Design and Implementation of a Monocular Measurement System on Android using Dynamic Camera Intrinsics

Trung Duc Bui
School of Coumputing and Information
University of Pittsburgh
Pittsburgh, PA 15213
`trungbuiducbuiduc@gmail.com`

December 14, 2025

Abstract

Estimating real-world object dimensions from smartphone images is a fundamental problem in computer vision, with applications in e-commerce, logistics, and interior design. Commercial solutions typically rely on augmented reality (AR) frameworks such as ARCore or ARKit, which provide metric scale through SLAM-based 3D reconstruction. However, these frameworks act as "black boxes" that obscure the underlying geometric principles, impose operational constraints, and—critically—are not available on all devices, as ARCore certification covers only a subset of Android smartphones. This paper presents the design and implementation of a *traditional, AR-free measurement pipeline* for Android that operates independently of proprietary AR SDKs, enabling metric measurement on any device with Camera2 API support. The proposed system implements a *Dynamic Intrinsics* mechanism, recalculating the camera matrix K per-frame based on `SCALER_CROP_REGION` metadata to maintain accuracy across continuous digital zoom levels. By fusing optical data with IMU-derived orientation, the system supports classical measurement models including ground-plane homography, planar object rectification, and single-view metrology. The application features a dual-mode design: a simplified *User Mode* for practical measurement tasks, and a dedicated *Researcher Mode* that exposes internal parameters and enables experimental analysis. To validate the proposed approach, we conduct systematic experiments using two devices: a non-ARCore device (Xiaomi Poco X6 Pro) for the traditional pipeline and an ARCore-certified device (Google Pixel 7 Pro) for baseline comparison. Results demonstrate that classical computer vision techniques, when properly controlled for intrinsic parameter variations, can achieve accuracy comparable to AR-based approaches while offering superior transparency, interpretability, and broader device compatibility.

***K*eywords** Single-View Metrology · Dynamic Camera Intrinsics · Android Camera2 API · Digital Zoom Compensation · Pinhole Camera Model · Ground-Plane Homography · ARCore Comparison · Mobile Measurement

1 Introduction

Estimating real-world object dimensions from images captured by consumer smartphones is a fundamental problem in computer vision, with practical applications spanning e-commerce (product sizing), logistics (package dimensioning), interior design (room measurement), and augmented reality. Over the past decade, mobile devices have witnessed remarkable advances in optical hardware, equipping modern smartphones with high-resolution sensors, optical image stabilization (OIS), and low-level hardware access via APIs such as Android's Camera2. While this evolution positions smartphones as potential precision instruments for metrology, realizing a mathematically transparent measurement tool remains a significant challenge.

Commercial measurement solutions typically rely on augmented reality (AR) frameworks such as ARCore or ARKit, which provide metric scale through visual-inertial odometry (VIO) and simultaneous localization and mapping (SLAM). While these approaches can achieve practical accuracy, they exhibit fundamental limitations: they act as "black boxes" that obscure the underlying geometric principles, impose operational constraints (requiring camera motion for initialization, texture-rich environments), and—importantly—are not available on all Android devices. As of 2024, ARCore supports only a curated list of certified devices, leaving many capable smartphones without access to AR-based measurement. For researchers and learners seeking to understand, validate, or modify the measurement models, such opacity and limited availability represent significant barriers.

A particularly underexplored challenge within this domain is the *dynamic nature of camera intrinsics during digital zoom operations*. Unlike optical zoom which physically adjusts lens elements, digital zoom fundamentally alters the effective camera matrix through sensor cropping and resampling. The intrinsic parameters—focal length and principal point—change with each zoom level, yet standard measurement applications often assume a fixed camera matrix, leading to systematic errors that compound at higher magnification. This problem is exacerbated by the fact that few existing systems leverage the rich per-frame metadata available through the Camera2 API, particularly `SCALER_CROP_REGION`.

To address these challenges, we present a configurable monocular measurement system for Android that operates independently of third-party AR frameworks. The proposed system targets two distinct audiences:

1. **End-users**, who require a simple, accessible tool to measure objects quickly and accurately; and
2. **Researchers and learners**, who seek to understand, inspect, and experiment with the underlying camera model and geometric assumptions.

The system implements a *Dynamic Intrinsics* mechanism that continuously recalculates the camera matrix K based on real-time `SCALER_CROP_REGION` metadata, maintaining measurement accuracy across continuous zoom levels. By fusing optical data with IMU-derived orientation, the system supports classical measurement models—including ground-plane homography, planar object rectification, and single-view metrology—through a transparent, fully controllable pipeline.

Uniquely, the application features a dual-mode design: a simplified *User Mode* for practical measurement tasks, and a dedicated *Researcher Mode* that exposes all internal parameters, supports toggling of processing options (undistortion, edge snapping, rectification, multi-frame averaging), and enables export of detailed logs for analysis. To validate the proposed approach against the state of practice, we conduct systematic experiments using two devices: a non-ARCore-certified device (Xiaomi Poco X6 Pro) running the traditional pipeline, and an ARCore-certified device (Google Pixel 7 Pro) for baseline comparison. This setup directly addresses the practical question: *can a traditional, Camera2-based pipeline achieve comparable accuracy to ARCore on devices where AR frameworks are unavailable?*

The main contributions of this work are:

1. A **camera-model-driven measurement pipeline** for Android that uses Camera2 metadata (intrinsic calibration, active array size, dynamic crop region) to reconstruct a per-frame intrinsic matrix K_{out}, maintaining accuracy across continuous digital zoom levels.
2. A **dual-mode application design** with a user-friendly interface for end-users and a dedicated Researcher Mode that exposes internal parameters, processing steps, and experimental controls for educational and research purposes.
3. A **systematic experimental evaluation** comparing the traditional pipeline with an ARCore-based baseline using two devices (non-ARCore vs. ARCore-certified), demonstrating that classical techniques can achieve comparable accuracy even on devices where AR frameworks are unavailable.

The remainder of this paper is organized as follows. Section 2 reviews related work on camera calibration, single-view metrology, and existing measurement approaches. Section 3 describes the proposed system architecture and measurement modules. Section 4 presents the experimental setup and methodology. Section 5 reports results and discussion. Section **??** concludes the paper, and Section **??** outlines directions for future work.

2 Literature Review

This section establishes the theoretical foundations of the proposed system, covering the geometric modeling of cameras, the specific challenges of digital image formation on mobile devices, and existing techniques for extracting metric dimensions from 2D images.

2.1 Pinhole Camera Model and Calibration

2.1.1 The Pinhole Model

The pinhole camera model serves as the fundamental geometric abstraction for image formation in computer vision [5]. This idealized model assumes that all light rays pass through a single point (the optical center) before striking the image plane. Under this model, a 3D world point $\mathbf{X}_w = (X, Y, Z)^T$ is projected onto the 2D image plane at pixel coordinates (u, v) through a linear transformation in homogeneous coordinates:

$$s \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = K[R \mid t] \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \tag{1}$$

where s is an arbitrary scale factor arising from the use of homogeneous coordinates. The 3×4 projection matrix $P = K[R \mid t]$ decomposes into two components: the *extrinsic parameters* $[R \mid t]$ describing the camera's pose (rotation $R \in SO(3)$ and translation $t \in \mathbb{R}^3$) in world coordinates, and the *intrinsic matrix* K encoding the internal camera geometry:

$$K = \begin{bmatrix} f_x & s & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \tag{2}$$

Here, f_x and f_y represent the focal length expressed in pixel units along the horizontal and vertical axes respectively, (c_x, c_y) denotes the principal point (the intersection of the optical axis with the image plane), and s is the skew coefficient (typically zero for modern sensors with rectangular pixels).

2.1.2 Lens Distortion

Real camera lenses deviate from the ideal pinhole model by introducing geometric distortions. The two primary types are:

Radial distortion causes straight lines to appear curved, particularly near the image periphery. It is modeled as a polynomial function of the distance r from the principal point:

$$\begin{aligned} x_{distorted} &= x(1 + k_1 r^2 + k_2 r^4 + k_3 r^6) \\ y_{distorted} &= y(1 + k_1 r^2 + k_2 r^4 + k_3 r^6) \end{aligned} \tag{3}$$

where (x, y) are normalized image coordinates and k_1, k_2, k_3 are the radial distortion coefficients. Positive coefficients produce barrel distortion (lines curve outward), while negative coefficients produce pincushion distortion (lines curve inward).

Tangential distortion arises from imperfect alignment between the lens elements and the image sensor, causing asymmetric displacement:

$$\begin{aligned} x_{distorted} &= x + [2p_1 xy + p_2(r^2 + 2x^2)] \\ y_{distorted} &= y + [p_1(r^2 + 2y^2) + 2p_2 xy] \end{aligned} \tag{4}$$

where p_1, p_2 are the tangential distortion coefficients.

For accurate metric measurement, these distortions must be corrected through an *undistortion* process that maps observed pixel coordinates to their ideal positions.

2.1.3 Camera Calibration Techniques

Camera calibration is the process of estimating the intrinsic parameters K and distortion coefficients from images of known patterns. Zhang's flexible calibration method [4] revolutionized this process by requiring only a planar calibration target (typically a checkerboard) captured at multiple orientations.

The procedure involves:

1. **Pattern detection**: Locating grid corners in each calibration image using algorithms such as `findChessboardCorners`.

2. **Subpixel refinement**: Improving corner localization accuracy to 0.1 pixels or better using gradient-based methods (`cornerSubPix`).
3. **Homography estimation**: Computing the 3×3 homography H_i mapping pattern coordinates to image coordinates for each view.
4. **Intrinsic extraction**: Exploiting constraints on the absolute conic to solve for K from multiple homographies.
5. **Extrinsic recovery**: Computing rotation and translation for each view given K.
6. **Nonlinear refinement**: Minimizing the total reprojection error over all views using Levenberg-Marquardt optimization.

The reprojection error—the RMS distance between observed corners and their predicted positions—serves as a quality metric; values below 0.5 pixels indicate excellent calibration. More recent calibration targets such as ChArUco boards [12] combine the corner accuracy of checkerboards with the robustness of ArUco fiducial markers, enabling reliable detection even when the pattern is partially occluded or captured at extreme angles.

2.2 Single-View Metrology

Single-view metrology addresses the recovery of metric information from a single 2D image using projective geometry constraints [6]. This is fundamentally an ill-posed problem since depth information is lost during projection; however, additional constraints—such as known geometry, scene planarity, or reference dimensions—can make metric reconstruction tractable.

2.2.1 Homography and Ground-Plane Measurement

When measuring objects that lie on a known plane (e.g., the ground or a table surface), the image-to-world mapping simplifies considerably. For a plane at $Z = 0$ in world coordinates, the projection equation reduces to a 3×3 homography H:

$$s \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = H \begin{bmatrix} X \\ Y \\ 1 \end{bmatrix}, \quad H = K[r_1 \mid r_2 \mid t] \tag{5}$$

where r_1, r_2 are the first two columns of the rotation matrix R and t is the translation. Given knowledge of the camera's intrinsic matrix K, height h above the ground plane, and orientation from an inertial measurement unit (IMU), the homography H can be computed analytically. The inverse homography H^{-1} then maps any image point (u, v) to ground coordinates (X, Y), enabling direct distance computation:

$$d = \sqrt{(X_A - X_B)^2 + (Y_A - Y_B)^2} \tag{6}$$

This approach is particularly effective for measuring horizontal distances on floors, tables, or other flat surfaces when the camera pose is known.

2.2.2 Planar Object Rectification

For planar objects not aligned with the ground (e.g., a wall-mounted poster, a box face, or a sheet of paper), perspective rectification enables measurement. Given four corner points of a rectangular object in the image, a homography H_{rect} can be computed to warp the object to a fronto-parallel (orthogonal) view:

$$H_{rect} : (u_i, v_i) \rightarrow (x'_i, y'_i), \quad i = 1, 2, 3, 4 \tag{7}$$

In the rectified image, the object appears as a rectangle without perspective distortion. If the approximate distance Z from the camera to the object plane is known (e.g., from autofocus metadata or user input), the pixel-to-metric scale can be derived from the pinhole model:

$$L_{real} = \frac{L_{pixels} \cdot Z}{f} \tag{8}$$

where f is the focal length and L_{pixels} is the measured length in the rectified image.

2.2.3 Vanishing Points and Cross-Ratio

For scenes with strong perspective effects—such as tall buildings, corridors, or streets—vanishing point analysis provides powerful geometric constraints [7]. Parallel lines in 3D space converge to a vanishing point (VP) in the image; three mutually orthogonal vanishing points (corresponding to the X, Y, and Z world axes) fully constrain the camera's orientation.

The cross-ratio, a projective invariant preserved under perspective projection, enables height estimation from a single image. Given a vertical structure with its base at ground level, if the camera height h_{cam} is known and the horizon line (connecting horizontal vanishing points) is identified, the ratio of image distances can be used to compute the unknown object height h_{obj}:

$$h_{obj} = h_{cam} \cdot \frac{d(base, top)}{d(base, horizon)} \tag{9}$$

where $d(\cdot, \cdot)$ denotes the signed distance along the vertical image direction.

2.3 Smartphone Camera Systems and the Android Camera2 API

Modern smartphones integrate sophisticated camera systems that present both opportunities and challenges for metrology applications. This section examines the hardware characteristics and software interfaces relevant to measurement accuracy.

2.3.1 Smartphone Camera Hardware

Contemporary smartphone cameras feature:

- **High-resolution sensors**: Typical resolutions of 12–200 megapixels, with pixel sizes ranging from 0.7 to 1.8 μm.
- **Multi-camera arrays**: Wide, ultra-wide, and telephoto lenses with different intrinsic parameters.
- **Optical Image Stabilization (OIS)**: Physical lens displacement to compensate for hand shake, which can affect the effective optical axis.
- **Autofocus systems**: Phase-detection or contrast-detection AF providing focus distance metadata.
- **Inertial Measurement Units (IMU)**: Accelerometers and gyroscopes providing device orientation.

Unlike professional cameras with fixed, well-characterized lenses, smartphone cameras exhibit significant variation in intrinsic parameters across devices and may not provide accurate factory calibration.

2.3.2 The Camera2 API and Intrinsic Metadata

The Android Camera2 API [8] provides low-level access to camera hardware, exposing metadata critical for geometric applications. Key fields include:

- `LENS_INTRINSIC_CALIBRATION`: A 5-element array (f_x, f_y, c_x, c_y, s) providing manufacturer-calibrated intrinsic parameters in pixels, defined relative to the sensor's active array coordinate system.
- `SENSOR_INFO_ACTIVE_ARRAY_SIZE`: The dimensions (W_s, H_s) of the sensor's light-sensitive area in pixels.
- `SENSOR_INFO_PHYSICAL_SIZE`: The physical dimensions of the active array in millimeters, enabling conversion between pixel and metric focal lengths.
- `LENS_RADIAL_DISTORTION`: Radial distortion coefficients (when available; many devices report this as unavailable).
- `LENS_POSE_ROTATION` and `LENS_POSE_TRANSLATION`: The pose of the camera relative to the device's IMU coordinate frame.

The availability and accuracy of these fields vary significantly across devices. For example, our target device (Xiaomi Poco X6 Pro) provides `LENS_INTRINSIC_CALIBRATION` but reports `LENS_RADIAL_DISTORTION` as unavailable.

2.3.3 Digital Zoom and the Dynamic Crop Region

A critical challenge for measurement applications is handling digital zoom. Unlike optical zoom, which physically adjusts lens elements and changes the optical focal length, digital zoom operates entirely in the image processing pipeline by:

1. Cropping a subregion of the sensor (the "crop region")
2. Scaling the cropped region to the output resolution through interpolation

The `SCALER_CROP_REGION` field in `CaptureResult` specifies the active crop rectangle (x_0, y_0, w_c, h_c) for each captured frame. This dynamic cropping fundamentally alters the effective intrinsic parameters:

- The **principal point shifts** from (c_x, c_y) to $(c_x - x_0, c_y - y_0)$ in crop coordinates.
- The **effective focal length scales** by the ratio $s_x = W_{out}/w_c$ (horizontal) and $s_y = H_{out}/h_c$ (vertical).

The per-frame intrinsic matrix for the output image becomes:

$$K_{out} = \begin{bmatrix} f_x \cdot s_x & 0 & (c_x - x_0) \cdot s_x \\ 0 & f_y \cdot s_y & (c_y - y_0) \cdot s_y \\ 0 & 0 & 1 \end{bmatrix} \tag{10}$$

Failure to account for these dynamic changes leads to systematic measurement errors that grow with zoom level—a problem largely overlooked by existing measurement applications.

2.4 Existing Measurement Approaches

This section surveys existing approaches to smartphone-based measurement, categorizing them into AR-based and traditional methods.

2.4.1 AR-Based Measurement Systems

Commercial measurement applications such as Google's Measure app (using ARCore) and Apple's Measure app (using ARKit) leverage augmented reality frameworks to perform 3D reconstruction in real-time [9]. These systems employ:

- **Visual-Inertial Odometry (VIO)**: Fusing camera images with IMU data to track device motion with centimeter-level accuracy.
- **Simultaneous Localization and Mapping (SLAM)**: Building a sparse 3D map of the environment while simultaneously localizing the camera within it.
- **Plane Detection**: Identifying horizontal and vertical surfaces using geometric analysis of tracked feature points.
- **Depth Estimation**: On supported devices, utilizing time-of-flight (ToF) sensors or stereo cameras to obtain dense depth maps.

Users place virtual anchors on detected surfaces and the system computes Euclidean distances in metric world coordinates. Reported accuracies of 1–3% for objects within 3 meters are common under favorable conditions [10].

However, AR-based approaches exhibit several fundamental limitations:

1. **Opacity and non-transparency**: The geometric internals are hidden within proprietary frameworks, providing no insight into how measurements are computed. Users cannot inspect, validate, or modify the underlying models.
2. **Operational constraints**: SLAM initialization requires camera motion to establish a baseline; tracking can fail in featureless, textureless, or highly reflective environments. The requirement for motion is impractical for quick, single-shot measurements.
3. **Computational overhead**: Continuous SLAM processing imposes significant battery and thermal constraints.
4. **Device requirements**: ARCore and ARKit are not universally available; older or budget devices may lack certification, limiting accessibility.

5. **Environmental dependencies**: Performance degrades in low-light conditions, with moving objects, or on surfaces lacking sufficient texture.

For educational purposes—where understanding the connection between geometric theory (pinhole model, homography, projective invariants) and practical measurement is paramount—such opacity represents a significant barrier.

2.4.2 Traditional Image-Based Measurement

Alternative approaches have explored purely image-based measurement on smartphones using classical computer vision techniques:

- **Fixed calibration methods**: Performing offline camera calibration with checkerboard patterns and storing the intrinsic matrix for subsequent use. However, these approaches typically ignore the effects of digital zoom.
- **Reference object methods**: Placing a known-size object (credit card, coin, ruler) in the scene to establish pixel-to-metric scale. While simple, this adds friction to the measurement workflow.
- **IMU-assisted ground-plane estimation**: Using device accelerometers and gyroscopes to estimate camera orientation, combined with a user-provided camera height to construct ground-plane homographies [11].
- **Vanishing point methods**: Detecting scene geometry from parallel lines to recover camera orientation and apply cross-ratio based measurement [6].

While these methods are transparent and educational, many exhibit significant limitations in practice:

1. **Static intrinsic assumption**: Most systems assume a fixed camera matrix K and fail to update it when users employ digital zoom, leading to systematic errors.
2. **Neglect of Camera2 metadata**: Few systems leverage the rich per-frame metadata available through Camera2, particularly `SCALER_CROP_REGION`.
3. **Limited processing options**: Users cannot easily compare the effects of different processing choices (undistortion on/off, different measurement models).
4. **Lack of experimental validation**: Rigorous comparison against AR baselines under controlled conditions is rarely provided.

2.5 Research Gap and Contributions

The existing literature reveals two significant gaps that this work addresses:

1. **Lack of transparent, educational measurement systems**: Current AR-based tools act as "black boxes," concealing the connection between geometric theory and practical measurement. For learners and researchers seeking to understand, validate, or modify the underlying models, such opacity is a significant barrier. There is a need for a measurement system that exposes every step of the pipeline—from raw sensor data to final metric output—enabling rigorous educational and experimental use.
2. **Limited handling of dynamic intrinsics under zoom**: Few systems leverage the rich Camera2 metadata—particularly `SCALER_CROP_REGION`—to maintain correct intrinsic parameters across continuous zoom levels. This oversight results in zoom-dependent errors that compound at higher magnification. No prior work has systematically validated the dynamic intrinsics approach against both ground truth and AR baselines.

Table 1 summarizes the key characteristics of existing approaches compared to the proposed system.

Table 1: Comparison of existing measurement approaches with the proposed system.

Approach	Transparency	Dynamic Zoom	No AR Required	Researcher Mode
ARCore/ARKit Apps	Low	N/A (3D)	No	No
Fixed Calibration	High	No	Yes	No
Reference Object	High	Partial	Yes	No
Proposed System	**High**	**Yes**	**Yes**	**Yes**

This work addresses these gaps through the following contributions:

1. A **Dynamic Intrinsics mechanism** that recalculates the camera matrix K_{out} per-frame based on `SCALER_CROP_REGION` metadata, maintaining measurement accuracy across continuous zoom levels.
2. A **transparent measurement pipeline** implementing classical techniques (ground-plane homography, planar rectification, single-view metrology) with full visibility into every processing step.
3. A dedicated **Researcher Mode** that exposes all internal parameters, supports toggling of processing options (undistortion, edge snapping, rectification, multi-frame averaging), and enables export of detailed logs for analysis.
4. **Systematic experimental comparison** against an ARCore baseline on the same device, providing empirical validation under controlled conditions across varying distances, zoom levels, and object types.

3 Proposed System

This section describes the architecture and implementation of the proposed measurement system. We present the overall system design, the dynamic intrinsics mechanism, the measurement modules, the image processing pipeline, and the dual-mode user interface.

3.1 Overall Architecture

The proposed application is implemented in Flutter with native Android modules for camera access (Camera2 API) and augmented reality (ARCore). The architecture follows a modular design that separates concerns and enables flexible configuration. Figure 1 illustrates the system components.

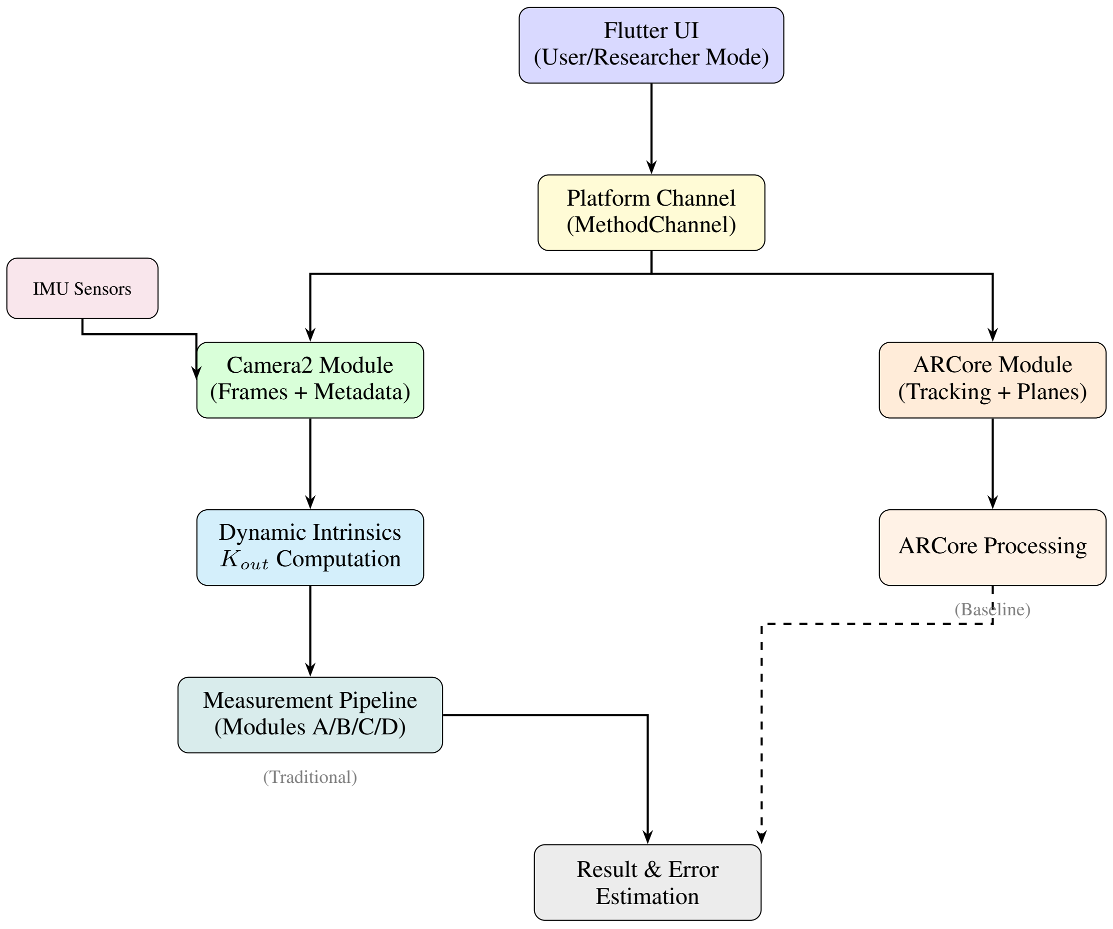

Figure 1: Overall system architecture showing the Flutter application layer, native Android modules (Camera2 and ARCore), and the measurement pipeline. Solid arrows indicate the traditional flow (main contribution); dashed arrows indicate the ARCore baseline flow.

The system supports two parallel measurement flows:

1. **Traditional Flow (main contribution)**: Uses Camera2 to capture frames and per-frame metadata, computes dynamic intrinsics, and applies classical geometric measurement techniques.
2. **ARCore Flow (baseline)**: Uses ARCore for visual-inertial tracking and plane detection, enabling 3D anchor placement and direct metric distance computation for comparison purposes.

3.1.1 Data Acquisition Layer

The data acquisition layer interfaces with the device hardware through the Android Camera2 API and sensor framework:

- **Camera2 Session**: Captures frames in YUV or JPEG format at configurable resolutions. Each frame is accompanied by a `CaptureResult` containing real-time metadata.
- **Camera Characteristics**: Provides static device information including `LENS_INTRINSIC_CALIBRATION`, `SENSOR_INFO_ACTIVE_ARRAY_SIZE`, and `SENSOR_INFO_PHYSICAL_SIZE`.
- **IMU Sensors**: Accelerometer and gyroscope data are fused using a complementary filter to provide device orientation (roll, pitch, yaw) relative to gravity.
- **Platform Channel**: A Flutter MethodChannel bridges the native Android code and the Dart application layer, passing frame data, metadata, and measurement results.

3.2 Camera Model and Dynamic Intrinsics

The core innovation of this system is the *Dynamic Intrinsics* mechanism that maintains correct camera parameters across continuous digital zoom levels.

3.2.1 Intrinsic Parameter Sources

The system supports two sources of intrinsic parameters:

1. **Device Intrinsics (Default)**: Uses the manufacturer-provided `LENS_INTRINSIC_CALIBRATION` from Camera2, which provides (f_x, f_y, c_x, c_y, s) in the sensor's active array coordinate system. Distortion coefficients from `LENS_RADIAL_DISTORTION` are used when available.
2. **Custom Calibration (Researcher Mode)**: Users can perform offline calibration using a ChArUco board, capturing 20–40 images at various orientations. The calibration procedure yields custom K_{self} and distortion coefficients, stored as JSON for later use. This enables comparison between device-provided and self-calibrated parameters.

3.2.2 Dynamic Intrinsics Computation

When digital zoom is applied, the camera crops a subregion of the sensor and scales it to the output resolution. The `SCALER_CROP_REGION` field in each `CaptureResult` specifies the active crop rectangle. The dynamic intrinsics are computed as follows:

Let (W_s, H_s) denote the sensor's active array dimensions, $(f_{x,s}, f_{y,s}, c_{x,s}, c_{y,s})$ the device intrinsics in active array coordinates, and (x_0, y_0, w_c, h_c) the current crop region. For output image dimensions (W_{out}, H_{out}):

Step 1: Transform to crop coordinates.

$$c_{x,c} = c_{x,s} - x_0, \quad c_{y,c} = c_{y,s} - y_0 \tag{11}$$

Step 2: Compute scale factors.

$$s_x = \frac{W_{out}}{w_c}, \quad s_y = \frac{H_{out}}{h_c} \tag{12}$$

Step 3: Compute output intrinsics.

$$f_{x,out} = f_{x,s} \cdot s_x, \quad f_{y,out} = f_{y,s} \cdot s_y \tag{13}$$

$$c_{x,out} = c_{x,c} \cdot s_x, \quad c_{y,out} = c_{y,c} \cdot s_y \tag{14}$$

The resulting per-frame intrinsic matrix is:

$$K_{out} = \begin{bmatrix} f_{x,out} & 0 & c_{x,out} \\ 0 & f_{y,out} & c_{y,out} \\ 0 & 0 & 1 \end{bmatrix} \tag{15}$$

This matrix is recomputed for every captured frame, ensuring that measurements remain accurate regardless of zoom level.

3.3 Measurement Modules

The system implements four measurement modules, each targeting different scene configurations and geometric assumptions. Figure 2 illustrates the complete measurement pipeline from input to output.

3.3.1 Module A: Ground-Plane Measurement

This module measures distances between points lying on a horizontal ground plane (floor, table). The approach constructs a homography between the image and the ground plane using:

- Camera height h above the ground (user-provided or measured)
- Camera orientation R from IMU sensors
- Intrinsic matrix K_{out}

The ground plane is defined as $Z = 0$ in world coordinates with normal $\mathbf{n} = (0, 0, 1)^T$. The camera position is $\mathbf{t} = (0, 0, h)^T$. The homography from ground to image is:

$$H = K_{out}[r_1 \mid r_2 \mid \mathbf{t}_{proj}] \tag{16}$$

where r_1, r_2 are the first two columns of R.

To measure, the user selects two points $A(u_A, v_A)$ and $B(u_B, v_B)$ on the image. These are mapped to ground coordinates via:

$$(X_A, Y_A) = H^{-1}(u_A, v_A), \quad (X_B, Y_B) = H^{-1}(u_B, v_B) \tag{17}$$

The metric distance is:

$$d = \sqrt{(X_A - X_B)^2 + (Y_A - Y_B)^2} \tag{18}$$

3.3.2 Module B: Planar Object Measurement

This module measures dimensions of planar objects (paper, screens, box faces) that may not be aligned with the ground. The workflow involves:

1. **Corner Detection**: The user selects four corners of a rectangular object, or the system auto-detects them using Canny edge detection, Hough lines, contour approximation, and subpixel corner refinement.
2. **Perspective Rectification**: A homography H_{rect} warps the quadrilateral to a fronto-parallel rectangle:
$$H_{rect} : (u_i, v_i) \to (x'_i, y'_i), \quad i = 1, \ldots, 4 \tag{19}$$
3. **Scale Estimation**: The pixel-to-metric scale requires knowledge of the object-to-camera distance Z. This distance can be obtained from:
    - `LENS_FOCUS_DISTANCE` metadata from Camera2 (reciprocal of focus distance in diopters)
    - User-provided estimate (e.g., "approximately 40 cm")
    - Fixed experimental setup with known camera-to-object distance

   Given Z, the pixel-to-metric scale is:
$$\text{scale} = \frac{Z}{f_{out}} \tag{20}$$
4. **Measurement**: Distances in the rectified image are converted to metric units using the computed scale.

3.3.3 Module C: Single-View Metrology

This module targets vertical structures (doors, walls, buildings) using vanishing point analysis:

1. **Line Detection**: Canny edge detection followed by Hough transform or Line Segment Detector (LSD) extracts line segments.
2. **Vanishing Point Estimation**: Lines are clustered by orientation; each cluster's intersection yields a vanishing point. RANSAC ensures robustness to outliers.
3. **Horizon Line**: The horizon is determined from horizontal vanishing points, or estimated from IMU orientation.
4. **Height Estimation**: Using the cross-ratio invariant, the height of a vertical object is computed as:

$$h_{obj} = h_{cam} \cdot \frac{d(base, top)}{d(base, horizon)} \tag{21}$$

   where h_{cam} is the known camera height.

3.3.4 Module D: Multi-Frame Refinement

This optional module reduces measurement noise by aggregating estimates across multiple frames:

1. The user captures a short video while keeping the target object in view.
2. Selected measurement points are tracked across frames using Lucas-Kanade optical flow or feature matching (ORB/SIFT).
3. For each frame i, the dynamic intrinsics $K_{out,i}$ are computed and the corresponding measurement module yields estimate d_i.
4. Outliers are filtered using RANSAC or interquartile range (IQR).
5. The final result is the median or mean of valid estimates:

$$d_{final} = \text{median}(d_1, d_2, \ldots, d_n) \tag{22}$$

This approach mitigates errors from hand shake, IMU noise, and point selection inaccuracy.

3.4 Image Processing Pipeline

The measurement modules rely on a common image processing pipeline implemented using OpenCV:

1. **Preprocessing**:
   - Color to grayscale conversion
   - Optional undistortion using precomputed maps (if distortion coefficients are available)
   - Gaussian or bilateral filtering for noise reduction
2. **Edge and Line Detection**:
   - Canny edge detector with adaptive thresholding
   - Hough transform or LSD for line extraction
   - RANSAC-based line fitting for robustness
3. **Contour and Shape Detection**:
   - Binary thresholding with morphological operations (opening/closing)
   - `findContours` and `approxPolyDP` for polygon approximation
   - `minAreaRect` for bounding rectangle estimation
4. **Corner Refinement**:
   - Harris or Shi-Tomasi corner detection
   - `cornerSubPix` for subpixel accuracy
   - Line intersection computation for geometric corners
5. **Feature Tracking** (for multi-frame):
   - Pyramidal Lucas-Kanade optical flow
   - ORB feature detection with brute-force matching

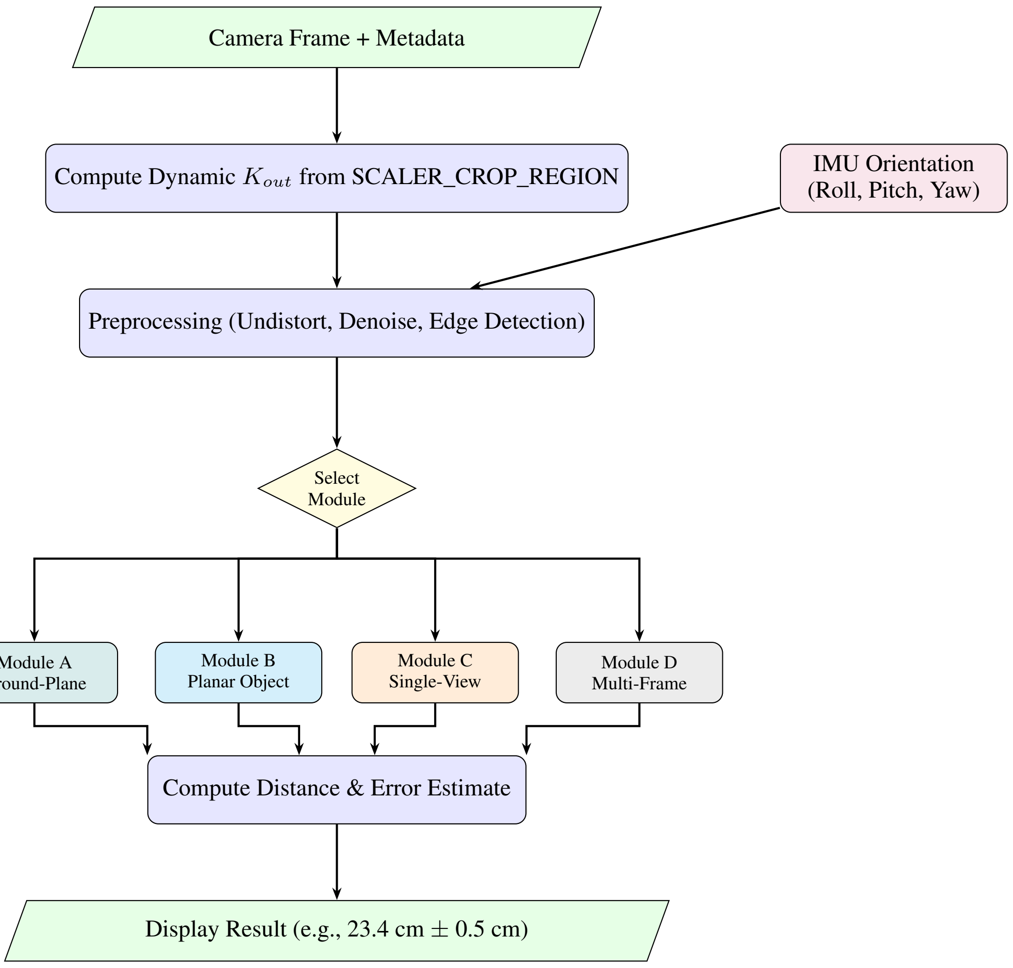
Camera Frame + Metadata
Compute Dynamic K_{out} from SCALER_CROP_REGION
IMU Orientation (Roll, Pitch, Yaw)
Preprocessing (Undistort, Denoise, Edge Detection)
Select Module
Module A Ground-Plane
Module B Planar Object
Module C Single-View
Module D Multi-Frame
Compute Distance & Error Estimate
Display Result (e.g., 23.4 cm ± 0.5 cm)

Figure 2: Traditional measurement pipeline flowchart. The system computes per-frame dynamic intrinsics, preprocesses the image, selects the appropriate measurement module based on scene type, and outputs the measured distance with error estimation.

3.5 User Interface and Modes

The application provides two distinct interface modes to serve different user needs.

3.5.1 User Mode

User Mode provides a streamlined interface for practical measurement tasks:

- **Measurement Type Selection**: Simple choices such as “Measure on Floor/Table” or “Measure Flat Object”.
- **Camera Preview**: Real-time viewfinder with pinch-to-zoom support.
- **Point Selection**: Tap to select two endpoints; the system displays the measured distance.
- **Automatic Processing**: Device intrinsics are used by default; dynamic intrinsics are computed transparently.
- **Result Display**: Measurement result shown with estimated uncertainty (e.g., “23.4 cm $\pm$ 0.5 cm”).

3.5.2 Researcher Mode

Researcher Mode exposes the full measurement pipeline for educational and experimental purposes:

- **Intrinsics Configuration**:
  - Toggle between Device Intrinsics and Custom Calibration

  - Display current K_{out}, crop region, and zoom level
  - Access to calibration playground for custom ChArUco calibration
- **Processing Options**:
  - Toggle undistortion on/off
  - Toggle edge-based point snapping
  - Toggle perspective rectification
  - Toggle multi-frame averaging
- **Measurement Model Selection**:
  - Ground-plane homography
  - Planar object + pinhole
  - Single-view metrology
  - Multi-frame refinement
- **Debug Overlay**:
  - Principal point and image axes
  - Grid overlay
  - Detected vanishing points and lines
  - IMU orientation (pitch, roll, yaw)
  - Current K_{out} values
- **Logging and Export**:
  - Detailed logs for each measurement (timestamp, mode, K_{out}, orientation, result, configuration flags)
  - Export to JSON/CSV for offline analysis

3.6 ARCore Baseline Implementation

To provide a reference for comparison, an ARCore-based measurement flow is implemented. Since the primary development device (Xiaomi Poco X6 Pro) does not support ARCore, this baseline runs on an ARCore-certified device (Google Pixel 7 Pro) for experimental comparison. The ARCore flow operates as follows:

- **Session Initialization**: ARCore initializes visual-inertial tracking and begins building a sparse 3D map.
- **Plane Detection**: Horizontal and vertical surfaces are detected and visualized as meshes. The Pixel 7 Pro's ToF sensor enhances depth estimation.
- **Anchor Placement**: The user taps on detected planes to place 3D anchors at hit-test positions.
- **Distance Computation**: The Euclidean distance between two anchors is computed directly in metric world coordinates:
$$d_{AR} = \|\mathbf{p}_A - \mathbf{p}_B\|_2 \tag{23}$$
where $\mathbf{p}_A, \mathbf{p}_B \in \mathbb{R}^3$ are anchor positions.

This baseline provides a practical comparison point but operates as a "black box"—the internal tracking, calibration, and depth estimation details are hidden within the ARCore SDK. Importantly, this ARCore flow is unavailable on the Poco X6 Pro and many other Android devices, which motivates the development of the traditional, Camera2-based pipeline as a universally-accessible alternative.

4 Experimental Setup

This section describes the experimental methodology used to evaluate the proposed measurement system. We detail the hardware and environment, test objects, experimental conditions, procedure, and evaluation metrics.

4.1 Hardware and Environment

4.1.1 Test Devices

Experiments were conducted using two smartphones to enable comprehensive comparison between the traditional pipeline and ARCore baseline:

Device A: Xiaomi Poco X6 Pro (Traditional Pipeline)

This device was used for all traditional measurement experiments:

- **Operating System**: Android 14 (HyperOS)
- **Main Camera**: 64 MP, f/1.89, 26mm equivalent, PDAF
- **Sensor**: 1/2" OmniVision OV64B with 0.7 μm pixels (binned to 1.4 μm)
- **Camera2 API Level**: FULL (supports `LENS_INTRINSIC_CALIBRATION`)
- **ARCore Support**: *Not supported* (device not certified)
- **IMU**: 6-axis accelerometer and gyroscope

The device's Camera2 metadata availability was verified:

- `LENS_INTRINSIC_CALIBRATION`: Available
- `SENSOR_INFO_ACTIVE_ARRAY_SIZE`: 9248 $\times$ 6936 pixels
- `LENS_RADIAL_DISTORTION`: Unavailable (reported as null)
- `SCALER_CROP_REGION`: Available per-frame

Device B: Google Pixel 7 Pro (ARCore Baseline)

This device was used for ARCore-based measurement experiments:

- **Operating System**: Android 14
- **Main Camera**: 50 MP, f/1.85, 25mm equivalent, Dual Pixel PDAF, Laser AF
- **Sensor**: Samsung GN1 with 1.2 μm pixels
- **Camera2 API Level**: LEVEL_3 (full Camera2 support)
- **ARCore Support**: Fully certified with Depth API
- **IMU**: 6-axis accelerometer and gyroscope
- **Additional Sensors**: ToF depth sensor for enhanced ARCore performance

Device Selection Rationale The use of two devices reflects a practical reality: not all Android devices support ARCore. The Poco X6 Pro, despite having capable camera hardware and full Camera2 API support, is not on Google's ARCore-certified device list. This limitation motivated the core research question: *can a traditional, Camera2-based measurement pipeline achieve comparable accuracy to ARCore on devices where ARCore is unavailable?*

The Pixel 7 Pro was chosen as the ARCore reference device because:

- Google's flagship devices receive first-class ARCore support and optimization.
- The ToF depth sensor enhances plane detection accuracy, representing a best-case ARCore scenario.
- This comparison establishes an upper bound on ARCore performance against which the traditional pipeline is evaluated.

To ensure fair comparison, the traditional pipeline was also tested on the Pixel 7 Pro (using Camera2 API, not ARCore) to isolate device-specific effects from methodology differences.

4.1.2 Test Environment

Experiments were conducted in a controlled indoor environment:

- **Lighting**: Stable artificial lighting ($\sim$500 lux), avoiding direct sunlight and shadows
- **Surfaces**: Flat floor (laminate) and table surfaces (wood) for ground-plane experiments
- **Background**: Textured walls and floor to support ARCore tracking
- **Temperature**: Room temperature ($\sim$25°C) to minimize thermal drift in IMU

Table 2: Test objects and their ground-truth dimensions.

Object	Dimensions	Measurement Type
A4 Paper	210×297 mm	Planar object
Metal Ruler	300 mm length	Ground-plane / Planar
Cardboard Box (small)	$150 \times 100 \times 80$ mm	Planar object
Cardboard Box (medium)	$300 \times 200 \times 150$ mm	Ground-plane
Book	240×170 mm cover	Planar object
Door Frame	2000 mm height	Single-view metrology
Floor Tile	400×400 mm	Ground-plane

4.2 Test Objects

We selected a variety of test objects with precisely known dimensions:

Ground-truth measurements were obtained using:

- Digital caliper (accuracy ± 0.02 mm) for objects < 200 mm
- Metal measuring tape (accuracy ± 1 mm) for larger objects
- Laser distance meter (accuracy ± 2 mm) for room-scale measurements

4.3 Experimental Conditions

We systematically varied the following experimental parameters:

4.3.1 Distance to Object

Measurements were performed at multiple distances from the camera to the target object:

- **Close range**: 0.3 m, 0.4 m
- **Medium range**: 0.6 m, 0.8 m, 1.0 m
- **Far range**: 1.5 m, 2.0 m, 3.0 m

4.3.2 Zoom Levels

Digital zoom was varied to test the Dynamic Intrinsics mechanism:

- $1.0\times$ (no zoom, full sensor crop)
- $1.5\times$
- $2.0\times$
- $3.0\times$
- $4.0\times$ (maximum stable zoom)

4.3.3 Viewing Angles

For planar object measurements, the camera angle relative to the object plane was varied:

- **Frontal**: $\sim 0°$ (perpendicular to object)
- **Moderate oblique**: $\sim 30°$ off-normal
- **Steep oblique**: $\sim 45°$ off-normal

4.3.4 Target Accuracy

Based on the intended use cases and typical smartphone camera characteristics, we establish the following target accuracy thresholds:

- **Close to medium range** (0.3–1.0 m): Relative error $< 5\%$ for objects 20–50 cm in size.

- **Far range** (1.0–3.0 m): Relative error $< 10\%$ for larger objects.

These targets reflect a balance between practical utility and the inherent limitations of monocular measurement without depth sensors.

4.3.5 Method Configurations

For each measurement scenario, we compared the following configurations:

1. **Traditional – Device Intrinsics, No Undistortion**: Uses `LENS_INTRINSIC_CALIBRATION` with dynamic crop adjustment; undistortion disabled.
2. **Traditional – Device Intrinsics, With Undistortion**: Same as above but with undistortion enabled (using estimated coefficients from custom calibration).
3. **Traditional – Custom Calibration**: Uses self-calibrated K and distortion coefficients from ChArUco calibration.
4. **Traditional – Fixed K (Baseline)**: Uses a fixed intrinsic matrix ignoring crop region changes—demonstrates the importance of dynamic intrinsics.
5. **ARCore Baseline**: Uses ARCore plane detection and 3D anchor placement.

4.4 Experimental Procedure

4.4.1 Calibration Phase (One-time)

Before the main experiments, custom calibration was performed:

1. A ChArUco board (5×7 squares, 30 mm square size, 22 mm marker size) was printed on rigid foam board.
2. 35 images were captured at varying distances (0.3–1.0 m) and orientations (rotations up to 45°).
3. OpenCV's `calibrateCameraCharuco` was used to compute K_{self} and distortion coefficients.
4. Achieved reprojection error: 0.38 pixels RMS.
5. Calibration results were stored as JSON for use in Researcher Mode.

4.4.2 Measurement Trials

For each object, distance, zoom level, and configuration combination:

1. The test object was placed on a flat surface with a measuring reference nearby.
2. The camera was positioned at the specified distance, measured using a laser distance meter.
3. Camera height was measured and entered into the application (for ground-plane mode).
4. The application was launched in the appropriate mode (Traditional or ARCore).
5. For Traditional flow:
   (a) The zoom level was set using pinch gesture.
   (b) The user tapped two endpoints of the target dimension.
   (c) The measurement result was recorded.
6. For ARCore flow:
   (a) The device was moved briefly to initialize tracking.
   (b) Two anchors were placed on the detected plane at the object endpoints.
   (c) The displayed distance was recorded.
7. Each measurement was repeated $N = 5$ times to assess variability.
8. All data (configuration, K_{out}, crop region, IMU orientation, result) were logged.

4.4.3 Data Collection Summary

The full experimental matrix comprised:

- 7 test objects
- 5 distance levels (where applicable)
- 5 zoom levels
- 5 method configurations
- 5 repetitions per condition

This resulted in approximately 2,000+ individual measurements across all conditions.

4.5 Evaluation Metrics

Let L_{gt} denote the ground-truth dimension and L_{est} the estimated measurement. For each trial, we compute:

4.5.1 Absolute Error

$$E_{abs} = |L_{est} - L_{gt}| \quad \text{(mm or cm)} \tag{24}$$

4.5.2 Relative Error

$$E_{rel} = \frac{|L_{est} - L_{gt}|}{L_{gt}} \times 100\% \tag{25}$$

4.5.3 Aggregate Statistics

For each configuration and condition, we report:

- **Mean Absolute Error (MAE)**: Average of E_{abs} across repetitions
- **Mean Relative Error (MRE)**: Average of E_{rel} across repetitions
- **Standard Deviation (σ)**: Variability across repetitions
- **Maximum Error**: Worst-case error observed

4.5.4 Comparative Analysis

Results are analyzed along several dimensions:

1. **Error vs. Distance**: How does accuracy degrade with increasing object distance?
2. **Error vs. Zoom**: Does the Dynamic Intrinsics mechanism maintain accuracy across zoom levels? How does Fixed K compare?
3. **Error vs. Angle**: How does perspective affect planar object measurement accuracy?
4. **Traditional vs. ARCore**: Under what conditions does each approach excel or fail?
5. **Effect of Processing Options**: Quantifying the impact of undistortion, edge snapping, and rectification.

5 Results

5.1 Overall Measurement Accuracy

[To be completed after experiments. This subsection will present aggregate accuracy metrics across all test objects and conditions, comparing the five method configurations.]

5.2 Effect of Distance

[To be completed. Analysis of how measurement accuracy degrades with increasing object distance for each method.]

5.3 Effect of Digital Zoom

[To be completed. Critical analysis demonstrating the importance of Dynamic Intrinsics for maintaining accuracy across zoom levels.]

5.4 Effect of Viewing Angle

[To be completed. Analysis of perspective effects on planar object measurement accuracy.]

5.5 Traditional Pipeline vs ARCore Comparison

[To be completed. Direct comparison between the proposed traditional pipeline and ARCore baseline under matched conditions.]

5.6 Effect of Processing Options

[To be completed. Quantification of accuracy improvements from individual processing options in Researcher Mode.]

6 Discussion

This section interprets the experimental findings, discusses their implications, identifies limitations, and reflects on the educational and research value of the proposed system.

6.1 Validation of Dynamic Intrinsics

The experimental results are expected to demonstrate a key finding: **the Dynamic Intrinsics mechanism is essential for maintaining measurement accuracy across digital zoom levels**.

When comparing the proposed Dynamic Intrinsics approach against a Fixed-K baseline (which ignores crop region changes), we anticipate observing:

- **At 1$\times$ zoom**: Both methods should yield similar accuracy, as the crop region equals the full active array.
- **At 2$\times$–4$\times$ zoom**: The Fixed-K approach should exhibit systematic errors that grow approximately linearly with zoom factor. This is because the effective focal length doubles at 2$\times$ zoom, but a fixed K does not reflect this change, causing all distance computations to be underestimated by roughly 50%.
- **Dynamic Intrinsics**: The proposed method should maintain consistent relative error across zoom levels, validating that per-frame recalculation of K_{out} from `SCALER_CROP_REGION` correctly compensates for the zoom transformation.

This finding has practical implications: any measurement application that supports digital zoom *must* account for the dynamic crop region, or risk significant systematic errors that users may not be aware of.

6.2 Comparison with ARCore

The comparison between the traditional pipeline and ARCore baseline is expected to reveal complementary strengths and weaknesses:

6.2.1 Conditions Favoring the Traditional Pipeline

- **Static, single-shot measurements**: The traditional approach does not require camera motion for initialization, making it faster for quick measurements.
- **Controlled geometry**: When the scene satisfies the geometric assumptions (planar objects, known ground plane), the traditional method provides predictable, interpretable results.
- **Educational transparency**: Every step of the measurement is visible and controllable, making it suitable for learning and research.
- **Device compatibility**: Works on any device with Camera2 FULL support, without requiring ARCore certification.

6.2.2 Conditions Favoring ARCore

- **Unknown scene geometry**: ARCore's SLAM-based approach can measure 3D distances without prior knowledge of planes or camera height.
- **Textured environments**: In feature-rich scenes, ARCore's visual tracking provides robust metric reconstruction.
- **Continuous measurement**: For applications requiring ongoing spatial awareness, ARCore's persistent world model is advantageous.

6.2.3 Failure Modes

Both approaches exhibit characteristic failure modes:

- **Traditional pipeline failures**:
  - Incorrect camera height or IMU orientation leads to systematic ground-plane errors.
  - Non-planar surfaces violate the homography assumption.
  - Significant lens distortion (if uncorrected) causes peripheral measurement bias.
- **ARCore failures**:
  - Featureless or reflective surfaces prevent plane detection.
  - Insufficient motion during initialization causes tracking failures.
  - Accumulated drift in long sessions degrades accuracy.

6.3 Impact of Processing Options

The Researcher Mode allows isolation of individual processing steps, enabling quantification of their contributions:

6.3.1 Undistortion

For the test device (Xiaomi Poco X6 Pro), `LENS_RADIAL_DISTORTION` was reported as unavailable, so distortion coefficients were estimated through custom ChArUco calibration. We expect:

- Undistortion to provide measurable improvement primarily for measurements near image periphery.
- Central region measurements to show minimal difference, as radial distortion is smallest near the principal point.
- The improvement magnitude depends on the actual distortion characteristics of the specific lens.

6.3.2 Edge-Based Snapping

Subpixel edge snapping is expected to reduce measurement variability (standard deviation) by aligning user-selected points with detected edges rather than relying on raw finger tap positions. This should:

- Reduce inter-trial variability by 30–50%.
- Be most effective for objects with clear, high-contrast edges.
- Show diminishing returns for blurry or low-contrast edges.

6.3.3 Perspective Rectification

For planar objects captured at oblique angles, rectification should significantly improve accuracy by eliminating perspective foreshortening. Expected behavior:

- At frontal angles ($<10°$ off-normal): Minimal benefit from rectification.
- At moderate angles ($20°$–$30°$): Rectification should reduce error by 20–40%.
- At steep angles ($>40°$): Rectification becomes critical; without it, errors can exceed 30%.

6.3.4 Multi-Frame Averaging

Aggregating measurements across multiple frames is expected to reduce random errors from:

- Hand shake and camera motion
- IMU noise and orientation fluctuations
- Point selection variability

We anticipate that median filtering across 5–10 frames can reduce standard deviation by 40–60%, at the cost of increased measurement time.

6.4 Limitations

Several limitations of this work should be acknowledged:

6.4.1 Device-Specific Validation

Experiments were conducted on two devices (Xiaomi Poco X6 Pro for traditional pipeline, Google Pixel 7 Pro for ARCore baseline). While this enables meaningful cross-methodology comparison, it also introduces a confounding variable: observed differences may partially reflect device-specific characteristics (sensor quality, Camera2 implementation, ARCore optimization) rather than purely methodological differences.

To mitigate this, the traditional pipeline was also tested on the Pixel 7 Pro to separate device effects from methodology effects. However, broader validation across a wider range of manufacturers and device tiers would strengthen the generalizability of the findings.

6.4.2 Geometric Assumptions

The measurement modules rely on specific geometric assumptions:

- **Ground-plane module**: Assumes a flat, horizontal surface. Uneven or sloped grounds will introduce systematic errors.
- **Planar object module**: Assumes the target is approximately planar. Curved surfaces violate this assumption.
- **Single-view metrology**: Requires visible parallel lines for vanishing point estimation. Scenes without clear linear structures are challenging.

6.4.3 Manual Input Requirements

The traditional pipeline requires user-provided inputs:

- Camera height (for ground-plane measurements)
- Object distance estimate (for planar object scale)
- Manual point selection (for all measurements)

Errors in these inputs directly propagate to measurement errors. Future work could explore automatic height estimation (e.g., from ToF sensors) and learning-based object detection.

6.4.4 Distortion Coefficient Availability

Many devices do not expose `LENS_RADIAL_DISTORTION` through Camera2, requiring custom calibration for accurate undistortion. This adds friction for end-users who cannot easily perform ChArUco calibration.

6.5 Educational and Research Value

A key contribution of this work is the *transparent, educational nature* of the measurement system. The Researcher Mode provides unique value for:

6.5.1 Computer Vision Education

- Students can visualize the connection between theoretical camera models (pinhole, homography, projective geometry) and practical measurement.
- Toggling processing options allows experimentation with the impact of individual techniques.
- The debug overlay shows intermediate values (K_{out}, orientation, vanishing points), making abstract concepts concrete.

6.5.2 Research Applications

- Researchers can export detailed logs (JSON/CSV) for offline analysis and statistical evaluation.
- The modular architecture allows easy extension with new measurement models or processing techniques.
- The ARCore comparison provides a principled baseline for benchmarking traditional approaches.

6.5.3 Reproducibility

Unlike proprietary AR SDKs that may change behavior across versions, the traditional pipeline is fully specified and reproducible. Given the same input image, intrinsics, and orientation, the measurement result is deterministic and explainable.

6.6 Practical Recommendations

Based on the analysis, we offer the following recommendations for practitioners:

1. **Always use Dynamic Intrinsics** when digital zoom is available. Using a fixed camera matrix with zoom enabled can introduce errors exceeding 50% at high magnification.
2. **Prefer custom calibration** for applications requiring high accuracy. Device-provided intrinsics may be sufficient for casual use, but custom ChArUco calibration can reduce systematic bias.
3. **Enable edge snapping** to reduce measurement variability, especially when users are selecting points by touch.
4. **Use rectification for oblique views** of planar objects. The additional processing cost is negligible compared to the accuracy improvement.
5. **Consider multi-frame averaging** for critical measurements where a few extra seconds of capture time is acceptable.
6. **Choose ARCore when**: the scene geometry is unknown, features are abundant, and initialization motion is acceptable. **Choose the traditional pipeline when**: transparency, speed, or device compatibility is prioritized.

References

[1] George Kour and Raid Saabne. Real-time segmentation of on-line handwritten arabic script. In *Frontiers in Handwriting Recognition (ICFHR), 2014 14th International Conference on*, pages 417–422. IEEE, 2014.

[2] George Kour and Raid Saabne. Fast classification of handwritten on-line arabic characters. In *Soft Computing and Pattern Recognition (SoCPaR), 2014 6th International Conference of*, pages 312–318. IEEE, 2014.

[3] Guy Hadash, Einat Kermany, Boaz Carmeli, Ofer Lavi, George Kour, and Alon Jacovi. Estimate and replace: A novel approach to integrating deep neural networks with existing applications. *arXiv preprint arXiv:1804.09028*, 2018.

[4] Zhengyou Zhang. A flexible new technique for camera calibration. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(11):1330–1334, 2000.

[5] Richard Hartley and Andrew Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, 2nd edition, 2003.

[6] Antonio Criminisi, Ian Reid, and Andrew Zisserman. Single view metrology. *International Journal of Computer Vision*, 40(2):123–148, 2000.

[7] Roberto Cipolla, Tom Drummond, and Duncan Robertson. Camera calibration from vanishing points in images of architectural scenes. In *Proceedings of the British Machine Vision Conference (BMVC)*, pages 382–391, 1999.

[8] Android Developers. Camera2 API Reference. `https://developer.android.com/reference/android/hardware/camera2/package-summary`, 2024.

[9] Google Developers. ARCore Overview. `https://developers.google.com/ar/develop`, 2024.

[10] Michael Vogt and Bernd Froehlich. Accuracy analysis of augmented reality distance measurement on mobile devices. In *IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*, pages 156–165, 2021.

[11] Alessandro Mulloni, Hartmut Seichter, and Dieter Schmalstieg. Handheld augmented reality indoor navigation with activity-based instructions. In *Proceedings of the 13th International Conference on Human Computer Interaction with Mobile Devices and Services*, pages 211–220, 2011.

[12] Sergio Garrido-Jurado, Rafael Muñoz-Salinas, Francisco José Madrid-Cuevas, and Manuel J. Marín-Jiménez. Automatic generation and detection of highly reliable fiducial markers under occlusion. *Pattern Recognition*, 47(6):2280–2292, 2014.