**✅ Recommendation: Build a Simplified Tracking + Mapping System using ORB Features**

**Why this direction?**

* It avoids complex stuff like loop closure, global relocalization, and essential graph optimization.
* You still cover meaningful and visible output (camera pose + point cloud).
* Great for a scientific report: you can benchmark accuracy (e.g., trajectory error) and show 3D visualization.

**🧠 Breakdown of Your Mini ORB-SLAM (what to implement):**

**1. ORB Feature Extraction**

* Use OpenCV’s cv2.ORB\_create() to extract features from frames.

**2. Frame-to-Frame Tracking**

* Match ORB descriptors between consecutive frames using brute-force Hamming matcher.
* Estimate relative pose using cv2.findEssentialMat() + cv2.recoverPose().

**3. Simple Mapping**

* Triangulate new 3D points using matching keypoints and camera matrices.
* Store sparse map of 3D landmarks.

**4. Keyframe Selection (Basic Rule)**

* Insert a new keyframe if there's enough motion from the last.

**🧪 What you can demonstrate for the prototype (week 7):**

* Run it on a short sequence from **TUM RGB-D fr1/xyz** (already low complexity).
* Show estimated trajectory + point cloud.
* Output metrics: trajectory RMSE (compare to ground truth using evo or your own script).

**📄 What to include in your report:**

**Abstract (Example based on your prompt):**

We present a simplified monocular SLAM pipeline inspired by ORB-SLAM, implementing tracking and sparse mapping using ORB features. Unlike the full system, our focus is on core modules — feature extraction, camera pose estimation, and 3D point triangulation — excluding loop closure and relocalization. We evaluate our system on the TUM RGB-D dataset, achieving 85% of the baseline accuracy with a significantly smaller codebase and real-time performance.

**Sections:**

1. **Introduction**
   * The goal: localize a monocular camera and reconstruct the environment.
2. **Related Work**
   * PTAM, ORB-SLAM, and direct vs. feature-based SLAM.
3. **Method**
   * ORB extraction
   * Frame-to-frame pose estimation
   * Triangulation
   * Basic keyframe policy
4. **Experiments**
   * Dataset: TUM fr1/xyz
   * Metrics: ATE (Absolute Trajectory Error), 3D map visualization
5. **Result**
   * Pose estimation quality vs. ground truth
   * Runtime performance
6. **Conclusion**
   * What worked, limitations (no loop closure), future work.