Project Overview and Process

So we will be working on the Project in which we will be implementing Visual Odometry. The purpose of this project is to create a prototype of a robot that has a camera embedded on it, to develop a robotic platform that can map its environment, determine its location, and detect obstacles in real time. The goal is to build a prototype of a robot with an embedded camera that can navigate autonomously based on visual inputs.

1. **Project Set Up** - We are building a **DIY Robot Base** with four motors powered by two motor drivers and controlled using a **Raspberry Pi**. The system will process visual data from a monocular camera and compute motion using Visual Odometry techniques.

   Hardware Requirements:

   - **Monocular Camera** – Used for capturing real-time video frames for VO processing.
   - **DIY Robot Base** – The foundation of the robot, including chassis and mounting components.
   - **Motor Drivers (L293D or L298N)** – To control the four **12V DC motors**.
   - **Power Source** – A **12V battery** for motor power and a separate **5V supply** for Raspberry Pi.
   - **Raspberry Pi 4 Model B** – The main computational unit for processing

   Software Tools & Libraries Used

   - **Ubuntu (on Virtual Machine / VMware Fusion)** – Environment for running Visual Odometry algorithms.
   - **Python** – Main programming language for implementing VO.
   - **OpenCV** – Computer vision library for feature detection & tracking.
   - **ORB-SLAM3** – SLAM implementation for visual tracking.
   - **KITTI Dataset** – Benchmark dataset for evaluating Visual Odometry algorithms.
   - **Matplotlib & NumPy** – Used for data visualization and matrix computations.

2. **Understanding Visual Odometry** - Visual Odometry is the process of estimating a robot's position by analyzing sequential images captured by a camera. We studied various approaches and implemented **Feature-Based VO** using **ORB (Oriented FAST and Rotated BRIEF) features**. - Understanding Vo and its approaches

3. **Building the Robot Base-**
    - Assembling the DIY Chassis with wheels, motors, and motor drivers.
    - Connecting Motors to Motor Driver:
        - Four motors connected to two L293D motor drivers.
        - Each motor driver is powered by a 12V battery.
    - Connecting Raspberry Pi to Motor Drivers:
        - Used GPIO pins 17, 18, 22, 23 for control signals.
        - Used PWM to control motor speed.
    - Setting up the Power System: using an external power source
    - Installing Raspbian OS and Required Libraries on Raspberry Pi.
    - Testing Motor Movement using Python Code.
    - Setting Up Raspberry Pi for programming - Pdf Link

4. **Programming motor movement** - Started Learning about differential drive in Kinematics and how to implement that in our code - So at this step we started learning how to implement motor control using a **Python Class**. - pdf Link
    - **Forward and Backward Motion** using GPIO signals.
    - **Implemented PWM** to control motor speed.
    - **Differential Drive Kinematics** applied for smoother motion.

5. **Research on Visual Odometry Methods -** We studied various Visual Odometry methods from recent research papers and compiled the findings into a structured table.

   All VO methods and their characteristics - for reference see VOMethods.pdf

6. **Implementing ORB SLAM3 on UBUNTU –**
    - **Installed Ubuntu on VMware Fusion**.
    - **Installed ORB-SLAM3 and Dependencies** (OpenCV, Pangolin, Eigen3).
    - **Downloaded KITTI Dataset** for testing.
    - **Built and ran ORB-SLAM3 with KITTI dataset** to track camera motion.

    📌 **Current Status:** Facing issues in running the kitti dataset facing some errors.

7. **Tracking Thread** – The tracking thread is responsible for estimating the camera's position in real time.
    - **Extracts ORB features** from input frames.
    - **Matches features** with previous frames to track motion.
    - **Estimates camera pose** and updates position.

    📌 **Current Status:** Tracking Thread has not been successfully implemented till now

Current Status & Next Steps

✅ **Completed:**

- Built and connected the robot base.
- Implemented motor control using Python.
- Studied Visual Odometry & ORB-SLAM3.
- Installed and tested ORB-SLAM3 on Ubuntu.

🔄 **In Progress:**

- Debugging tracking thread fault.
- Testing ORB-SLAM3 with different KITTI sequences.
- Implementing loop closure detection & map optimization.