我选择的第二个问题是(请将不做的题目编号删除)： F

因水平限制未能作出改进后可运行的代码。下面给出一些思路。对于 clustering 函数：

```
bool clustering(point* p, point a[], int n)
{

    double prex = p->x, prey = p->y, prez = p->z;
    double sumx = 0, sumy = 0, sumz = 0;
    int count = 0;
    for (int i = 0; i < n; i++) {

        if (a[i].number == p->number) {
            sumx += a[i].x;
            sumy += a[i].y;
            sumz += a[i].z;
            count++;
        }
    }
    if (count != 0)
    {
        p->x = sumx / count * 1.0;
        p->y = sumy / count * 1.0;
        p->z = sumz / count * 1.0;
    }
    else {
        int tempval = random2(n);
        p->x = a[tempval].x;
        p->y = a[tempval].y;
        p->z = a[tempval].z;
    }
    if (p->x == prex && p->y == prey && p->z == prez)
    {
        return false;
    }
    else
    {
        return true;
    }
}
```

可对 for 循环应用多线程技术。将 for 循环中的迭代操作封装为一个函数，同时建立多个任务，每个任务在不同的线程中运行。

在 clustering 函数中，添加 thread t[n]; //创建线程

将 for 语句改为：

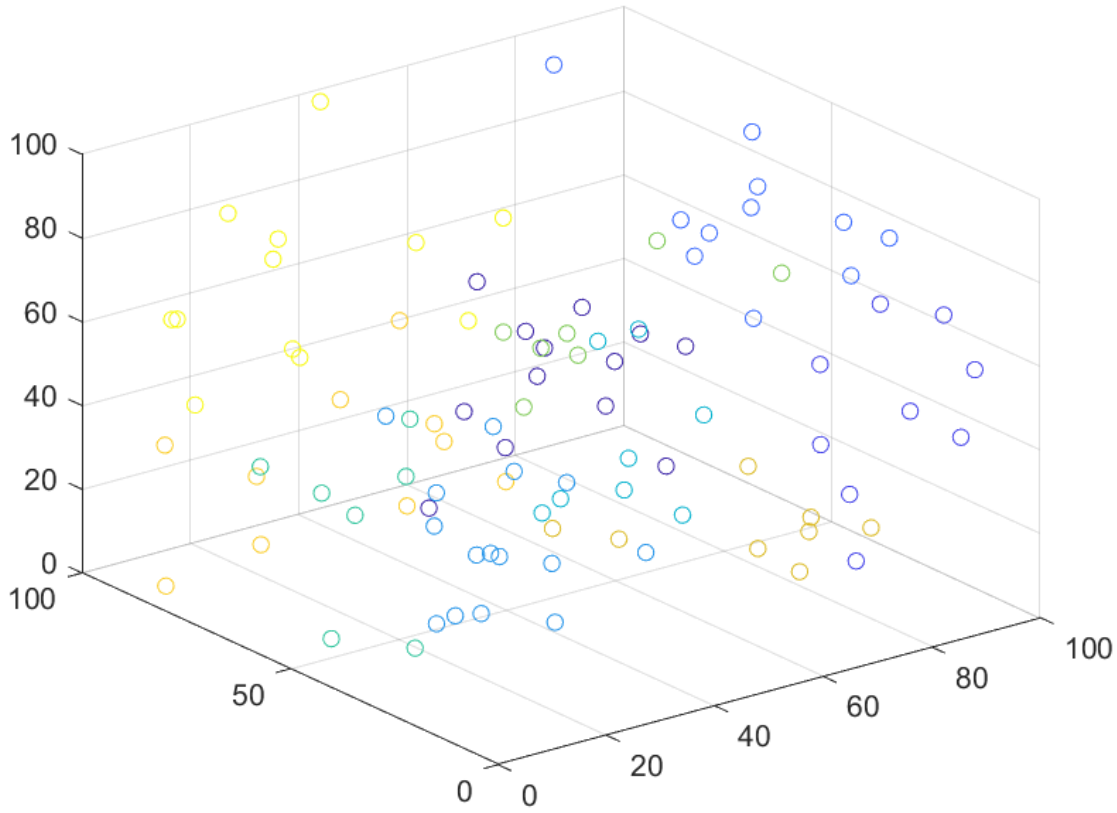
for（int i = 0;i = n;i++）{ t[i]=function()}//其中 function 执行原来循环体的语句，

最后再结束线程：

```
for (int i=0; i<n; i++) {
        t[i].join();
}
```

对于结果正确行的验证，可用 C 语言文件的读取实现。将原代码随机产生的初始结果传入改进后的代码，使二者初始条件相同。运行后编写程序比较二者的 dataclusers、datapoints 文本文档内容是否相同。若相同则验证了改进后代码的正确行。（具体代码及实验所用 txt 文件见附件）

最后，是对结果进行可视化。因为并未成功完成多线程代码的实现，此部分进行简化，采用 MATLAB 自身的 kmean 函数产生结果进行可视化。实验数据由未改进代码产生，存于 data.txt 文件。点云模型点击 Bonus 文件夹中的.m 文件即可运行产生。
运行结果如图：



参考： K-Means 中文介绍： https://www.cnblogs.com/luxiaoxun/archive/2013/05/09/3069594.html

C++thread 多线程英文介绍： https://www.geeksforgeeks.org/multithreading-in-cpp/

视频链接：c++11 并发与多线程视频课程_哔哩哔哩_bilibili