find libraries like `react-native-opencv3` or similar.
- Estimate: 2-4 weeks, depending on complexity and existing solutions

3. Implement Image/Video Capture Functionality

- **Task**: Implement functionality to capture video or photos continuously using the camera. React Native libraries like `react-native-camera` can be used for camera access.
- Estimate: 1 week

4. Develop Image Processing Algorithms with OpenCV

- **Task**: Implement the image processing logic to detect pallets, their orientation, shapes on the photo, and barcodes within those shapes. This involves using OpenCV functions for image analysis, shape detection, barcode recognition, and corner detection.
- Estimate: 4-6 weeks, depending on the complexity of detection algorithms and the learning curve with OpenCV

5. Calibration and Coordinate System Setup

- **Task**: Implement logic to use detected pallet sizes to calibrate the coordinate system for further processing.

- Estimate: 1 week

6. API Integration for Verification

- **Task**: Implement functionality to compile the detected objects into a list (with sizes, coordinates, and recognized barcodes) and send this list to a specified API for verification.
- Estimate: 1 week

7. Testing and Optimization

- **Task**: Test the app extensively to ensure it can process images/videos without operator interaction, optimise performance, and fix any bugs. This includes testing on multiple devices to ensure compatibility and performance optimization.
- Estimate: 2-4 weeks

8. Documentation and Cleanup

- **Task**: Write documentation for the app, including how to set up, use, and contribute to the project. Clean up the code and ensure it follows best practices.
- Estimate: 1 week

Total Estimated Time: 3-4 months

It's also worth noting that testing and optimization can take longer than anticipated, especially for performance-intensive applications like real-time image processing. Consider adding buffer time for unexpected challenges.