基于 Kinect 的室内环境建图方案

孙沁璇，2014.11.18

1、同时定位与建图（SLAM）

同时定位与建图的概念最早是由 Smith Self 和 Cheeceman 提出用于解决移动机器人在自主导航过程中如何确定移动机器人自身位置的问题。同时定位与建图的过程就是在未知环境中机器人利用自身的传感器感知和建模外部的环境，即能够描述出移动机器人所处的环境，并测量出如障碍、路标等物体在空间中的位置，并同时运用该环境地图确定机器人所处的位置。同时定位与建图问题一经提出就受到了很多国内外研究人员的关注，并逐渐成为移动机器人研究领域的焦点问题之一。是否具有同时定位与建图的能力成为了判断移动机器人是否具备自主能力的关键条件之一。因此，对移动机器人 SLAM 的研究具有十分重要的理论价值和现实价值。

随着技术的进步，SLAM 的应用范围也越来越广，而随着各种移动机器人平台的出现，比如四旋翼机器人和足式机器人，其通过机器人搭载的传感器获得机器人的相对位姿变化十分困难，而在路面环境复杂的应用中通过机载传感器获得的数据可信度不高，所以现在的 SLAM 领域研究的一个方向是将 SLAM 作为一个单独的模块进行研究。即在 SLAM 过程中不依赖搭载传感器设备的内部传感器反馈，只依靠采集设备得到的环境信息来完成 SLAM 任务。室内环境作为人类生活与工作的最重要的场所，应用于室内环境的同时定位与建图研究也就显得尤其的重要。

2、Kinect 简介

Kinect 是美国微软公司为 XBOX 360 游戏机和 Windows PC 机开发的一款体感外设，最初的适用领域只是游戏领域，它可以使游戏者脱离鼠标、键盘、控制板等设备，而是直接利用动作和语音等来控制游戏的整个进度以及进行一系列的人机交互。但是基于其优越的性能以及低廉的成本，之后很多其他领域的研发者们都在不同领域中进行开发，将其应用于游戏领域之外，如人工智能，人机交互，体感互动等领域，其探究和研发取得了不错的进展。

Kinect 的外观如图 1 所示，它由一个基座和一个感应器组成，基座和感应器之间有一个电动的马达，开发者可以通过程序调整感应器的俯仰角度，在一定的应用场合中可以通过俯仰的调整获得很方便的应用效果。在上面的感应器中有一个红外投影仪，两个摄像头，四个麦克风和一个风扇。打开外面的盖子可以看到里面的构造（图 2）。这些感应器可以用来捕捉彩色和深度数据，而通过程序可以直接获得这些数据，再加上已知的摄像头内部参数等就可以完成三维坐标点的计算。在面对 Kinect 方向看，最左边是红外光源，下一个是 LED 指示灯，再下一个是彩色摄像头，最右边是红外摄像头用来采集深度数据。彩色摄像头用来收集 RGB 数据，且成像时支持的最大分辨率是 1280*960，红外摄像头是用来采集深度数据，成像支持的最大分辨率是 640*480。

图 1 Kinect 外观

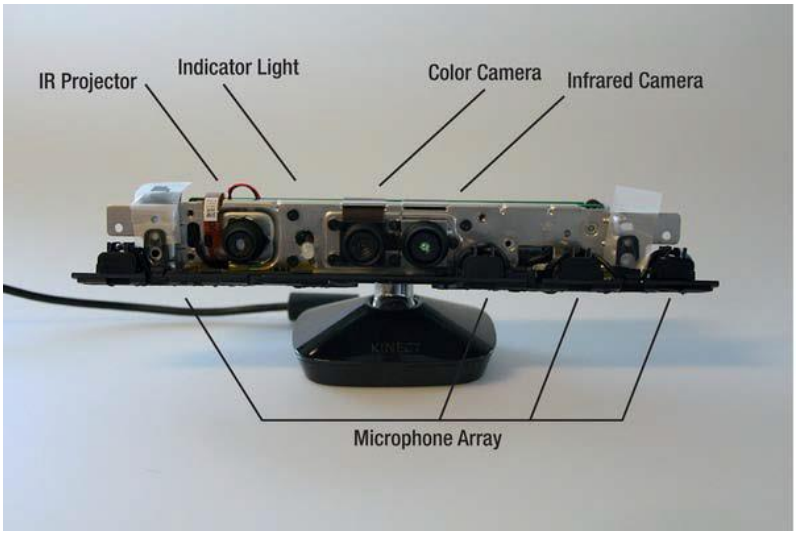


图 2 Kinect 组成

Kinect 将一个标准 RGB 摄像机与深度传感器相结合，对于图像中的每一个像素点，都能得到其对应的颜色信息及深度信息，利用颜色与深度信息以及摄像头的内部参数便可以直接得到被采集区域环境的三维特征信息。但由于 Kinect 的红外摄像机对深度敏感距离有限制，一般为距摄像头 4 米以内，所以一般不能得到完整的深度信息，甚至产生一些无效的点，如果不经过处理会影响后期的处理过程。在理想条件下，深度信息的分辨率可以达到 3 毫米。

传统的摄像头无法直接获取深度信息。专业的深度摄像头大多采用 TOF（time of fight），即摄像头主动射出红外光线，红外光线经过物体表面反射后再被摄像头接收，并根据往返的相位差来得到深度信息。Kinect 使用的并不是 TOF 技术，而是 PrimeSense 公司提供的光编码（Light Coding）技术。所谓光编码技术，是用红外光线为测量空间进行编码，从而获得深度信息，Kinect 通过红外摄像头向采集区域中投射满足一定规律的点阵信息，当场景深度发生变化时，摄像头捕捉到的点阵信息也会随之发生变化，基于此，通过分析点阵模式的变化情况便可以推断出场景的深度距离信息。与 TOF 相比，Light Coding 不需要特制的感光芯片，仅依靠普通的 COMS 感光芯片和连续的照明，就能得到周围环境的图像信息。

3、点云数据的处理

由于一般点云数据具有非结构化的特点，点与点之间没有关联信息，这就造成 3D 点云

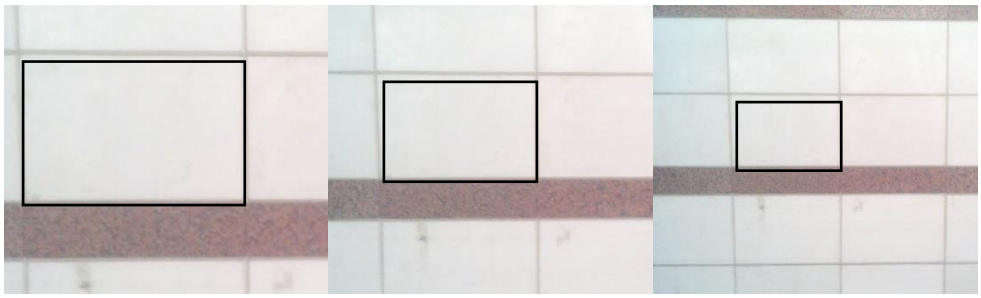
模型通常需要很大数据量才能较为精确地反映真实环境。另一方面，由于传感器本身的物理特性限制，Kinect 获取到的数据不可避免地会包含噪声。同时，因为数据传输的问题，Kinect 对数据进行的规格化处理；以及采集过程中传感器的移动都会加剧噪声。要利用如此巨大的数据实时的完成地图生成和定位的目标，需要有效的特征抽取手段和匹配方法。

（1）Kinect 点云误差分析

由于传感器本身固有的物理特性限制，Kinect 采集得到的点云不可避免地会包含噪声；同时，对采集数据的规格化处理以及采集过程中传感器的移动都会加剧噪声。此外由于 Kinect 的测量方式其在物体边缘处更易产生深度计算错误的问题，所以最终得到的点云数据中包含一些离群点(远离被扫描物体表面的点)；最后，由于扫描场景中物体前后位置产生的遮挡，一些特殊材质的物体(如显示屏等)会吸收红外光，环境中的强光会导致无法得到红外图像，物体的位置限制(超出量程)等因素，使得采集到的点云数据中通常会包含空洞(即无深度信息返回的点)，上述因素均会对抽取平面的结果产生影响，所以对采集到的数据要进行处理，消除或者减弱上述问题对实验结果产生的影响。图 3 是 Kinect 采集到的原始的 RGB-D 图像，在图中可以看出有很多的空洞，并且多数物体的边缘都成锯齿状。对原始点云进行处理的以滤波算法为主，比如高斯滤波、双边滤波和卡尔曼滤波，或者是移除离群点、中值滤波，以及使用不同的算子对其进行局部滤波处理。但是这些滤波算法通常计算复杂度较高，出于实时性的考虑，很多滤波算法难以胜任，在综合考虑了算法效果和实时性，以及点云数据的特性之后，本文选择的数据处理方法包括直通滤波、下采样以及移除离群点三种。其中直通滤波移除距离 Kinect 过远的精度不高的数据点，移除离群点去除由于 Kinect 在测量时产生的错误数据，而下采样则可以在保持空间物体结构不变的基础上有效地降低点云中点的数量。

图3 Kinect采集到的RGB-D图像

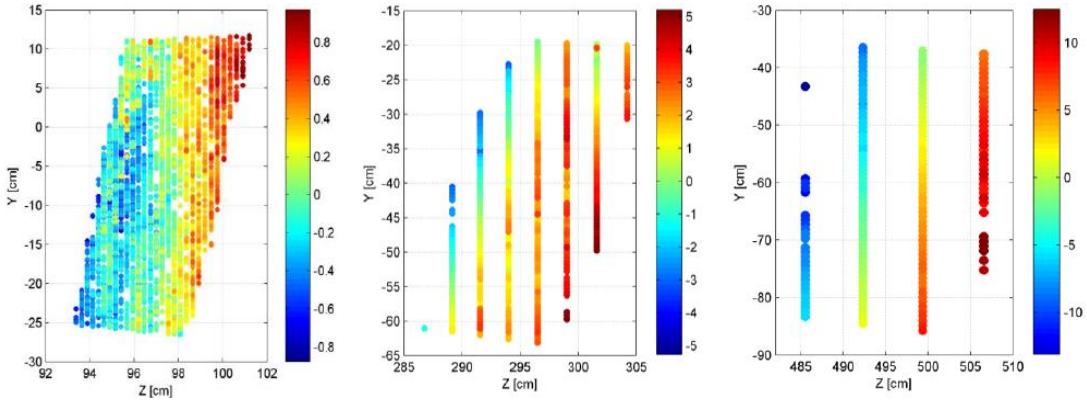
为了决定滤波所要执行的具体的参数，本文进行了一系列的实验，如图 4 是让 Kinect 在不同的距离上对同一面墙进行测量的示意图，图(a)(b)(c)分别是在 Kinect 距离物体 5m，3m 和 1m 时采集到的彩色图像，选择图中黑框中的像素点点的深度信息进行分析，不同图中的黑框对应现实中的区域的大小是一样的。



(a) (b) (c)

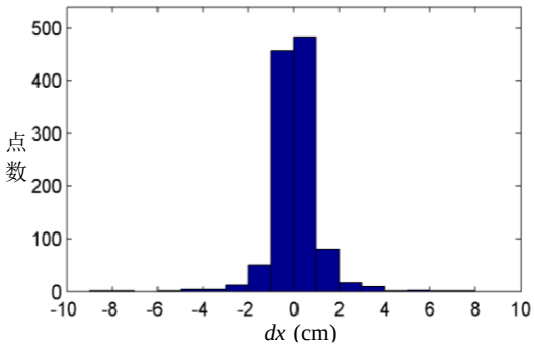
图4 Kinect在不同位置对同一场景进行采集

首先使用属于整个墙面的点拟合得到墙面的方程，计算图 4 中黑框内的对应的点距离拟合得到的平面的距离得到如图 5 所示的结果。图 5 中的(a)(b)(c)分别对应于图 4 中(a)(b)(c)图中黑框中的点，图中的每个点代表真实探测得到的一个点，点的颜色表示该点与墙面拟合得到的平面的距离。从图 5 可以看出，随着 Kinect 距离的增大，落在现实中同样大小区域内的点的数量随着距离的增大越来越少，而且随着 Kinect 距离物体的距离的增大，Kinect 测量的误差明显增大。从结果统计来看，距离 Kinect 3m 的物体其测量结果误差小于 3cm 的点占 84%，当物体距离 Kinect 的距离为 5m 的时候，点的测量误差小于 3cm 的点仅有 64%。单独分析点的 x， y， z 三个分量上的误差得到的结果如图 6 所示的结果。



(a) (b) (c)

图5 在不同距离得到的点云数据与参考平面的距离



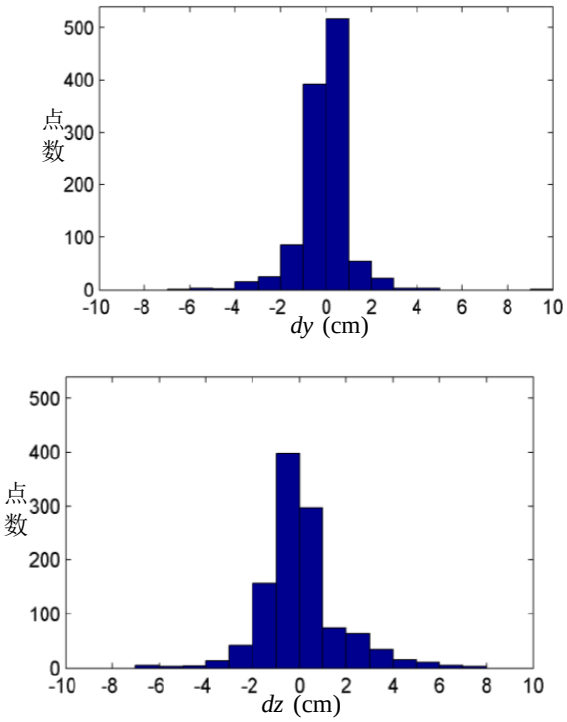


图6 测量结果在XYZ方向上的偏差

由图 6 可以推测在每个分量的测量结果的误差应该服从相同类型的分布。所以本文对 Kinect 的误差产生原因进行了进一步的分析，如图 7 是 Kinect 深度测量的模型，其中深度坐标系的原点C表示红外摄像机的光心，Z 轴与摄像机的光轴重合，Y 轴垂直指向图片的内部，X 轴垂直于Z 轴指向激光发射器I，Y 轴和X 轴与Z 轴垂直满足右手坐标系定则，激光发射器与红外摄像机的距离为b。k是真实世界的一个点，k'是其理论上在红外摄像机图像平面的投影点，o'是实际测量时得到的k点的投影点，d称作测量视差。

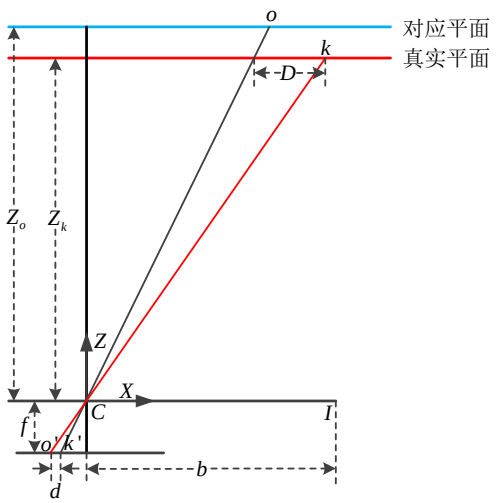


图 7 相对深度和测量误差

Z_k 是 k 点所在真实平面(图中的红色直线表示)距离 Kinect 的真实距离，Z_o 是 o' 所对应的测量平面(图中的蓝色直线表示)距离 Kinect 的距离。D 为 k 点和 o 在世界坐标系下 X 轴方向的偏差，Y 轴方向上的偏差和 X 轴方向的误差类似，由相似关系可得式(1)和式(2)：

$$\frac{D}{b}=\frac{Z_o-Z_k}{Z_o} \tag{1}$$

$$\frac{d}{f}=\frac{D}{Z_k} \tag{2}$$

其中 f 是红外摄像机的焦距。由式(1)和式(2)可以得到：

$$Z_k=\frac{Z_o}{1+Z_o d/fb} \tag{3}$$

在测试条件下 Z_o，b，f 为常数，其中 b，f 可以通过摄像机标定测得。点在 Z 方向的距离和 f 的大小决定了点的尺度，点的平面坐标可以通过式(4)计算得到。

$$X_k=-\frac{Z_k}{f}(x_k-x_o+\delta x) \tag{4}$$

$$Y_k=-\frac{Z_k}{f}(y_k-y_o+\delta y) \tag{5}$$

其中，x_k，y_k 为点在图像平面的坐标，x_o，y_o 为基准点的坐标，δx，δy 为畸变校正参数，同样可以通过摄像机标定得到。在现实情况下中，o' 在每个测量单元上的位置是随机分布的，可以用 $md'+n$ 来代替 d，其中 d' 是归一化的距离差，而 m，n 是一个线性归一化模型的参数，所以式(3)可以变成式(6)的形式。

$$Z_k^{-1}=\frac{m}{fb}d'+(Z_o^{-1}+\frac{n}{fb}) \tag{6}$$

式(6)表示了一个点的逆深度和其相应的归一化误差之间的线性关系，通过一些已知与 Kinect 位置关系已知的物体，比如平面，通过标定就可以测得 m 和 n 的值。对于 Kienct 红外摄像机的标定，只需要在遮挡 Kinect 红外发射器的情况下，在明亮的阳光下或者使用红外光源进行补光就可以使用一般的摄像机标定方法得到红外摄相机的内部参数。

式(6)的其它参数可以通过标定准确确定，假设 d' 服从正态分布，那么对于深度测量方差可以得到如下的计算方法：

$$\sigma_z^2=\left(\frac{\partial Z}{\partial d}\right)^2\sigma_{d'}^2 \tag{7}$$

化简后，会得到如式(8)所示的深度标准差的表达式。

$$\sigma_Z = \left(\frac{m}{fb}\right)^2 Z^2 \sigma_{d'} \tag{8}$$

其中$\sigma_{d'}$和σ_Z分别表示归一化误差及深度的标准差，在更多的距离进行测试得到如图 8 所示的结果。从图中可以看出深度测量的误差与传感器到被测量对象的距离的平方成正比，这里的距离仅指沿摄像机光轴(Z 轴)方向的距离。可以推断在 X 和 Y 轴方向误差也是一个二阶的关于距离的函数。假定图像坐标 x，y 的随机误差可以忽略不计，则由式(8)可以得到 X 和 Y 中的随机误差如式(9)和式(10)：

$$\sigma_X = \left(\frac{mx}{f^2 b}\right)^2 Z^2 \sigma_{d'} \tag{9}$$

$$\sigma_Y = \left(\frac{my}{f^2 b}\right)^2 Z^2 \sigma_{d'} \tag{10}$$

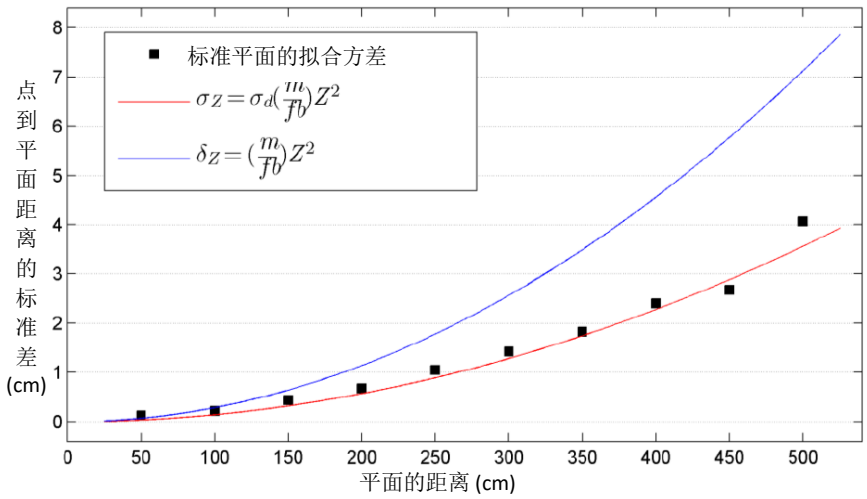


图8 不同距离的平面的标准偏差

点在 Z 轴方向的标准方差σ_Z可以用式(11)简化表示为：

$$\sigma_Z = \tau Z^2 \tag{11}$$

其中τ为一个由摄像机标定参数和$\sigma_{d'}$共同确定的参数。因为之后要使用这些测量得到的数据点进行抽取平面等进一步的运算，需要一个评价标准来判断拟合出的平面的结果的好坏，所以需要一个衡量测量得到的点的准确度的方法。为简化计算，在本文中直接用点在 Z

轴方向的标准方差σ_z来表示点的测量的标准方差$\sigma(p)$，计算方法如式(12)所示：

$$\sigma(p)=1-\sigma_z=1-\tau Z^2 \tag{12}$$

其中$\sigma(p)$的取值范围为$[0,1]$，$\sigma(p)$的取值越接近 1 表示点P的测量越准确，在实际计算时取$\tau=0.167$。

（2）直通滤波

综合考虑一般的室内环境的规模以及 2.1 节中得到的 Kinect 的测量误差随深度的变化而发生改变的规律，本文把在深度方向上与 Kinect 的距离小于 5m 的点作为有效数据点。具体实现时，对采集到的原始点云数据进行处理，去除z分量大于 5 的所有数据点。在进行直通滤波的同时移除那些因为各种原因产生的无效数据点(即图中的空洞)。

（3）数据下采样

若采用一般的数据处理方法，则首先去除错误的数据再进行其它处理，所以对于直通滤波之后得到的点云数据应该先去除错误的数据点(离群点)之后再进行下采样降低点云的密度。但是在实际操作中发现，经过直通滤波之后得到的点云的数据量依旧很大，而在某些空间比较小的室内场景采集到的数据经过下采样处理后得到的点云同原始点云相比，数量没有显著减小，所以如果对得到的点云直接进行移除离群点操作会非常耗时，如本文对一个点数为 307200 的点云进行移除离群点的操作所花费的时间为 2300ms，如此长的数据处理时间是 SLAM 算法所不能容忍的。而对同一个点云先进行尺度为 20mm 的下采样之后，再进行移除离群点操作，总共花费的时间为仅为 53ms。所以为了提高数据处理的实时性，本文先进行下采样操作，再进行移除离群点的操作。

下采样首先建立一个以摄像机坐标系原点为起点的栅格化的三维空间，将点云数据投入到该栅格化空间中，记录所有包含点的栅格。之后对每一个包含数据点的栅格，用该栅格中的所有点的质心来代替栅格中的点，计算方法如式(13)。

$$P_v=\sum_{i=1}^{n}P_i/n \tag{13}$$

其中，$P=\{P_1 \quad P_2 \quad \cdots \quad P_n\}$是每个体素中的点的集合，$P_v$是用来代替点集$P$的点。

使用这种下采样方法不仅可以降低点的总数，还可以保持空间中的如平面，曲面等数学特性不发生改变，除此以外对数据进行下采样还具有一定的平滑数据的功能。进行下采样时选取的栅格的大小(称为下采样尺度)是很重要的，大的栅格可以将点云数据的总数降低到一个很低的级别，但同时也会造成某些几何信息的缺失，小的栅格固然可以保存物体的几何信息，但数据总量没有有效的降低，对之后数据处理的帮助不明显。根据 Kinect 的参数可以计算出其在不同距离上每个像素对应的实际区域的长度：

$$dx=\frac{2L\tan\frac{\alpha}{2}}{640} \tag{14}$$

$$dy=\frac{2L\tan\frac{\beta}{2}}{480} \tag{15}$$

其中L表示距离 Kinect 在深度方向的距离，α和β是 Kinect 在水平方向和竖直方向的

视角，640 和 480 是深度图像在水平方向和竖直方向的分辨率。因为 Kinect 的测量点在空间中不是均匀分布的，所以在进行下采样的时候，对距离 Kinect 不同距离的范围使用不同的下采样尺度，保证最后得到的数据点的数量大约是原始数据点的 10%。本文在实验时，对深度小于 1.5 的数据的下采样尺度为 15mm；对深度从大于 1.5m 且小于 3m 的数据，下采样尺度为 25mm；对深度大于 3m 的数据下采样尺度为 35mm。这样不仅可以保存空间中物体的几何特征，同时可以得到一个比较好的运算速度。

（4）移除离群点

测量必然伴随着错误，而由于 Kinect 本身的测量特性，其更易在物体边缘处产生误差错误，这些因为测量错误产生的数据点被称为离群点。离群点会对计算其附近点的法向量产生比较大的影响，更严重的是在生成地图时，离群点所处的位置会被认为存在障碍物，将实际上可以通行的区域当作不可通行的区域，所以需要将这些点移除掉。

本文判定一个点是否是离群点的标准是距离该点一定范围内是否有足够多的近邻点。如图 9 所示，假设一个点的近邻点距离该点的最大距离为 Δd，这时如果指定点是非离群点的标准是其至少要有 1 个近邻点，那么只有黄色的点会被当作离群点移除，而如果指定非离群点的标准是至少要有 2 个近邻点，那么黄色和绿色的点都将被删除。假设非离群点判断标准是至少有 n 个近邻点的距离小于 d，那么对于点云中的每个点 p，首先找它的 n 个最近邻点，若每个邻近点与 p 点的距离都不大于 d，则这个点不是离群点，反之，则是离群点。但是这样的方法每次都需要遍历整个点云来寻找点的最近邻点，为了加速这个过程，本文使用 K-d 树的方法组织点云，以达到快速查找最近邻点的目的。

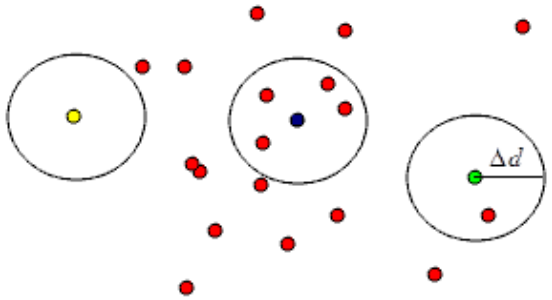


图9 离群点判定

K-d 树实际上是二叉树的空间推广，其区别在于 K-d 树每一层都有一个辨别器，由关键码来决策 K-d 树分支的走向，本文主要针对三维数据点组成的三维空间来进行 K-d 树的构建。空间中的任一点可以表示为 $p_i(x_i, y_i, z_i)$，其中 $i \in N^+$，构建 K-d 树时首先计算点云数据中所有点的 x， y， z 坐标的中值，分别把各个分量的中值作为 K-d 树第 0 层，第 1 层和第 2 层的判别值。

对于任意一点 $p_i(x_i, y_i, z_i)$ 将其插入 K-d 树时，在第 n ($n \in N^+$)层判断点 p_i 属于左还是右分支的比较关键码 m 由其式(16)确定。

$$m = n \bmod 3 \tag{16}$$

当 m 为 0 时，比较点的 x 分量，当 m 为 1 时，比较点的 y 分量，当 m 为 2 时，比较点

的 z 分量。当所取分量的值小于等于比较划分值时，该点就划分到左分支；否则，划分到右分支。重复上述过程知道点云数据中的所有的点都加入到 K-d 树中，图 10 是 K-d 数的示意图。

在判断某个点 p_i 是否是离群点时，首先寻找满足距该点距离小于给定距离 d 的近邻点，并记录邻近点的个数 k 。若 k 小于给定的阈值则点 p_i 是离群点。反之则不是。对于判定是离群点的点 p_i，记录其在 K-d 数中的位置。重复上述的过程直到移除所有的离群点，在查找离群点时，已经被当作离群点记录的点不能作为其它点的近邻点。

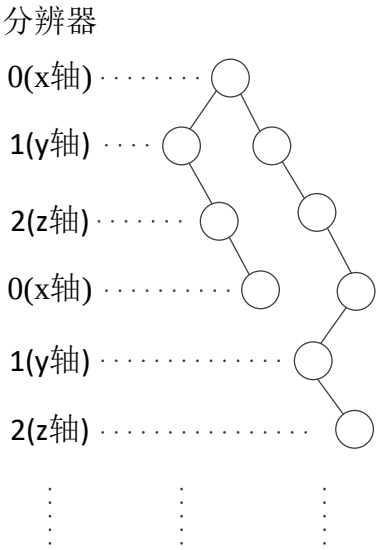


图10 K-d树示意图

移除离群点的过程完成后需要更新 K-d 树，将所有标记为离群点的点从 K-d 树中删除出去，因为之后计算点的法向量时同样需要查找点的近邻点。删除 K-d 数中的节点方法如下：

1) 设待删除的结点为 P ，其分辨器值为 d ；若 P 结点没有孩子结点，则删除该结点；

2) 若 P 结点仅有左孩子结点，则需查找 P 结点左子树中第 d 维最大的结点，设为 P1 结点，删除 P ，用 $P1$ 代替 P 结点；若 P 结点仅有右孩子结点，则需查找 P 结点右子树中第 d 维最大的结点，设为 P2 结点，删除 P ，用 P2 代替 P 结点；

3) 若 P 结点左右孩子都有，设为 P3、P4，则需比较 P 结点左右孩子第 d 维的坐标值，若最大的结点为左孩子 P3，删除 P，用 P3 代替 P 结点，同时右孩子 P4 及其子树成为 P3 的右子树；若最大的结点为右孩子 P4，删除 P ，用 P4 代替 P 结点，同时左孩子及其子树成为 P4 的左子树。

重复上述的过程直到删除所有标记过的离群点。本文的移除离群点方法特别适合移除孤立离群点(即在点附近没有其它点的离群点)，而这类离群点也是 Kinect 设备最常产生的离群点。

4、三维点云地图的生成

（1）基于 ICP 算法的点云匹配

因为 Kinect 在采集图像并生成点云数据的时候，是在每帧独立的摄像头坐标系下，而 Kinect 本身是处在不断运动的过程中，不同两帧点云之间是相对独立的关系，所以首要的问题就是将每一帧独立的点云数据统一在同一个坐标系即世界坐标系下。对于相邻的两帧数据生成的三维点云而言，会有一定范围的重叠区域，而对这部分重叠区域来说，理论上来讲它们在经过一定的旋转和平移变换之后是能够达到完全重合的，同时如果计算出使两帧点云数据完全重合的旋转矩阵和平移向量，也就相当于得到了两帧的摄像头坐标系之间的旋转和平移变换，即完成了两帧点云之间的匹配工作。基于此，本文中考虑应用 ICP（Iterative Closest Point，迭代最近点）算法来实现点云的匹配工作。

迭代最近点算法即 ICP 算法是近几年在点云配准中用的最为广泛的算法，由 Paul J. Besl 和 Neil D. Mckay 在 1992 年首次提出，是一种基于自由形态曲面的配准方法，定义一个阈值，通过迭代来搜索最近点，最终完成多视图的拼合，通常可以用于三维重建中的三维点云或形状的匹配，而且不需要事先设定对应点。

ICP 算法的基本原理如下。假设三维空间 R3 中存在两个点集 P_d 和 P_m，这两个点集各自包含 n 个坐标点，分别为 $P_d=\{P_{d0},P_{d1},P_{d2}...P_{dn}\}$ 和 $P_m=\{P_{m0},P_{m1},P_{m2}...P_{mn}\}$，点集 P_d 中的点经过三维变换后可以与 P_m 中的点一一对应，即

$$P_{mi}=R\cdot P_{di}+T \tag{17}$$

其中 R 是三维旋转矩阵，T 是平移向量。

在 ICP 配准方法中，空间变换参数向量可表示为 $RT=\begin{bmatrix}q_0 & q_1 & q_2 & q_3 & t_0 & t_1 & t_2\end{bmatrix}^T$，其中满足约束条件：$q_0^2+q_1^2+q_2^2+q_3^2=0$。而旋转矩阵 R 与 RT 的对应关系为：

$$R=\begin{bmatrix} q_0^2+q_1^2-q_2^2-q_3^2 & 2(q_1q_2-q_3q_4) & 2(q_1q_3+q_0q_2) \\ 2(q_1q_2+q_0q_3) & q_0^2-q_1^2+q_2^2-q_3^2 & 2(q_2q_3-q_0q_1) \\ 2(q_1q_3-q_0q_2) & 2(q_2q_3+q_0q_1) & q_0^2-q_1^2-q_2^2+q_3^2 \end{bmatrix} \tag{18}$$

ICP 算法流程如下：

（1）计算两个点集 P_d 和 P_m 的重心，并分别对点集进行中心化的处理生成新的点集。

（2）由新的点集计算正定矩阵 N，并计算 N 的最大特征值及其对应的最大特征向量，对应于空间变换参数向量 RT，进而得到旋转矩阵 R。

（3）由两个点集的重心点与旋转矩阵 R 确定平移向量 T。

（4）计算点集 P_d 经过旋转矩阵 R 与平移向量 T 变换后得到的点集 P_d'，并计算两个点集之间的距离平方和值作为迭代判断数值。

（5）当满足一定条件后停止迭代，若不满足则重复迭代过程。

通过以上步骤就可以计算出旋转矩阵 R 以及平移向量 T，进而得到变换矩阵，将两个数据点集以最小的误差配准到同一个坐标系下。

但是利用 ICP 算法完成点云匹配时有两个重要的问题需要解决。

首先是当前处理器下算法运行速度的问题。根据 Kinect 采集的信息所生成的三维点云

每帧的数据量是很大的，滤波前每帧数据量为480*640即307200，经过滤波后每帧也有19000左右的数据量，而ICP算法的运算量是O（N*N），所以尽管点云数据在传输会经过一定的滤波处理大幅地降低了数据量也排除了一些额外的干扰，但对于一般PC机的处理器来说每帧如此巨大的运算量也无法满足实时性的要求。通常来讲，针对 ICP 算法迭代过程运算速度无法达到要求这一问题，可以采用kd-tree（k-dimensional tree）的方法来加速最近点的搜索过程进而提高迭代过程整体的运算速度。kd-tree 是一种数据结构，用来分割 k 维空间，主要可以用在多维空间的数据搜索等领域。

其次，ICP算法能够实现精确配准的前提条件是，两个点集经过变换后能够完全重合，即点集中的每一个点与另一点集中的每一个点是一一对应的关系，也就是说，存在一种旋转和平移变换，使得某一点集在变换过后可以与另一点集实现完全重合。而在本文的应用环境中， Kinect 的位置是在不断的变化中，所以相邻两帧点云所对应的真实环境中的场景是一种存在重叠部分但并不完全一致的情况，这样情况下 ICP 的配准效果往往会大打折扣。因为 ICP 算法过程中，两个点集中的每个点都会参与运算，而这种情况下，其中有很多的点其实并不能在另外一个点集中找到对应点，对于 ICP 算法来说，两个点集中的点并没有严格的对应关系，只是在不断迭代的过程中使整体的配准能达到一个最优的状态，于是这一部分无法在另一点集中寻找到对应点的坐标点一定程度上就变成了一种干扰，使得迭代过程中将它们也纳入一种最优状态的考虑中，影响了配准的效果。

为了解决这两个问题，除了要在配准前对点云进行滤波处理，降低点云数据量之外，还需要运用一定的方法对两个点集中的点进行筛选，过滤掉那些在另一点集中找不到对应点的点，使得筛选出来的两个点集尽量满足变换后能够重合的条件，从而提高 ICP 算法的准确性。这就涉及到两个点云中对应点或对应区域的选取，这也是上文所提到的 SLAM 过程中经常会遇到的难题之一，也就是两个不同视图中的某些点或区域是否对应于真实三维空间的同一物理对象。因为若要做到筛选出的两个点云子集数据能满足 ICP 算法的变换后重合的条件，就必须要保证这两个点云子集在三维空间中的对应关系，只有这种对应关系得到满足，才有可能会实现两个点云子集中的点达到一一对应或近似一一对应的条件。考虑到在利用Kinect获取到的信息生成点云数据的时候，实际上通过计算得到的点云数据是有序的，即点云中的每一个点都对应于彩色图像中的一个像素点，所以本文采用的解决办法是在彩色图像上进行特征点的提取和匹配，再用这些提取到的特征点来定位点云中的对应区域。

（2）图像特征提取与 SURF 特征检测算子

由上文可知，若想要找到两帧点云的对应区域，可以对相邻的两帧图像进行特征点的检测与匹配，再将这些特征点投影到三维空间的坐标点云中来确定两帧图像对应的点云区域。要完成这一任务，要求特征提取的过程有较强的鲁棒性，对于尺度和旋转都具有一定的不变性，而且能够基本满足实时性的要求。基于以上考虑，本文拟采用SURF特征检测算子来完成对特征点的提取。

在SURF算法之前Lowe等人提出的SIFT（Scale-invariant Feature Transform，尺度不变特征转换）算法也是一种鲁棒性较好的尺度不变的特征描述方法，广泛应用于人脸识别、图像拼接、图像配准等领域。但是SIFT算法的计算数据量大，复杂度高，计算耗时长，适合于可以进行离线数据处理的系统，并不适于实时性的方法和系统。2006年Bay等人在SIFT的基础上首次提出了SURF（Speeded Up Robust Feature）算法。SURF算子由SIFT算子改进而来，最大的特征在于采用了haar特征以及积分图像（Integral Image）的概念，在基本保证了SIFT算法原有的鲁棒性强以及尺度不变性等特点的基础上，大大加快了程序的运行速度，为实时性运行的系统提供了一个更好的选择。

与SIFT类似，SURF特征点检测依然基于尺度空间的理论，假设图像中一点 $p(x,y)$，

在尺度 σ 上的 Hessian 矩阵定义为：

$$H(p,\sigma)=\begin{bmatrix} L_{xx}(p,\sigma) & L_{xy}(p,\sigma) \\ L_{xy}(p,\sigma) & L_{yy}(p,\sigma) \end{bmatrix} \tag{19}$$

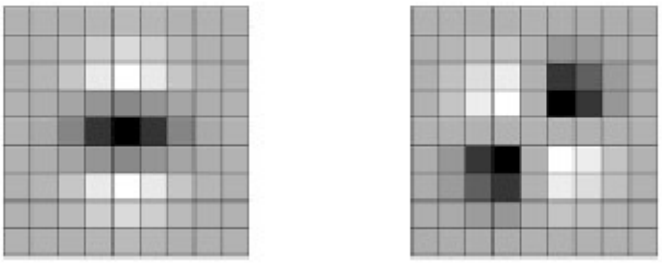
其中 L_{xx} 的含义是高斯滤波二阶导数 $\frac{\partial^2}{\partial x^2}g(\sigma)$ 在 $p(x,y)$ 点处与整个图像卷积的结果，L_{xy} 与 L_{yy} 的含义与 L_{xx} 类似。

Bay 等人在论文中提出用方框滤波近似代替二阶高斯滤波，用积分图像来加快卷积的计算速度，用盒子型滤波器对实际的滤波器进行近似（图 11，图 12）。积分图像计算的是图像中某一个矩形区域内的像素和，例如图像 $I(x,y)$ 在像素点 $p(x,y)$ 处的积分图像定义为

$$I_{\Sigma}(p)=\sum_{i=0}^{i\le x}\sum_{j=0}^{j\le y} I(i,j) \tag{20}$$

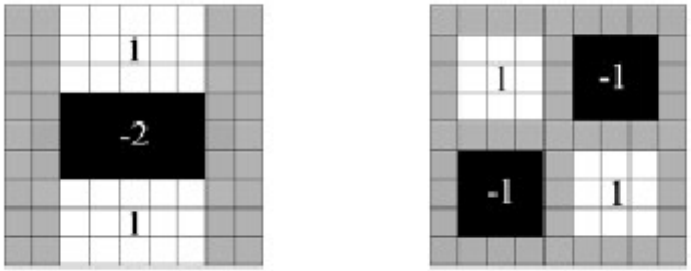
如果一个图像的每一个像素点的积分图像都已经计算出来，则对于这个图像来说，计算任意一个矩形区域内像素和的时候，只需要利用这个矩形区域的四个顶点对应的积分图像，进行三次加（减）法的运算便可以完成，因此在总体上便提高了效率。

而对于盒子滤波器来说，可以假定其每一个连续区域内的权值是相同的，这样一来，计算盒子滤波器与图像在某点处的卷积等价于计算原图像在该点的积分图，而积分图的计算如上文所说只需要三次加（减）法就可以完成。大大减少了计算量。



（a）计算L_{yy}　　（b）计算L_{xy}

图 11 计算 $L(p,\sigma)$ 的实际模板[21]



（a）计算L_{yy}　　（b）计算L_{xy}

图 12 计算$L(p,\sigma)$的近似模板（Box Filter）[21]

若用D_{xx}，D_{yy}，D_{xy}表示采用近似模板后得到L_{xx}，L_{yy}，L_{xy}的近似值，则 Hessian 矩阵的行列式可以近似表示为：

$$\det(H)=D_{xx}D_{yy}-(0.9D_{xy})^2 \tag{21}$$

其中 0.9 是通过实验得到的一个调节参数，理论上此调节参数的值与尺度 σ 有关，但一般都被设置为常数，实验证明对结果不会有太大的影响。

在构建尺度空间时，SIFT 是通过不断对图像进行高斯平滑与降采样的过程来得到金字塔，在这个过程中高斯滤波器的大小是不会变的，而与 SIFT 不同的是，SURF 只是改变滤波器的大小，图像本身不进行变化（图 13）。

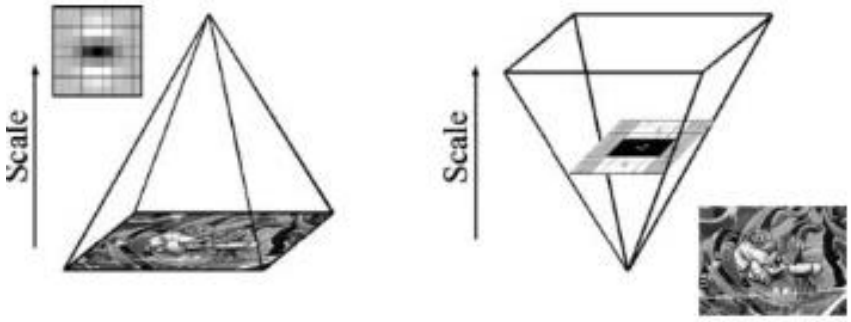


图 13 构建尺度空间（左 SIFT，右 SURF）

与 SIFT 相类似，在尺度金字塔的每一阶中，选择四层的尺度图像，如果图像尺寸远大于模板大小，则继续增加阶数。用 Hessian 矩阵求出极值后，在$3\times3\times3$邻域内进行非极大值抑制，然后在尺度空间和图像空间中进行插值运算，得到稳定的特征点位置及所在的尺度值。

为了保证旋转不变性，需要对每个特征点进行主方向的确定。具体方法是计算一定半径（6σ）内邻域内的点在 x、y 方向的 Haar 小波响应，再将高斯权重系数加在这些响应上，使得离特征点越近的响应贡献越大，离特征点越远的响应贡献越小，然后遍历整个圆形区域的$60°$范围内响应相加从而形成了新的矢量，这样一来最长的矢量方向即可确定为该特征点的主方向。之后便可以沿主方向构建一个正方形，并针对每一个特征点，形成一个 64 维的描述向量。

描述向量也包含了特征点邻域的信息，用向量的最近邻匹配法便可以完成两帧图像特征点的匹配。

5、实验结果

实验平台为南开大学机器人与信息自动化研究所的家族服务机器人平台，完成对实验环境的建图，Kinect 可以同时采集分辨率为 640×480 像素的深度和彩色图像。对同一场景持续测量就可以得到该场景的多幅数据，经过平面拟合、匹配与变换获得环境地图。建立的三维点云地图结果如图 14 所示。全程共 342 帧点云数据，平均处理速度为 2~3fps。

图 14 三维点云地图