1、《视觉SLAM十四讲》第二章中关于Eigen库的使用方法的总结：

1. **安装Eigen库**：
   - 在Ubuntu系统中，可以通过命令 `sudo apt install libeigen3-dev` 来安装Eigen库。
   - 安装后，可以通过 `locate eigen3` 查找库的位置，通常位于 `/usr/include/eigen3` 。
2. **包含Eigen头文件**：
   - 在程序中使用Eigen库时，需要包含相关的头文件。可以通过 `#include <eigen3/Eigen/Core>` 来包含核心部分，或者 `#include <Eigen/Core>` 如果已经将Eigen目录移动到 `/usr/include` 中。
   - 也可以在CMakeLists.txt中添加库路径： `include_directories( "/usr/include/eigen3" )` 。
3. **基本矩阵和向量运算**：
   - Eigen中所有向量和矩阵都是 `Eigen::Matrix` ，它是一个模板类，前三个参数为数据类型、行、列。例如， `Matrix<float, 2, 3> matrix_23;` 声明了一个2x3的float矩阵。
   - 可以使用 `<<` 运算符来初始化矩阵和向量，例如 `matrix_23 << 1, 2, 3, 4, 5, 6;` 。
4. **矩阵运算**：
   - 可以进行矩阵的乘法、转置、求和、迹运算等，例如 `matrix_33.transpose()` 、 `matrix_33.sum()` 、 `matrix_33.trace()` 。
   - 矩阵分解和求解线性方程组，例如使用 `colPivHouseholderQr()` 进行QR分解，然后使用 `solve()` 求解方程组。
5. **几何变换**：
   - 使用 `Eigen::Isometry3d` 进行欧氏变换，它是一个4x4矩阵，可以表示旋转和平移。
   - 可以使用 `rotate()` 和 `pretranslate()` 方法来设置旋转和平移。
6. **四元数和旋转**：
   - 使用 `Eigen::Quaterniond` 表示四元数，可以直接从旋转向量 `Eigen::AngleAxisd` 或旋转矩阵 `Eigen::Matrix3d` 转换得到。
   - 四元数可以用来表示旋转，并且可以与向量进行乘法运算来实现旋转。
7. **欧拉角**：
   - 可以将旋转矩阵直接转换成欧拉角，使用 `eulerAngles(2, 1, 0)` 方法，其中参数代表ZYX顺序，即yaw-pitch-roll顺序。
8. **仿射和射影变换**：
   - 对于仿射变换，使用 `Eigen::Affine3d` ； 对于射影变换，使用 `Eigen::Projective3d` 。

这些是Eigen库在SLAM中的基本使用方法，涵盖了从基本的矩阵和向量运算到复杂的几何变换和四元数操作。Eigen库因其高效和易于使用，在SLAM领域中得到了广泛的应用。

2、Eigen库在SLAM中的应用作业

摘要

通过一个具体的代码示例，展示Eigen库在SLAM（Simultaneous Localization and Mapping）中的应用。Eigen库是一个高效的C++库，用于线性代数、矩阵和向量运算、几何变换等。在SLAM中，这些功能是基础且关键的，因此掌握Eigen库的使用对于SLAM的学习和实践至关重要。

1. 包含Eigen库

在C++程序中使用Eigen库之前，需要包含其核心模块和几何模块的头文件。以下是代码示例中的包含指令：

```
#include <Eigen/Core>
#include <Eigen/Geometry>
```

2. 旋转和平移的表示

Eigen库提供了多种方式来表示3D空间中的旋转和平移。

2.1 旋转矩阵

旋转矩阵可以直接使用 `Eigen::Matrix3d` 或 `Eigen::Matrix3f` 来创建。以下是创建单位矩阵的示例：

```
Eigen::Matrix3d rotation_matrix = Eigen::Matrix3d::Identity();
```

2.2 旋转向量

旋转向量使用 `Eigen::AngleAxisd` 表示，它底层不是矩阵，但在运算中可以当作矩阵处理。以下是沿Z轴旋转22.5度的示例：

```
Eigen::AngleAxisd rotation_vector(M_PI/8, Eigen::Vector3d(0,0,1));
```

3. 矩阵运算

3.1 旋转矩阵的转换

可以使用 `matrix()` 方法将 `Eigen::AngleAxisd` 对象转换为旋转矩阵：

```
cout << "rotation matrix =\n" << rotation_vector.matrix() << endl;
```

3.2 坐标变换

使用 `Eigen::AngleAxisd` 或旋转矩阵进行坐标变换：

```
Eigen::Vector3d v(1,0,0);
Eigen::Vector3d v_rotated = rotation_vector * v;
```

4. 欧拉角

可以将旋转矩阵直接转换成欧拉角，顺序为ZYX（即roll, pitch, yaw）：

```
Eigen::Vector3d euler_angles = rotation_matrix.eulerAngles(2,1,0);
```

5. 欧氏变换矩阵

欧氏变换矩阵使用 `Eigen::Isometry3d` 表示，它是一个4x4矩阵，可以表示旋转和平移：

```
Eigen::Isometry3d T = Eigen::Isometry3d::Identity();
T.rotate(rotation_vector);
T.pretranslate(Eigen::Vector3d(1,3,4));
```

6. 四元数

四元数可以直接从 `Eigen::AngleAxisd` 或旋转矩阵创建，也可以将四元数用于旋转向量：

```
Eigen::Quaterniond q = Eigen::Quaterniond(rotation_vector);
```

结果

以下是程序运行的结果：

```
rotation matrix =
 0.924 -0.383      0
 0.383  0.924      0
     0      0      1
(1,0,0) after rotation = 0.924 0.383     0
(1,0,0) after rotation = 0.924 0.383     0
yaw pitch roll = 0.393    -0     0
Transform matrix =
 0.924 -0.383      0      1
 0.383  0.924      0      3
     0      0      1      4
     0      0      0      1
v tranformed = 1.92 3.38    4
quaternion =
    0
    0
0.195
0.981
quaternion =
    0
    0
0.195
0.981
(1,0,0) after rotation = 0.924 0.383     0
```

结论

通过本报告的代码示例和结果，我们可以看到Eigen库在SLAM中处理旋转、平移、欧拉角和四元数等几何变换的强大能力。这些功能对于SLAM系统的实现至关重要，Eigen库的高效和灵活性使其成为SLAM领域的首选库之一。