第一阶段：地基 —— 矩阵、几何与工程规范

核心目标：熟悉机器人开发的“普通话”（数学）和“书写规范”（日志与格式）。
涉及内容：`Eigen`，`Sophus`，`fmt`，`spdlog`

1. **理论学习**：

- **线性代数**：复习矩阵乘法、逆、特征值分解、SVD分解。
- **三维刚体变换**：旋转矩阵、欧拉角、四元数（Quaternion）、齐次变换矩阵。
- **李群李代数**：了解 及其指数/对数映射（这是 `Sophus` 的基础）。

2. **C++库实战**：

- **`Eigen` (必须)**：它是所有后续库（PCL, Ceres, g2o）的基石。不要只看文档，要写代码实现坐标变换。
- **`Sophus` (推荐)**：用它来代替 Eigen 处理复杂的旋转（李群操作）。
- **`fmt` & `spdlog` (可选)**：从现在开始，你的所有 `std::cout` 都换成 `spdlog::info`，`printf` 换成 `fmt::format`。

3. **练习项目**：

- 编写一个程序：定义一个机器人在世界坐标系的位姿 （使用 Eigen），结合传感器相对机器人的位姿 （使用 Sophus），计算传感器在世界系下的坐标。
- 使用 `fmt` 格式化打印矩阵数据，使用 `spdlog` 记录计算流程。

第二阶段：上帝视角 —— 全局规划与视觉可视化

核心目标：在地图上找路，并把它画出来。
涉及内容：`OpenCV`，`A* / Dijkstra / JPS`，`RRT`

1. **理论学习**：

- **图搜索算法**：BFS（广搜）、DFS（深搜）、Dijkstra。
- **启发式搜索**：A*（核心中的核心）、JPS（跳点搜索，A*的进阶版）。
- **采样搜索**：RRT（快速扩展随机树）。

2. **C++库实战**：

- `OpenCV` **(推荐)**：这里不让你做图像识别。**把 OpenCV 当作绘图板**。创建一个黑白的 `cv::Mat` 作为栅格地图（0是空地，255是障碍物）。
- **STL 容器**：熟练使用 `std::priority_queue`（优先队列）来实现 A* 的 OpenList。

3. **练习项目**：

- 在 OpenCV 画布上随机生成障碍物。
- 手写 *A 算法**（不要调库），找到起点到终点的格子路径。
- 用 OpenCV 把搜索过的节点画成灰色，最终路径画成红色（可视化调试）。

第三阶段：数学之美 —— 轨迹优化与平滑

核心目标：把 A* 走出的“折线”变成“丝滑曲线”。
涉及内容：`贝塞尔曲线`，`B样条`，`QP (二次规划)`，`Eigen`

1. **理论学习**：

- **多项式曲线**：贝塞尔曲线（Bézier）、B样条（B-Spline）、最小化 Snap/Jerk 轨迹。
- **凸优化基础**：理解什么是二次规划（QP），形如 。

2. **C++库实战**：

- `Eigen`：再次进阶，利用 Eigen 求解线性方程组来计算多项式系数。
- **QP求解器**（如 OSQP 或 Eigen 只有简单的）：理解如何构建 矩阵和约束矩阵。

3. **练习项目**：

- 接上一阶段，读取 A* 算出的折线路径点。
- 实现 **B样条（B-Spline）** 或 **贝塞尔曲线** 对路径进行插值和平滑。
- （进阶）构建一个 **Minimum Snap** 轨迹优化问题，用 Eigen 求解系数，让路径不仅平滑，而且速度、加速度连续。

第四阶段：感知与定位 —— 点云处理与前端匹配

核心目标：处理激光雷达数据，弄清楚“我在哪”。

涉及内容： `PCL`, `small_gicp`, `Eigen`

1. **理论学习：**

- **点云基础**：滤波（降采样、离群点去除）、分割。
- **配准算法**：ICP（迭代最近点）、NDT（正态分布变换）、GICP（广义ICP）。

2. **C++库实战：**

- **`PCL` (必须)**：主要学习它的数据结构 `pcl::PointCloud<pcl::PointXYZ>` 和基础滤波（VoxelGrid）。
- **`small_gicp` (必须)**：这是你要重点学的。它是一个现代、高效的 GICP/VGICP 实现。用它来替代 PCL 自带的低效 ICP。

3. **练习项目：**

- 下载两个有重叠的 KITTI 数据集点云。
- 先用 `PCL` 进行降采样。
- 使用 **`small_gicp`** 将两帧点云配准，计算出相对位姿（变换矩阵）。这一步你就能体会到 Eigen 和 Sophus 在其中的作用了。

第五阶段：集大成者 —— 优化与控制

核心目标：让机器人精准地跑起来（后端优化 + 控制）。
涉及内容： `Ceres` / `g2o`, `MPC`, `PID`, `TEB`

1. **理论学习：**

- **非线性最小二乘**：SLAM 后端优化的核心，也是 TEB 规划器的原理。
- **自动控制**：PID（经典）、MPC（模型预测控制，现代控制之王）。MPC 本质上就是在每一步解一个 QP 或 NLP 问题。

2. **C++库实战：**

- **`Ceres` 或 `g2o` (推荐)**：这两个都是非线性优化库。
- 学 **g2o**：有助于你看懂 **TEB** 算法的源码（TEB 基于 g2o 里的超图优化）。
- 学 **Ceres**：Google 出品，文档友好，适合做离线的轨迹平滑或 NLP 求解。

- **MPC 实现**：基于 Eigen 手写一个简单的线性 MPC，或者调用 QP 求解器。

3. 练习项目：

- 复现 **DWA 的升级版 —— MPC**。
- 定义车辆运动学模型（Kinematics）。
- 构建代价函数（和终点的距离 + 速度 + 控制量变化率），将其转化为 QP 问题求解。
- （进阶）使用 **g2o** 定义一个“路径点跟随”的图优化问题，这其实就是手写了一个简易版的 TEB。

总结：知识重合度图谱

看这个图谱，你就知道为什么要这样安排了：

学习阶段	重点算法	必须掌握的库	推荐/辅助库	备注
1. 基础	刚体变换	**Eigen**, fmt	**Sophus**, spdlog	所有算法的数学底座
2. 规划	A*, Dijkstra, RRT	(STL容器)	**OpenCV** (可视化)	逻辑强，库依赖弱
3. 轨迹	B-Spline, 贝塞尔	**Eigen**	求解器(OSQP等)	强依赖矩阵运算
4. 感知	ICP, GICP	**PCL**, **small_gicp**	Eigen	点云处理专用
5. 控制/优化	MPC, TEB, PID	**Eigen**	**g2o**, **Ceres**	MPC需要解方程，TEB需要图优化

建议路线图：

1. 先攻克 **Eigen**（贯穿始终）。
2. 用 **OpenCV** 画格子实现 **A***（搞定全局规划）。

3. 用 **Eigen** 实现 **B样条** 平滑 A* 的路径（搞定轨迹优化）。
4. 玩转 **PCL** 和 **small_gicp** 配准点云（搞定感知）。
5. 最后挑战 **MPC** 或 **g2o/Ceres**（搞定高阶控制与后端）。