# 体积测算

# 引言

利用三维重建的技术流程，可以通过输入二维图片序列的方式获取到场景中的三维稀疏点云或者稠密点云，以视觉的方式来表达场景中的信息。虽然这些点云可以描绘出空间中的信息，但是还存在以下两个这些问题需要解决：

1. 在三维重建的过程中，由于点云坐标系都是以第一帧的相机为参考坐标系，因此点云坐标系和真实世界坐标系无法对应，对于很多实际应用场景，都需要获取到该场景的实际水平面，来进行下一步的导航和定位；
2. 由于是单目相机，无法获取特征点的深度信息，因此对于构建出的三维重建点云也没有一个绝对尺度的概念。

在解决上述两个问题的基础上，可以对空间中的封闭物体进行高度，面积或者体积的测算，对比传统方法中用激光雷达等设备来测算体积的方式，现在就可以通过单个摄像头以纯视觉的方式来完成上述过程，并获取到一个精确的结果。

本章将以三维重建的点云结果为基础，为了解决点云的尺度问题，以及求解三维点云的水平面方程，本文将在场景中引入Aruco二维码，因为Aruco二维码本身携带尺度信息，每一种二维码都有唯一的ID值，并且其还具备较为容易检出的角点坐标。在获取到尺度和水平面方程后，随后结合点云信息可以再进一步计算三维场景的体积，具体流程如图所示。

# 1解算平面

对于大尺度的场景，都需要得到该场景的水平面所在的平面的方程，一方面由于物体的遮挡，水平面很难直接通过视觉的方法构建出来，从而导致整个场景非闭合，因此添加水平面后可以使得整个场景封闭；另一方面，水平面的存在能够更好的计算场景的几何特性，例如场景的高度，面积，体积等。由于三维重建本身的点云结果都是基于参考坐标系，没有和世界坐标系对齐，本文考虑到可以结合点云中能够获取到世界坐标系的Aruco二维码作为媒介，进行坐标系的转换。

## 1.1检测2D点坐标

## 1.2寻找对应3D坐标

在上一小结中，可以获得连续视频帧中每一帧的Aruco二维码下侧两个角点的坐标值，同样的，可以直接将如图所示的这些包含二维码的视频帧序列作为三维重建的输入以获取场景的点云，考虑到Aruco的角点被作为特征点时极易被检出，因此只需要通过稀疏点云，即可获取角点在二维图像中的坐标和三维重建点云中的三维坐标之间的对应关系。

在三维重建的过程中，可以获取到每一帧图像中所有特征点所对应的三维重建点云中的三维坐标，无论是稠密点云或者稀疏点云，都可以得到一个对应关系，但是考虑到遍历的效率问题，则选择稀疏点云中的点击作为遍历对象。在寻找对应关系的实际过程中，会遇到2维坐标无法映射到三维坐标点的情况，那么则会优先选择距离二维点坐标欧氏距离最近的点作为替代对象。此外，还需要注意的是，若所布置的Aruco二维码的所有下侧点都位于同一条直线上，那么这样就无法获取准确的水平面方程，因为经过同一直线的平面并不唯一，因此在布置坐标的过程中需要注意将所有Aruco二维码尽可能对立放置。

## 1.3 平面方程解析

根据上述两个步骤，可以将待求的水平面方程问题转化成求解N个三维点所在水平面方程的非线性优化问题。本文选择谷歌开源的Ceres solver库作为解决该非线性问题的工具，Ceres可以解决边界约束鲁棒非线性最小二乘法优化的问题，表达式如下

minxs.t.12\sum i\rho i(\parallel fi(xi1,...,xik)\parallel2)\\lj\leq xj\leq uj

在利用Ceres解决非线性问题时，通常分为以下三个步骤：

1. 构建代价函数，也就是具体问题所对应的目标式，本小结具体解决N个三维点所在水平面的方程；
2. 通过代价函数构建待求解的优化问题；
3. 配置求解器参数并求解问题，在这一过程中主要是设定求解方程的方式。

上述步骤如代码所示，对于任意平面方程，都可以利用以下方程通过确定4个参数的方式来确定唯一解析解。

\begin{equation}ax+by+cz+d= 0\end{equation}

在迭代的过程中，同时需要确定每个带估计参数的初始值，最终可以得到方程的四个参数a，b，c，d。

# 2解算尺度

由单目视觉重建出来的点云地图都存在没有一个确定尺度的问题，地图中的特征点坐标和相机的位姿都是相对尺度，即最常见的采取初始化成功后的前两帧作为单位尺度，后续的所有关键帧都以此为参考确定尺度。这样的做法可以获取到地图中所有描述的绝对尺度，但无法计算出实际尺度，本文通过在场景中引入已知尺度的二维码来确定所构点云地图的真实尺度，主要针对同一场景，通过计算两种SLAM的相机位姿来确定相对尺度和绝对尺度之间的比例，从而确定绝对尺度。

## 2.1计算绝对尺度和相对尺度

首先，获取同一批图像序列的相对尺度，通过三维重建的结果，就可以得到每一帧相机在参考坐标系下的位姿，该位姿为相对尺度，因为只需要考虑到尺度的大小关系，因为不需要讨论不同坐标系之间的相对转化问题。

其次，获取绝对尺度的数据，本文主要提出了两种方法。由图所示同时可以得到相机在某一个确定的Aruco下的位姿，因为二维码自带确定的边长信息，因此可以得到带有绝对尺度的相机位姿（tx，ty，tz）。在实际的工程中需要注意，必须选择不同状态的相机在同一二维码下的位姿，则可以通过多个图像序列获取到多个不同位姿。

## 2.1尺度估计方法

通过上一小节，可以获取到多个针对同一场景的两种SLAM相机位姿估计结果，因为两种SLAM方法估计出的相机位姿是基于不同的坐标系得到的，本文提出利用相机在不同坐标系下移动的欧式距离之间的比例来确定尺度，简化公式为：

同4.2.1节，利用Ceres库得到最优解K值，计算如代码所示，

只需要杰出唯一参数K即可，在计算的过程中需要随机获取任意两帧之间的位姿，以减小误差。

上述步骤如代码所示，对于任意平面方程，都可以利用以下方程通过确定4个参数的方式来确定唯一解析解。

\begin{equation}ax+by+cz+d= 0\end{equation}

在迭代的过程中，同时需要确定每个带估计参数的初始值，最终可以得到方程的四个参数a，b，c，d。

# 3解算体积

根据以上两节可以获取到三维重建后点云的水平面和比例尺度，再此基础上计算出物体的实际体积，本文提出一种计算点云实际体积的方法，距离流程如下：

1. 将所有点云结果投影在水平面上，根据场景的实际空间约束和过滤方法得到点云中的有效3D点集；
2. 通过空间变换，将三维空间点转化为水平面上的点2D点集
3. 对所有2D点集求Delauncy三角形，计算每个三角形对应的水平高度，得到三棱柱体积；
4. 剔除异常三棱柱，将所有有效的三棱柱积分求和得到总体积。

## 3.1获取有效3D点集

通过第3章的方法，可以对场景获取到三维稀疏点云和稠密点云，在生成点云的过程中会把部分非感兴趣区域的内容和一些离散的误差噪音点添加至点云结果中。针对这两种情况，本文分别提出以下对应的解决方法：

1. 针对非感兴趣区域，本文通过三维空间平面方程对非感兴趣区域直接进行切割剔除；
2. 针对随机误差噪音点，本文通过离群点检测算法进行剔除。

对于方案1，在三维重建后的点云中，只需要收集感兴趣区域内的3D点集，如图所示，只有红色框图内区域是感兴趣区域，可以通过上一节求出的水平面方程和二维码的低边点坐标共同约束求出空间切面方程对场景进行切割。

对于方案2，本文采用统计滤波器即为对每个点的邻域进行一个统计分析，并修剪掉一些不符合标准的点，具体方法为在输入数据中对点到临近点的距离分布的计算，**对每一个点，计算它到所有临近点的平均距离（假设得到的结果是一个高斯分布，其形状是由均值和标准差决定），那么平均距离在标准范围之外的点，可以被定义为离群点并从数据中去除。采用统计滤波对点云中的离散点进行滤波对比如图所示。**

空间变换

通过上一小结，可以求出有效的3D点集，在本小节中需要将所有三维点云投影至二维平面上获取2D点集，简略步骤如下：

1. 通过节获取点云水平面和参考坐标系中的xy平面的水平面的法向量P和Q
2. 将所有的3D点集根据以下公式p0投影水平面上，获取新的3D点集p1；
3. 计算旋转角度：

4计算旋转轴：在计算旋转角时可知，旋转角所在的平面为有P和Q所构成的平面，那么旋转轴必垂直该平面。

假定旋转前向量为**a**(a1, a2, a3)， 旋转后向量为**b**（b1, b2, b3)。由叉乘定义得：

所以旋转轴**c**(c1, c2, c3)为

5根据罗德里格旋公式将旋转角，旋转轴的表达转化为旋转矩阵R

6将3D点p2右乘上述旋转矩阵R可以得到一系列在同一平面上（所有转变后的点z值都相同）的3D点集，提取所有3D点的x,y值构成2D点集。

## 3.3获取Dealunay三角形

通过4.2小结，可以得到在由在同一平面上的3D点集转化成的2D点集和每一个2D点集对应的距离值d。在本节中将对2D点集进行三角剖分，获得Dealunay三角形，德洛内(Delaunay)三角网、是一系列相连的但不重叠的三角形的集合, 而且这些三角形的外接圆不包含这个面域的其他任何点，如图所示。

获取Delaunay三角网格的一般步骤为  
（1）构造一个超级三角形，包含所有散点，放入三角形链表。   
（2）将点集中的散点依次插入，在三角形链表中找出其外接圆包含   
插入点的三角形（称为该点的影响三角形），删除影响三角形的公共边，将插入点同影响三角形的全部顶点连接起来，从而完成一个点在Delaunay三角形链表中的插入。   
（3）根据优化准则对局部新形成的三角形进行优化。将形成的三角形放入Delaunay三角形链表。   
（4）循环执行上述第2步，直到所有散点插入完毕。

由于Delaunay剖分具备具备最接近（以最近的三点形成三角形，且各[线段](https://baike.baidu.com/item/%E7%BA%BF%E6%AE%B5)(三角形的边)皆不相交）和区域性（新增、删除、移动某一个顶点时只会影响临近的三角形）的特性：可以对点集所构成的平面进行拟合。

通过以上流程可以获得三角形的集合以及三角形中每个顶点对应的距离，取三个顶点距离的平均值作为三棱柱的高，即可获得所有三棱柱的体积。

* 1. 积分三棱柱

通过以上步骤，可以将三维重建后点云体积的求解转化成对多个三棱柱积分的求解，因为所有三棱柱的体积值都分布在某一个区间内（并不完全服从正太分布），对于部分偏差较大的三棱柱体积值应该通过算法进行剔除。本文采用箱形图的方法来处理这一问题，箱形图不受异常值的影响，能够准确稳定地描绘出数据的离散分布情况，同时也利于数据的清洗。箱形图如图所示，主要需要求解出数据中的5,个特征数据值，包括**下四分位数Q1，上四分位数Q3，**四分位距IQR=Q3-Q1，**以及上限**Q3+1.5IQR**和下限**Q1-1.5IQR**，对于上下限之外的内容需要剔除，最终结合4.3所估算出的尺度，以及所有有效三棱柱的体积按照式积分即可估计出物体中感兴趣区域的实际体积值。**

由单目视觉重建出来的点云地图都存在没有一个确定尺度的问题，地图中的特征点坐标和相机的位姿都是相对尺度，即最常见的采取初始化成功后的前两帧作为单位尺度，后续的所有关键帧都以此为参考确定尺度。这样的做法可以获取到地图中所有描述的绝对尺度，但无法计算出实际尺度，本文提出两种估计尺度大小的方法，这两种方法都主要是通过在场景中引入已知尺度的二维码来确定所构点云地图的真实尺度，主要针对同一场景，通过计算两种SLAM的相机位姿来确定相对尺度和绝对尺度之间的比例，从而确定绝对尺度。

流程图如图~\ref{fig:getVolume\_getK}所示。

\begin{figure}[H] % use float package if you want it here

\centering

\includegraphics[height=12cm]{getVolume\_getK.png}

\caption{估计尺度流程图}

\label{fig:getVolume\_getK}

\end{figure}

\subsection{计算绝对尺度和相对尺度}

首先，获取绝对尺度的数据，由图~\ref{fig:getVolume\_Aruco\_detect}所示同时可以得到相机在某一个确定的Aruco下的位姿，因为二维

码自带确定的边长信息，因此可以得到带有绝对尺度的相机位姿（$T\_x$,$T\_y$,$T\_z$）。在实际的工程中需要注意，必须选择不同状态的

相机在同一二维码下的位姿，则可以通过多个图像序列获取到多个不同位姿。

其次，再获取同一批图像序列的相对尺度，通过三维重建的结果，就可以得到每一帧相机在参考坐标系下的位姿（$T\_x$,$T\_y$,$T\_z$),

该位姿为相对尺度，因为只需要考虑到尺度的大小关系，因为不需要讨论不同坐标系之间的相对转化问题。

\subsection{尺度估计方法}

通过上一小节，可以获取到多个针对同一场景的两种SLAM相机位姿估计结果，因为两种SLAM方法估计出的相机位姿是基于不同的坐标系得到

的，本文提出利用相机在不同坐标系下移动的欧式距离之间的比例来确定尺度，简化公式为：

\begin{equation}K\;=\;\frac{\triangle Pose\_{rel}}{\triangle Pose\_{abs}}

\label{equ:getVolume\_K}\end{equation}

同4.2.1节，利用Ceres库得到最优解K值，计算如代码~\ref{code:chap3:get\_k}所示，

\begin{lstlisting}[

language=C++,

numbers=left,

numberstyle=\footnotesize,

frame=single,

basicstyle=\small\tt,

escapeinside = '',

caption={获取尺度值的~C++~实现},

label={code:chap3:get\_k}]

vector<vector<float>> colmap, opencv;

vector<float> colmap\_data1 ,opencv\_data1;'//待优化量'

'// 第一部分：构建代价函数模型'

struct CURVE\_FITTING\_COST\_k

{

CURVE\_FITTING\_COST\_k (double x,double y)

'// 残差的计算'

template <typename T>

bool operator() (

const T\* const k,

T\* residual ) const

{

residual[0] = T ( \_y ) - ( k[0]\*T ( \_x ) ); // y-ax

return true;

}

const double \_x, \_y;

};

int main ( int argc, char\*\* argv )

{

double k[1] = {0};

'// 第二部分：构建最小二乘问题'

ceres::Problem problem;

for ( int i=0; i<colmap\_data1.size(); i++ ){

problem.AddResidualBlock

(new ceres::AutoDiffCostFunction<CURVE\_FITTING\_COST\_k, 1, 1>

(new CURVE\_FITTING\_COST\_k

(opencv\_data1[i], colmap\_data1[i])),

nullptr,

k);

}

'// 第三部分：配置求解器'

ceres::Solver::Options options;

options.linear\_solver\_type = ceres::DENSE\_QR;

options.minimizer\_progress\_to\_stdout = true;

ceres::Solver::Summary summary;

ceres::Solve ( options, &problem, &summary );

for ( auto a:k ) {

outfile\_kabc<<a<<endl;

}

return 0;

}

\end{lstlisting}

只需要杰出唯一参数K即可，在计算的过程中需要随机获取任意两帧之间的位姿，以减小误差。

通过坐标点估计尺度

对于尺度的估计，可以通过场景中已知点之间的距离通过投影获得点云之中点的实际坐标，但是空间中的已知点难以确定，而且也不易保持固定不变，符合以上规则的点的数量又较少。因此考虑到

场景中二维码标记的角点为研究对象，可以通过SLAM求得每一个二维码在图像中的位置以及四个角点的位置，因为角点极易识别，且任意两个角点之间的实际空间距离都能够简易测量得到。

通过上一章三维重建的流程，可以获取到每一个点在二维图像中和三维空间点之间的对应关系，如图所示，表示的是同一个点在二维和三维之间的对应。因为图像中的任意两个二维码角点之间的距离都是已知的，那么也可以得到三维坐标中任意两个角点之间的绝对尺度，将这一尺度因子应用到整个地图中进行缩放，即可获得一个带有真实尺度的地图。