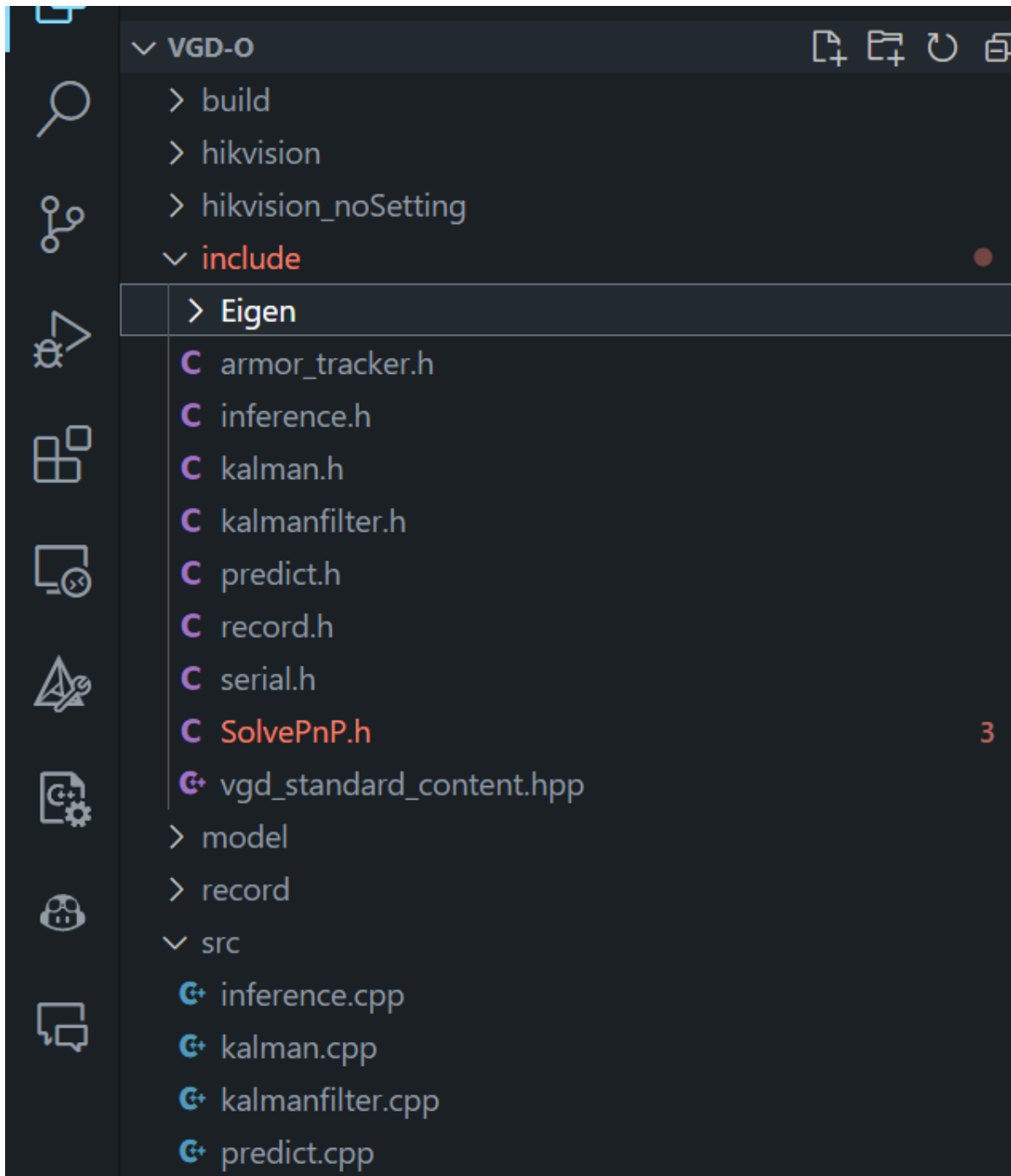


vscode项目配置Eigen库

先参考队内的Eigen库目录



然后就可以直接开始了

1.CMakeLists

```
cmake_minimum_required(VERSION 3.17)

set(CMAKE_CXX_STANDARD 17)

SET(PROJECT_NAME kalmanTest)
project(${PROJECT_NAME})
```

```

include_directories(
    ${PROJECT_SOURCE_DIR}/include/
) ##包含源路径下的所有的头文件

aux_source_directory(src DIR_SRCS) ##自动搜寻指定目录下的所有需要的文件 (.h .cpp)都可
#打印调试src获取的文件
MESSAGE(STATUS "Src file: ${DIR_SRCS}")
#获取src中的所有源代码，存放在sources中

#编译添加可执行程序，命名为project name
add_executable(${PROJECT_NAME}
    main.cpp
    ${DIR_SRCS}
) ##添加可执行目标

```

直接把Eigen库放include文件夹里面就行

2..vscode

task.json参考

```

{
  "version": "2.0.0",
  "tasks": [
    {
      "type": "shell",
      "label": "c++pro",
      "command": "D:/mingw-opencv/mingw64/bin/g++.exe",
      "args": [
        "-g",
        //"${file}",
        //"${cwd}/src/*.cpp",
        "${cwd}/*.cpp",
        "-I", "${workspaceFolder}\\include", //这里是包含主目录下的cpp和src里的
        //cpp文件，再链接好include里的头文件
        "-o",
        "${workspaceFolder}\\Debugger\\${fileBasenameNoExtension}.exe" //
        //把生成的可执行文件放到Debugger文件夹里
      ],
      "options": {
        "cwd": "D:/mingw-opencv/mingw64/bin"
      },
      "problemMatcher": [
        "$gcc"
      ],
      "group": {
        "kind": "build",
        "isDefault": true
      }
    }
  ]
}

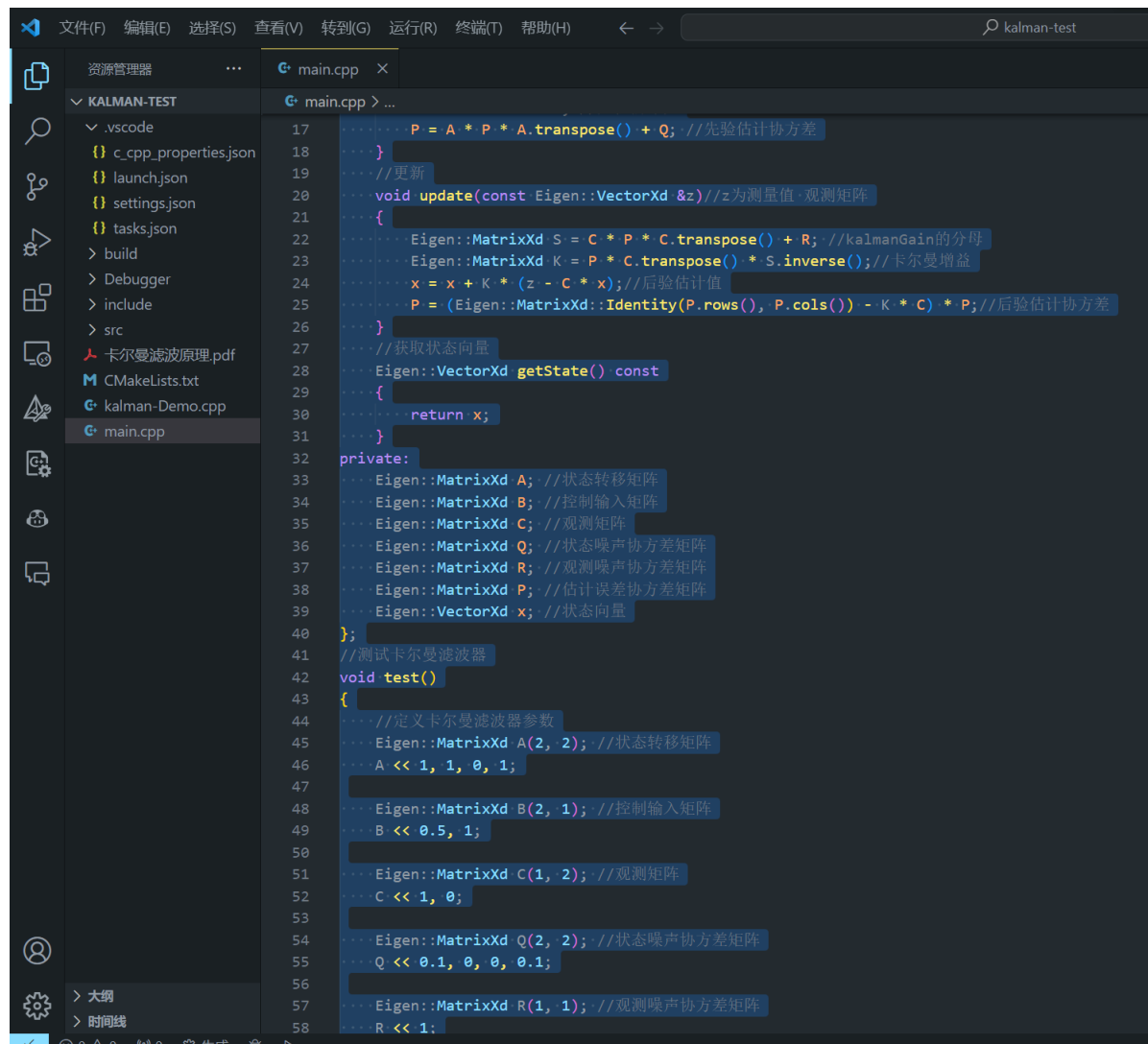
```

上面没有加src目录是因为我的测试文件只有一个main.cpp 所以没有加

3.其他

我的Eigen总目录：G:\temp\vgd\eigen

小demo目录：



```
17 .....P = A * P * A.transpose() + Q; //先验估计协方差
18 .....}
19 .....//更新
20 .....void update(const Eigen::VectorXd &z)//z为测量值-观测矩阵
21 .....{
22 .....    Eigen::MatrixXd S = C * P * C.transpose() + R; //kalmanGain的分母
23 .....    Eigen::MatrixXd K = P * C.transpose() * S.inverse(); //卡尔曼增益
24 .....    x = x + K * (z - C * x); //后验估计值
25 .....    P = (Eigen::MatrixXd::Identity(P.rows(), P.cols()) - K * C) * P; //后验估计协方差
26 .....}
27 .....//获取状态向量
28 .....Eigen::VectorXd getState() const
29 .....{
30 .....    return x;
31 .....}
32 private:
33 ..... Eigen::MatrixXd A; //状态转移矩阵
34 ..... Eigen::MatrixXd B; //控制输入矩阵
35 ..... Eigen::MatrixXd C; //观测矩阵
36 ..... Eigen::MatrixXd Q; //状态噪声协方差矩阵
37 ..... Eigen::MatrixXd R; //观测噪声协方差矩阵
38 ..... Eigen::MatrixXd P; //估计误差协方差矩阵
39 ..... Eigen::VectorXd x; //状态向量
40 .....};
41 .....//测试卡尔曼滤波器
42 void test()
43 {
44 .....//定义卡尔曼滤波器参数
45 ..... Eigen::MatrixXd A(2, 2); //状态转移矩阵
46 ..... A << 1, 1, 0, 1;
47 .....
48 ..... Eigen::MatrixXd B(2, 1); //控制输入矩阵
49 ..... B << 0.5, 1;
50 .....
51 ..... Eigen::MatrixXd C(1, 2); //观测矩阵
52 ..... C << 1, 0;
53 .....
54 ..... Eigen::MatrixXd Q(2, 2); //状态噪声协方差矩阵
55 ..... Q << 0.1, 0, 0, 0.1;
56 .....
57 ..... Eigen::MatrixXd R(1, 1); //观测噪声协方差矩阵
58 ..... R << 1;
```