**Master Development Plan for Image Processing Module**

**1. Input Data (60Hz Image Frames)**

* **Objective:** Capture real-time frames from the smartphone’s rolling shutter camera at 60Hz to track user hand movements using LED-equipped gloves.
* **Details:**
  + Continuous image stream from the Camera2 API.
  + Ensure stable, high-quality input for accurate gesture tracking, mitigating artifacts from the rolling shutter.

**2. Pre-Processing (Image Enhancement)**

* **Objective:** Prepare raw frames for feature extraction by enhancing clarity and reducing noise.
* **Sub-Steps:**
  + **Denoising:** Remove ambient lighting interference and noise.
    - Convert images to grayscale for simplified processing.
  + **Sharpening: Make LED features more distinct.**
    - Apply a brightness threshold to isolate light sources from background data.
  + **Privacy Filtering:** Filter out irrelevant background information, focusing only on the LED.
* **Tools:** Real-time processing using OpenCV.

**3. Feature Extraction**

* **Objective:** Track the LED in each frame, extracting relevant features for movement tracking.
* **Sub-Steps:**
  + **Signature Extraction:** Identify the LED from multiple light sources by leveraging its PWM signature.
  + **Edge Detection:** Highlight the LED’s position using cv2.Canny for edge detection.
  + **Center Detection:** Calculate the LED’s position via the center of mass.
    - **Output:** Center data (X, Y).
  + **Blob Detection:** Track relative depth by analyzing the change in the LED’s surface area.
    - **Output:** Depth data (Z).
  + **Time Data:** Capture frame-specific time data using the Camera2 API.
    - **Output:** Frame time data (T).
  + **Final Output:** Combine (X, Y, Z, T) into a NumPy array and append to a data list for further processing.
* **Tools:** OpenCV for detection algorithms and NumPy for data handling.

**4. Path Reconstruction**

* **Objective:** Rebuild the 3D path of the LED using extracted features from each frame.
* **Sub-Steps:**
  + **3D Position Calculation:** Use (X, Y, Z, T) data to create a continuous 3D path representing the LED’s movement.
  + **Time Synchronization:** Ensure the 3D path is synchronized with the frame time for accurate tracking.
* **Tools:** NumPy to manage the 4D array (X, Y, Z, T) and compute the trajectory.

**5. Path Smoothing (Post-Processing)**

* **Objective:** Smooth the reconstructed 3D path to eliminate irregularities and ensure fluid movements.
* **Sub-Steps:**
  + **Moving Average Filter:** Apply to create smoother, continuous movement.
  + **Spline Interpolation:** Use splines to generate smooth curves that reflect realistic LED motion.
* **Tools:** FilterPy for filtering, SciPy for spline interpolation.

**6. Output Data**

* **Objective:** Finalize the data for further machine learning analysis.
* **Details:**
  + Output the processed data as a 4D NumPy array (X, Y, Z, T).
  + This dataset is ready for use by the Machine Learning Module for gesture recognition or further analysis.

A diagram of a machine

Description automatically generated