## 对题目的理解

本课题的关键点在于：

1. 手机3D的交互显示方案与算法/库的选用
2. 传感器的数据同步与数据融合方案与算法选用
3. 导航算法的设计与优化
4. 相机的配置与拍照以及后续的图像处理
5. 相关位置点的图片嵌入与历史浏览
6. 根据地理位置数据以及相机的图像处理识别特定场景

## 交互显示方案

通过调研分析以及实践，总结有如下工具便于开发和使用：

1. OpenGL-ES
2. jPCT-AE（3D游戏开发库，基于OpenGL）
3. vuforia（AR库，最后优化以及创新使用）

其中OpenGL是一个关键性且强大的工具，通过此工具学习掌握了一系列CG相关的知识与技术，包括坐标系的变换、物体材质的渲染以及三维对象的设计流程和目标物体拾取的相关算法等。

**我通过OpenGL-ES完成了最初版本完成了的交互功能如下：**

1. 能够生成坐标系并且可以在坐标系中绘制相关的点线进行数据可视化
2. 可以通过单指、双指操作对三位坐标系进行旋转、平移、缩放以及通过重置按钮将数据清空，坐标系还原的动画设计
3. 能够通过外部手动添加新的三维点并且能够同步刷新在坐标系中
4. 物体的拾取正处于实验测试阶段

**尚待解决的问题：**

1. 图形设计过于简陋
2. 交互操作时存在优化问题导致随着使用时间增加使得主UI卡顿明显

**方案一缺点：**

1. 各种物体图形显示需要从底层设计并且繁琐，需要在最初的数据结构设计方面下足功夫
2. 以后不断添加的需求容易导致之前的代码重构

**方案二优点：**

这是一个不错的3维游戏设计的工具包，基于OpenGL，并且对其的改动不大，能够很好的重构之前使用OpenGL设计的功能，并且鲁棒性有显著提升。通过先接触OpenGL然后使用该工具包，也能够更好的理解OpenGL的相关算法及其效果

**方案二缺点：**

1. 坐标系与OpenGL有区别

## 传感器数据同步与融合方案

Android里面通过订阅系统服务来获取传感器相关信息

### 传感器信息同步：

**方案一：**通过建立具有时间戳的数据结构，然后将系统服务获取的单一传感器信息加入对应的消息队列，如此，需要维护相应的多种传感器对应的消息队列，通过时间戳比对的方式同步传感器信息。

优点：可以很好的准确同步多个传感器信息

缺点：工作量大，需要新的线程来维护消息队列，内存消耗比较大，鲁棒性差

**方案二：**同时监听多个传感器消息，然后通过判断传感器来源，通过收集需要种类的传感器数据，收集完成便完成一次同步

优点：实现方式简单，鲁棒性强

缺点：不能准确的同步相应的传感器数据

折衷方案：由于传感器的刷新频率通过设置可以比较高（<0.1s），而人作为手机的载体很少出现高速运动情况，所以方案二满足需求。

### 传感器信息融合：

关于运动传感器可选的种类有：

1. 加速度传感器（G-Sensor）
2. 磁力传感器（M-Sensor）
3. 方向传感器（O-Sensor）
4. 重力传感器（GV-Sensor）
5. 线性加速度传感器（LA-Sensor）
6. 陀螺仪传感器（Gyro-Sensor）
7. 旋转矢量传感器（RV-Sensor）

由于这些传感器的特征为噪声大，具有迟滞性等，并且测量的数据均为二阶数据。我们需要的导航数据为物体在三维空间中的点，所以需要对上述传感器数据种类以及融合进行妥当处理。

## 导航算法方案

**方案一：**通过线性加速度以及旋转矢量传感器进行积分法求解手机位置以及姿态

优点：数学原理简单

缺点：容易受噪声干扰，对传感器数据精度要求较高，通过测试显示效果不理想

**方案二：**通过加速度传感器、方向传感器来通过EKF滤波以及使用龙格库塔求解微分方程进行估算位置和姿态

优点：位置姿态估算较为准确，不容易受传感器噪声干扰，工业界使用

缺点：数学原理较为复杂，运算开销比较大，我正在学习掌握中

## 相机配置与拍照

通过调研Android的相机使用API，主要分为Android5.0之前的Camera类以及Android5.0之后的camera2的服务框架，由于Android的不断发展以及新版本框架的向下兼容性，选取camera2的服务框架，已经完成的功能有：

1. 自动识别后置摄像头并且获取相关配置数据以及信息数据
2. 能够预览相机数据
3. 拍照前自动对焦
4. 图像数据的旋转、编码保存

## 相关图片信息嵌入与图片浏览

此部分需要完成

1. 相关的数据结构设计，即每一个点的属性以及其成员的相关设计，已经完成，等待测试
2. 图形显示的不同状态的信息数据点不同颜色高亮显示
3. 图形位置的物体拾取算法设计

## 地理位置数据分析及场景识别

此部分依赖于之前的数据准确度，我暂时将人位置信息数据分为以下几种：

1. 正常行走模式（后续通过移动速度更新其他状态）
2. 拍照模式（此位置插入一张照片）
3. 电梯模式（通过Z轴数据的显著性变化）
4. 走楼梯模式（X，Y，Z的数据关联性变化）