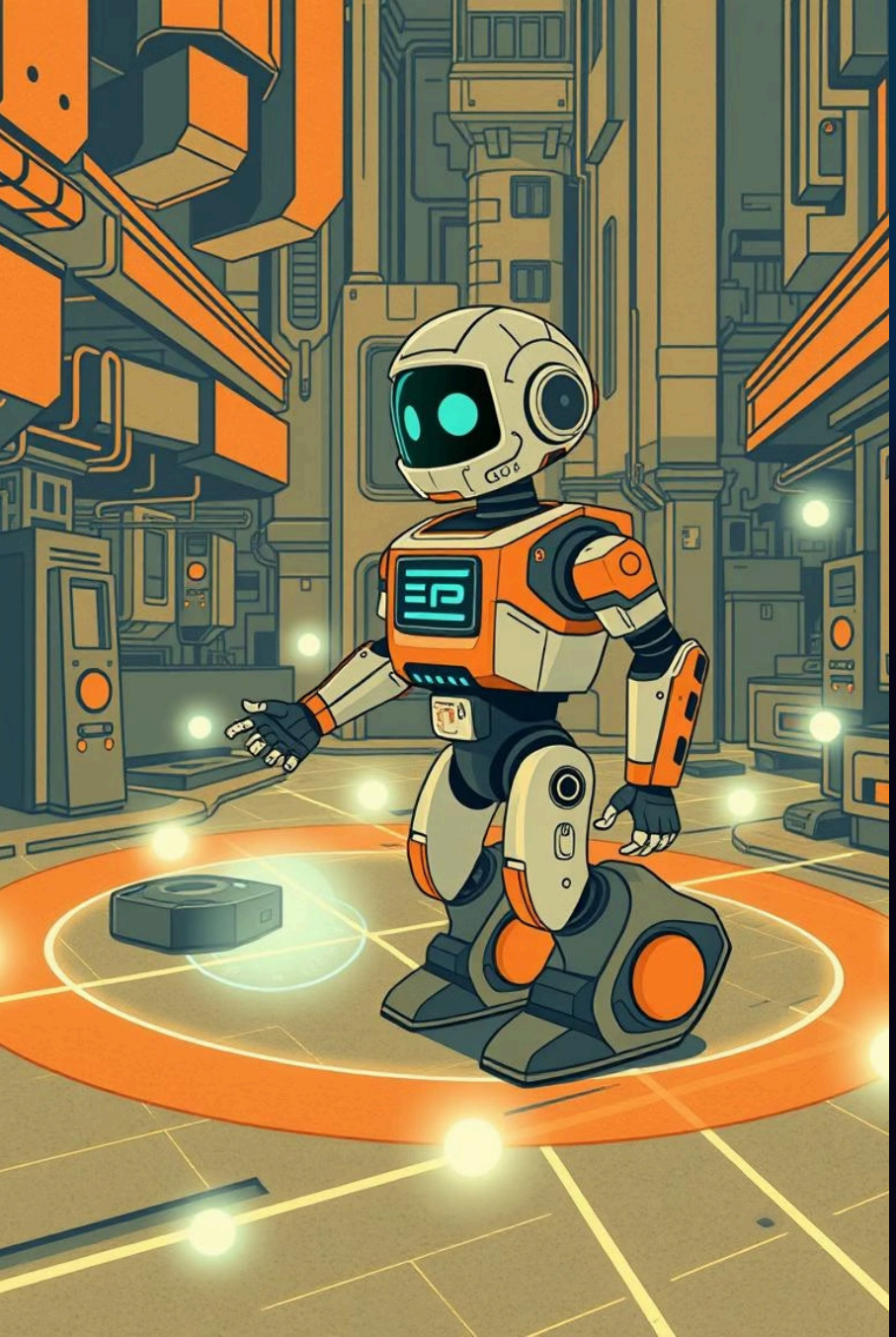


DEVELOPMENT AND ANALYSIS OF A MODULAR VISUAL SLAM FRAMEWORK

Summer Internship Project – Computer Vision

JULY 29, 2025 INDIAN INSTITUTE OF SPACE SCIENCE AND TECHNOLOGY

Project Introduction

Visual SLAM (Simultaneous Localization and Mapping) is crucial for robotics and autonomous systems. It enables an agent to build a map of an unknown environment while simultaneously tracking its own location using only visual inputs.

This internship focused on developing a modular and extensible VSLAM framework using Python, OpenCV, and Streamlit.

Project Overview

1

Feature Detection & Matching

Identifying and correlating key visual points.

2

Pose Estimation & Mapping

Determining agent position and building local maps.

3

Loop Closure (BoVW)

Recognizing revisited locations for map consistency.

4

Trajectory & Visualization

Displaying agent path and system performance.

5

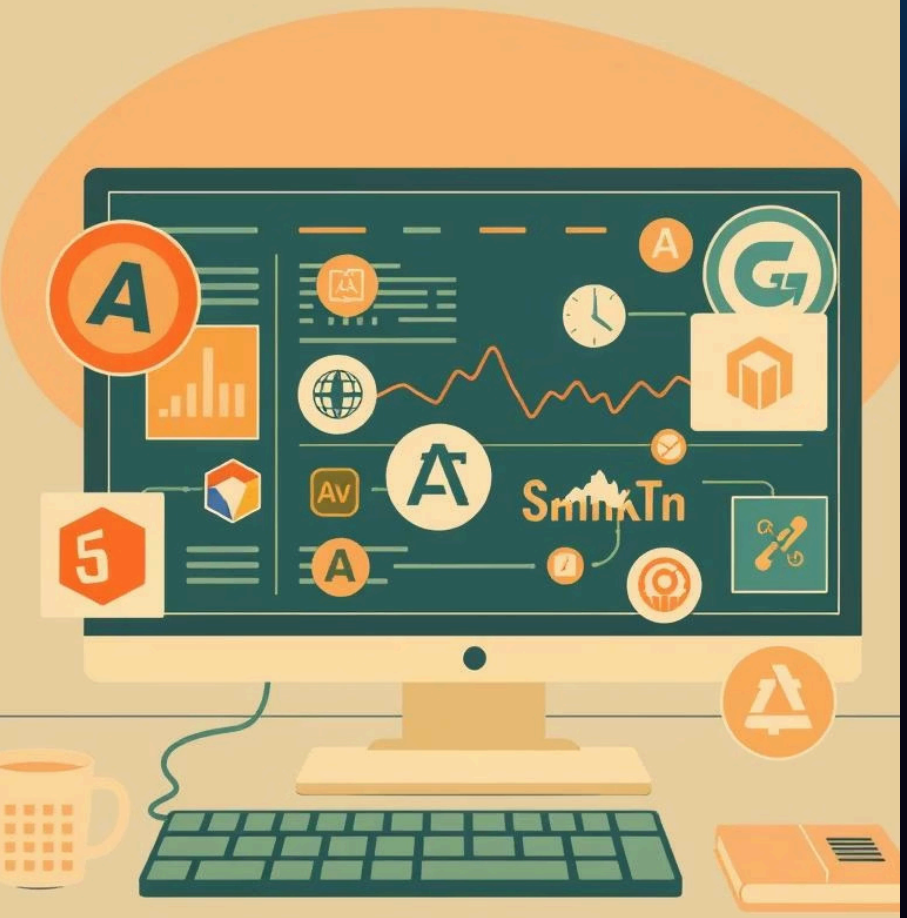
Streamlit Dashboard

Interactive interface for simulation and testing.

6

Docker & ROS2 Integration

Enabling modular and scalable deployment.



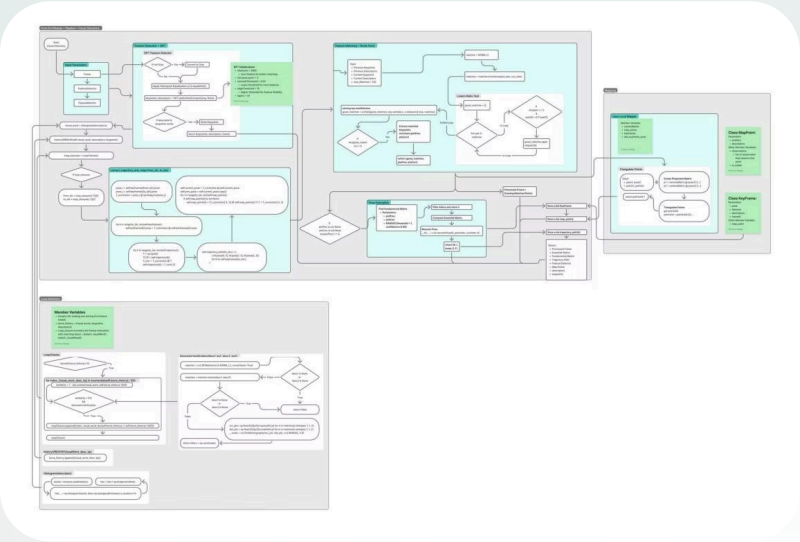
Tools and Technologies

Programming & Libraries

- Python 3.8+
- OpenCV, NumPy, Scikit-learn
- Streamlit, Plotly

Deployment & Data

- Docker for containerization
- KITTI Odometry Dataset for simulation
- Matplotlib, Plotly for visualization



VSLAM Workflow

The core VSLAM pipeline operates through the following stages:

Each module is implemented in a dedicated class for scalability and debugging efficiency.

Core Modules

Frontend Pipeline

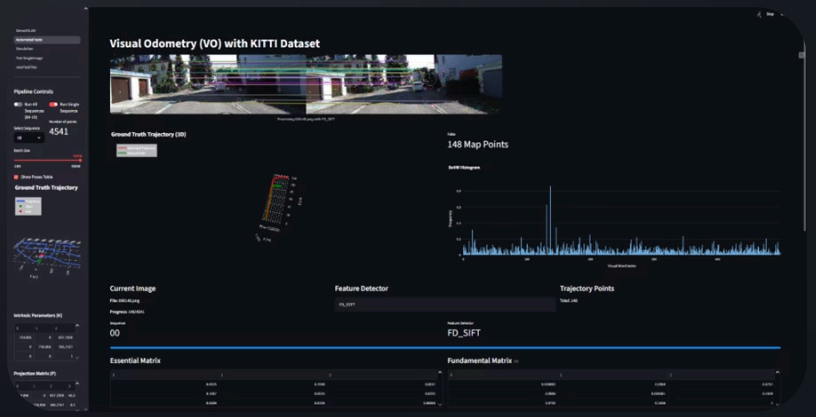
Implements the full VSLAM pipeline, supporting SIFT, ORB, BRISK, Affine variants for feature detection. Pose estimation uses Essential/Fundamental matrix with RANSAC. Loop closure is handled via BoVW.

Mapping & Visualization

Manages landmark triangulation and keyframe maintenance. Provides 3D trajectory visualization using Plotly for clear representation of the agent's path.

Loop Detection

Utilizes a pre-trained Bag-of-Visual-Words (BoVW) model. Employs cosine similarity and geometric verification to accurately identify revisited locations.



Simulation and Testing Interface

The Streamlit Dashboard enables batch simulations across KITTI sequences, visualizing estimated vs. ground truth trajectories and logging MSE/RMSE for each frame.

Key Simulation Components

Simulation.py

- End-to-end simulation with histogram evolution.
- Real-time trajectory generation and 3D animations.
- Displays BoVW loop closure events.

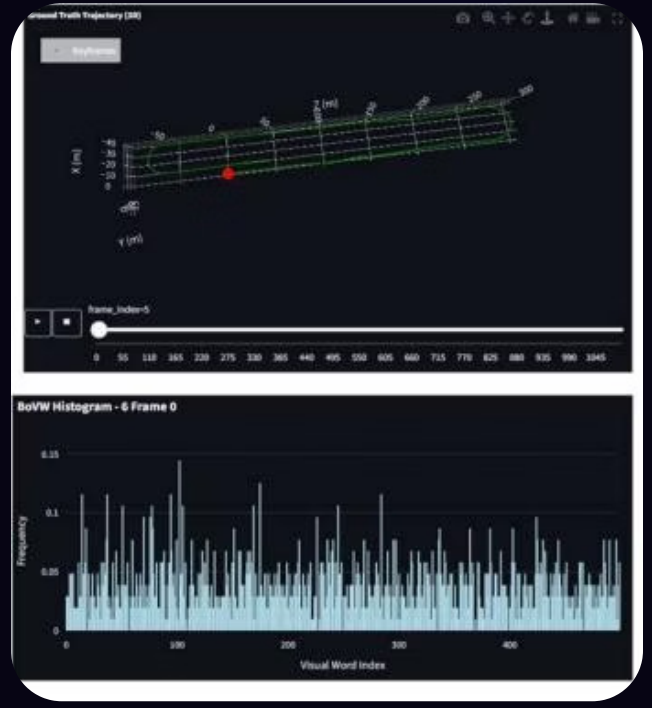
Test_SingleImage.py & viewTestFiles.py

- Matches features between two frames and visualizes descriptors.
- Loads CSVs, compares metrics, displays heatmaps and rankings.

Performance Evaluation

The system was tested using SIFT, ORB, BRISK, and Affine variants across KITTI dataset sequences.

Sequence	Feature Detector	Composite Score	Total Error	RMSE	Average Frame Rate	Error Per Frame
07	FD_AffineSIFT	0.0334	12.931	29.1561	7.3853	0.013
05	FD_AffineSIFT	0.0392	15.958	40.812	7.5036	0.016
06	FD_AffineSIFT	0.0463	19.142	64.2269	7.3566	0.019

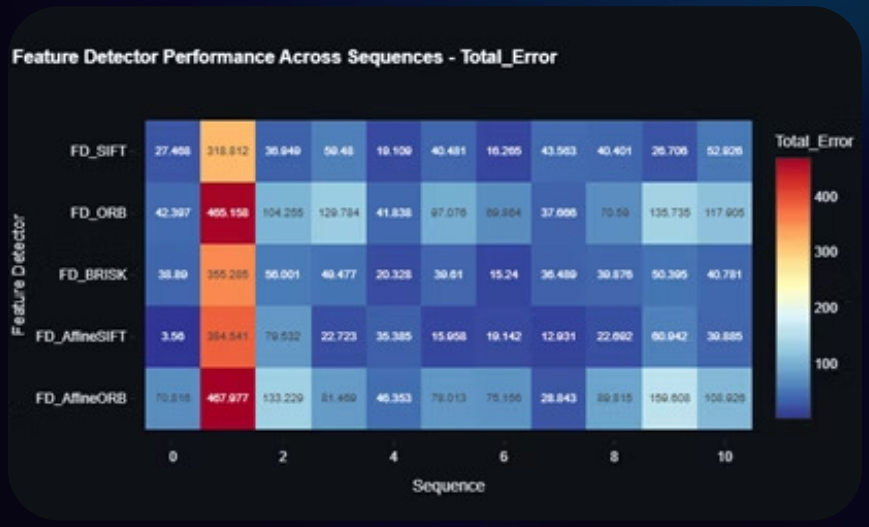


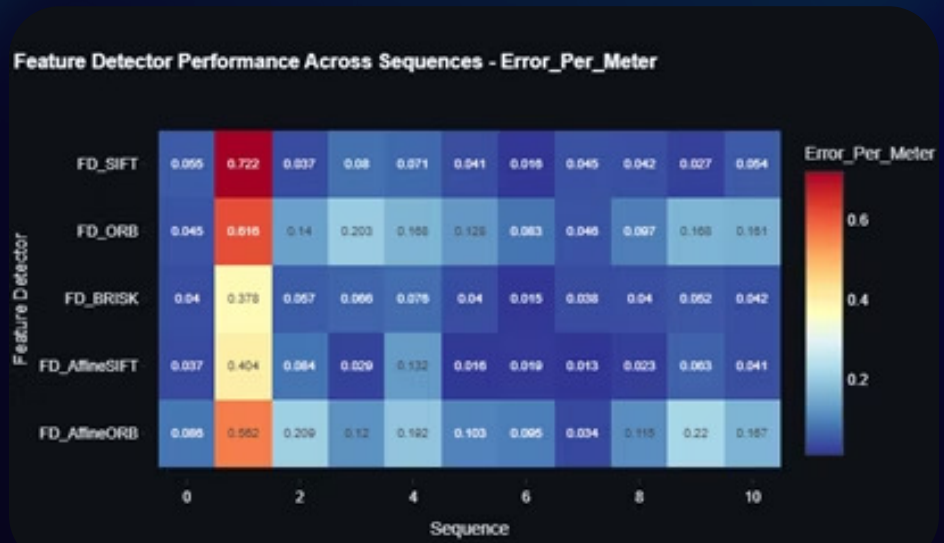
Feature Detector Performance

Feature Detector	Average Total Error	Average Time Taken	Average Error Per Frame
Affine ORB	121.8386	43.6627	0.173
Affine SIFT	63.3901	7.3142	0.0782
BRISK	67.4884	38.6308	0.0767
ORB	119.2971	19.178	0.1688
SIFT	62.0145	14.1204	0.1082

Performance Evaluation

The system was tested using SIFT, ORB, BRISK, and Affine variants across KITTI dataset sequences.





Conclusion and Future Work

This internship provided hands-on experience in VSLAM, covering theory, practical implementation, and performance benchmarking.

Key Learnings:

- VSLAM theory and mathematical foundations.
- Practical implementation of pipeline modules in Python.
- Performance benchmarking of different feature descriptors.
- Use of Streamlit for intuitive simulation.

Future Work:

- Integrating depth sensors for scale-aware SLAM.
- Optimizing for real-time applications.
- Publishing the tool as a ROS2 package.