Final Project Proposal

24783: Advanced Engineering Computation

Project Name: Monocular Visual SLAM System
Team Name: Will_Code_for_Food
Team Members: Aditya Ramakishran, Neel Pawar, Samiran Gode, Shri Kiran Srinivasan

Introduction

SLAM is the computational problem of constructing or updating a map of an unknown environment while simultaneously keeping track of an agent's location within it. Our project objective is to implement a SLAM to localize a monocular camera within a static environment.

Impact

SLAM is being increasingly deployed in a variety of real-world settings, from self-driving cars to mobile devices. SLAM techniques will be increasingly relied upon to provide reliable metric positioning in situations where infrastructure based solutions such as GPS are unavailable or do not provide sufficient accuracy. The global SLAM technology market now has nearly 100 players with a variety of commercialized implementations, and a projected market size of US$ 2866.7 million by 2028.

Learning Objective

Learn how to build complex software with multiple libraries by creating our own monocular visual SLAM system from scratch as shown in Fig. 1 below.

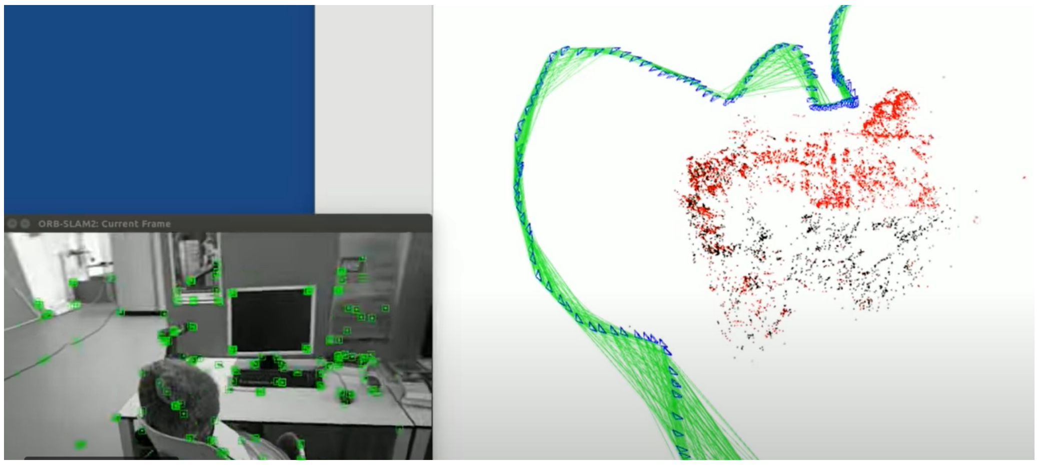

Fig. 1 - Left: Original image with detected key points highlighted
Right: Camera localization and trajectory over time with landmark point cloud

Project Requirements

Build System: Low-level external libraries (GTSAM, Pangolin, Eigen, Doxygen etc.), unit testing. Writing CMake lists to build all of the above

Component Integration: Frontend, Backend and Viewer will communicate with each other. Processes will have different runtimes and will need to be synchronized all while being fast - requires Multithreading.

Data handling: Input is video in the form of multiple image frames. We will need to utilize correct Data structures to ensure optimal space and memory complexity.

Design pattern: Optimal design; Classes employ effective constructors (move/copy) for efficient storage.

Libraries used:

- OpenCV -> For frontend Keypoint detection and tracking
- PANGOLIN -> Visualization
- GTSAM -> Backend for factor graph based optimisation
- EIGEN -> Required for Linear Algebra in GTSAM
- TBB -> Multithreading

System Overview

The system will detect landmark keypoints and track these across the input image keyframes giving us visual factors. We will use these factors to create a posegraph and optimize it incrementally(or in batch) to give us final camera poses and final landmark positions in a map. In parallel we will be showing the map and the input frames with the key points.

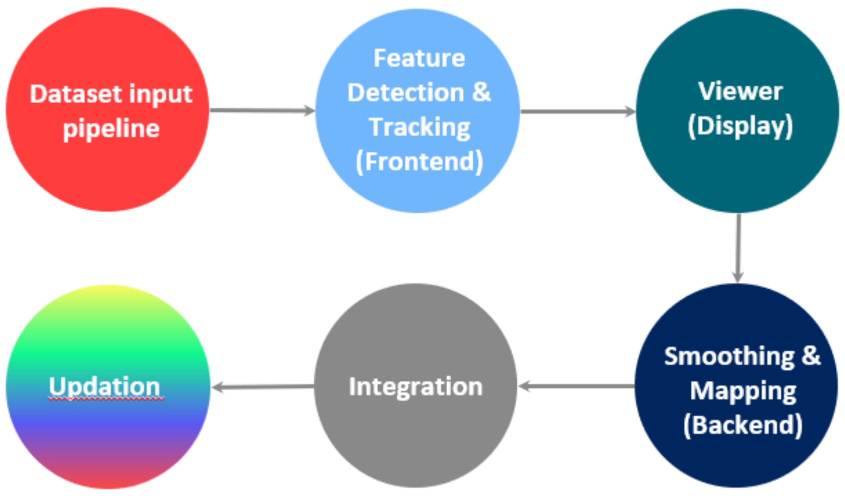
Dataset input pipeline
Feature Detection & Tracking (Frontend)
Viewer (Display)
Smoothing & Mapping (Backend)
Integration
Updation

Fig. 2 - Broad overview of how we plan to tackle the project

Our software pipeline connects each independent module as described above. A final integration of the modules is required to achieve localization objectives and accurately represent the monocular camera motion.