**第4章 基于纯视觉的堆体体积测量方法研究**

4.1 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，以视觉特征的方式来表达场景中的信息。再进一步，可以利用重建后的堆体点云来测量其体积，这些点云可以描绘出空间信息，对对于体积测量的实现，还存在以下问题需要解决：

1. 在三维重建的过程中，由于点云坐标系以相机坐标系为参考坐标系，因此点云坐标系和真实世界坐标系无法对齐；
2. 对于大尺度的堆体场景，需要得到该场景的水平面所在方程，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，；

3. 由于三维重建的信息来源于单目相机，所构建出的点云也没有绝对尺度，无法得到实际的体积真值。

在解决上述两个问题的基础上，就可以对堆体进行高度，面积或者体积的测量，对比传统方法中用激光雷达，或者称重器械来测算体积的方式，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，以及求解堆体三维点云的水平面方程，本文将在场景中引入二维码视觉标签，依靠其因为携带尺度信息和唯一ID值的特性估计出尺度信息，并且还具备易检出的角点坐标用来估计水平面方程。在获取到尺度和水平面方程后，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~\ref{fig:4GetV\_pipeline}所示。

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

\centering

\includegraphics[height=5cm]{getVolume.png}

\caption{体积测算流程图}

\label{fig:getVolume}

\end{figure}

4.2 堆体水平面方程解析方法

4.2.1 2D/3D点坐标检测

对于水平面的求解，其基本思路为求解落在堆体水平面上二维码的角点所在的坐标平面，或者求解已知角点和堆体水平面的实际空间位置关系，通过角点求解堆体实际水平面方程，二维码如图~\ref{fig:2VSLAM\_Marker\_detection}所示，将其作为媒介来估计平面方程。

首先检测2D点，可以利用二维码检测算法检测出场景中二维码的ID值，位置，大小，角点坐标等信息，检测结果如图~\ref{fig:4GetV\_Marker\_detection}所示。因为在场景中，二维码的布置都会有下侧两个角点直接落在水平面上，因此可以认为所有二维码的下侧角点所构成的平面即是场景的水平面。那么接下来即可将水平面的求解转化为二维码角点所在平面的求解。在检测的过程中，需要将所有2D图片的索引和角点的位置记录下即可。

其次再检测3D点，通过上述步骤，可以获得连续视频帧中每一帧的二维码角点的坐标值，同样的，可以直接将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，考虑到二维码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，会遇到二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二维码尽可能分散放置。

3.2.3平面方程优化方法

根据上述步骤，可以将待求的堆体水平面方程问题转化为求解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：

\begin{equation}

\begin{array}{l}\underset x{min}\;\;\frac12\sum\rho\_i(\parallel f\_i(x\_{i\_1},...,x\_{i\_k})\parallel^2)

\\s.t.\;\;l\_j\leq x\_j\leq u\_j\\\end{array}

\end{equation}

其中$\rho\_i(\parallel f\_i(x\_{i\_1},...,x\_{i\_k})\parallel^2$为残差块, $f\_i(.)$为代价函数，代价函数的求解依赖于一系列参数$[x\_{i\_1},...,x\_{i\_k}]$所有参数构成参数块,限制边界分别为$l\_j$,$u\_j$。$\rho\_i$为损失函数，其作用是减少异常值对优化结果的影响。

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~\ref{code:chap3:get\_plane}所示。

\begin{lstlisting}[

language=C++,

numbers=left,

numberstyle=\footnotesize,

frame=single,

basicstyle=\small\tt,

escapeinside = '',

caption={获取平面方程参数的~C++~实现},

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

vector<vector<float>> plane\_data1;

vector<float> planex\_data1 ,planey\_data1, planez\_data1;'//待优化量'

'// 第一部分：构建代价函数模型'

struct CURVE\_FITTING\_COST\_plane

{

CURVE\_FITTING\_COST\_plane (double x,double y,double z)

:\_x( x ), \_y ( y ), \_z ( z ) {}

'// 残差的计算'

template <typename T>

bool operator() (

const T\* const plane,

T\* residual ) const

{

residual[0] = plane[0]\*T(\_x) + plane[1]\*T(\_y)

+ plane[2]\*T(\_z) + plane[3];

return true;

}

const double \_x, \_y, \_z;

};

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

{

double plane[4] = {0.1,0.1,0.2,0};

'// 第二部分：构建最小二乘问题'

ceres::Problem problem2;

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

problem2.AddResidualBlock

(new ceres::AutoDiffFunction<CURVE\_FITTING\_COST\_plane,1,4>

(new CURVE\_FITTING\_COST\_plane

(planex\_data1[i], planey\_data1[i],planez\_data1[i])),

nullptr,

plane);

}

'// 第三部分：配置求解器'

ceres::Solver::Options options2;

options2.linear\_solver\_type = ceres::DENSE\_QR;

options2.minimizer\_progress\_to\_stdout = true;

ceres::Solver::Summary summary2;

ceres::Solve ( options2, &problem2, &summary2 );

for ( auto a:plane ) {

outfile\_kabc<<a\*1/plane[3]<<endl;

}

return 0;

}

\end{lstlisting}

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\label{equ:plane}\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。

3.3 堆体尺度估计方法

由单目视觉重建出来的点云地图都存在没有确定尺度的问题，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，本文提出两种结合二维码估计尺度大小的方法，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，计算两种SLAM的相机位姿或者两种SLAM中地图点坐标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。

3.3.1 位姿法估计尺度

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二维码下的位姿，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，必须选择不同的相机对于同一二维码下的位姿，可以通过多帧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。

以上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，本文提出通过位姿来估计尺度的方法，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~\ref{sec:4.2.2}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}

只需要解出唯一参数K即可，在计算的过程中可以随机获取任意两帧之间的位姿，以减小误差。

3.3.2 空间坐标点估计尺度

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~\ref{fig:GstV\_sizebyPoint}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。

3.4 本章小结

本章研究了基于传统三维重建系统的实现流程，包括数据的采集，特征的提取匹配，SfM过程获取稀疏点云，MVS过程获取稠密点云等，只需要向系统中输入围绕堆体拍摄的连续图像，即可得到关于堆体场景的点云模型。

本章揭示了传统三维重建存在的问题，包括系统耗时以及点云模型结果精确度不高等问题，提出了基于传统三维重建的改进方法，即结合实时运行的SLAM系统得出的粗糙scene graph及点云环境地图结果，为三维重建提供先验信息，结合三维重建全局大尺度地图及运动轨迹的优化目标，使用鲁棒性更高的特征描述算子，进行全局BA优化，得到更加精确的全局稀疏地图和运动轨迹。

通过本章基于改进后方法获取到的堆体点云模型，可以提供给基于纯视觉方法测量堆体体积。因此，本章需要获取到高精度的堆体点云结果，以减小后续体积测量误差。