**第2章 面向堆体测量的无人机自主定位研究**

2.1 引言

基于SLAM方法的即时建图和定位方法目前已经发展到的相当成熟，尤其是在移动设备定位领域，基于视觉SLAM方法进行定位有着积极的应用。对于实际落地的场景，该系统输出的相机位姿和构建的地图必须具备真实尺度才能够进行导航与定位，此外依靠视觉方法构建的地图，其坐标系往往取决于初始化成功后的第一帧所建立的坐标系，难以与真实世界坐标系有所关联，这些问题的存在都限制着视觉SLAM在实际场景中的存在。

对于单目视觉SLAM算法，设备简易，处理数据较少，满足实时性的要求，但存在无法获取真实世界坐标系下的相机位姿以及无法获取地图实际尺度的问题；对于双目视觉的SLAM算法，可以解决地图尺度的问题，但面临成本较高，相机标定难度大且精度不高，鲁棒性较差等问题；对于融合视觉和IMU传感器的SLAM系统，可以获取真实尺度，以及得到真实世界坐标系下的相机和地图位姿，但该系统对IMU的精度要求较高，存在引入IMU后容易产生累计误差，难以初始化成功等问题。

针对上述问题，本章提出一种面向堆体场景进行测量功能的无人机自主定位方法，该方法将融合二维码视觉标签，在传统SLAM功能的基础上，可以得到相机和地图的真实尺度，并且根据一定的坐标转换，实时获取真实世界坐标系下的相机位姿和地图。在真实的堆体场景中，布置的二维码和普通的自然关键点相比，更加容易捕获，此外，二维码的布置固定，在重定位的流程中，往往会有更好的效果。该系统能够具备以下优良特性：

能够对地图进行保存，复用和更新，通过不断完善的先验地图提高无人机定位的精确度；

引入二维码视觉标签后，通过二维码的先验尺度在仅用单目相机的情况下估计出带有真实尺度的SLAM输出；

结合二维码中坐标系和真实世界的坐标系的转化，可以获取无人机在堆体场景中的实际位姿。

2.2 二维码识别与使用方法研究

通过在堆体场景中添加二维码视觉标签，可以使得视觉SLAM算法在实际场景中得以应用，二维码和一般的自然关键点相比较，具备比周围环境更易检出的显著特点，如图~\ref{fig:2VSLAM\_MarkerandGradient}所示；且二维码本身是一个四边形的区域，可以凭借该特点估计出相机包含绝对尺度的位姿；并且每一个二维码通过解码都可以获得一个独一无二的ID序号，在SLAM重定位的过程中，可以避免相似区域的的误检情况，在包含二维码的视觉SLAM中，希望尽可能多的检出场景中的二维码视觉标签，随后再通过编码公式对误检值进行剔除。本节将验证二维码得到检测和实现过程，以及如何通过二维码解算相机位姿。

\begin{figure}[H]

\centering%

\subcaptionbox{二维码原始图\label{fig:2VSLAM\_Marker}}{%

\includegraphics[height=6cm]{2VSLAM\_Marker.png}}\hspace{4em}%

\subcaptionbox{二维码梯度图\label{fig:2VSLAM\_Marker\_Gradient}}{%

\includegraphics[height=6cm]{2VSLAM\_Marker\_Gradient.png}}

\caption{二维码原始图和梯度图}

\label{fig:2VSLAM\_MarkerandGradient}

\end{figure}

2.2.1 二维码检测与识别过程

二维码的检测和识别过程主要包括检测出二维码4个角点在图像中的位置，以及检出二维码对应ID序号，在检测4个角点的位置时，需要检测图像中的线段和构成二维码的四边形。

在线段检测阶段，首先会计算图片中每一个像素梯度强度的大小和方向，随后对计算得到的梯度进行聚类，对所有满足聚类条件的像素点进行合并，得到一组连续点，即检测出线段。

检测完线段后，进一步需要检测构成二维码边缘的四边形，针对上一步中获取到的所有线段，对线段进行分组，若满足，上一线段的末端点和下一线段的起始点之间的距离小于某一阈值，即可首尾按照逆时针进行连接，若所连接的线段数量达到4时，即认定生成的闭环可能为一个二维码的边缘四边形。

在检测二维码ID值之前，会先设定好一个包含所有二维码的字典，字典的大小即包含所有二维码的数量。检测出构成二维码的四边形后，需要对图像进行透视变化规范图像，随后通过设定阈值分离出二维码上的黑色位和白色位，通过位数情况判断出该二维码在字典中对应的特定ID值，识别结果如图~\ref{fig:2VSLAM\_Marker\_detection}所示。通过这样的方式检测和识别二维码具备非常好的鲁棒性，并且可以对错误值进行筛除。

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

\centering

\includegraphics[height=6cm]{Aruco\_detection.png}

\caption{二维码识别结果示意图}

\label{fig:Aruco\_detection}

\end{figure}

2.2.2利用二维码估计相机外参

对于单帧图像，在提取完二维码的四边形轮廓,4个角点以及检测出对应的唯一ID值后，接下来可以估计相机位姿，包括平移向量和旋转向量，相机的位姿主要是通过PnP方法来求解。

以二维码坐标系作为世界坐标系，二维码边长为S，则四个角点的坐标分别为A（-S/2，S/2，0），B（S/2，S/2，0），C（S/2，-S/2，0），D（-S/2，-S/2，0），分别在图像中对应的像素点为a($u\_a$,$v\_a$)，b($u\_b$,$v\_b$)，c($u\_c$,$v\_c$)，d($u\_d$,$v\_d$)。因为相机的内参K提前标定，则三维空间中的点和像素坐标中的点之间的转换关系可以表示为：

\begin{equation}

\begin{split}

Z{

\left[ \begin{array}{ccc}

a\\1

\end{array}

\right ]}=Z{

\left[ \begin{array}{ccc}

u\_a\\v\_a\\1

\end{array} \right ]}=K{

\left[ \begin{array}{ccc}

R&t

\end{array} \right ]}{

\left[ \begin{array}{ccc}

A\\1

\end{array} \right ]} \\

={

\left[ \begin{array}{ccc}

f\_x & 0 & u\_0\\0 & f\_y &v\_0 \\0 & 0 & 1

\end{array} \right ]}{

\left[ \begin{array}{cccc}

r\_{11}&r\_{12}&r\_{13}&t\_1\\r\_{21}&r\_{7722}&r\_{23}&t\_2\\r\_{31}&r\_{32}&r\_{33}&t\_3

\end{array} \right ]}{

\left[ \begin{array}{ccc}

-s/2\\s/2\\0\\1

\end{array} \right ]}

\end{split}

\label{equ:mark2pose}

\end{equation}

通过公式~\ref{equ:mark2pose}，可以通过二维码的角点信息求解出相机的旋转矩阵和平移矩阵。

2.3 联合二维码的SLAM系统描述

2.3.1引入二维码对SLAM优化研究

SLAM是一个在导航的过程中同时进行建图的工作，在使用单目相机时，即使目前表现较好的ORB-SLAM或者LDSO等算法也很难达到预期想要的效果。第一个问题：所构成的地图的尺度未知，这样就造成无法在实际的导航任务中发挥作用；其次，在纯旋转的移动过程中，会导致算法失效；第三个就是单目视觉SLAM往往需要场景中存在比较丰富的材质才能便于跟踪；最后当场景中的视点发生变化，或者场景重复度较高时，对于纯视觉SLAM的重定位，表现效果会很受到很大的限制。

除了用自然点参与视觉SLAM外，还可以使用人为设定的二维码来估计相机位姿，例如SPM-SLAM方法等，该方式可以较好的解决上述问题。因为二维码的放置不需要按照特定的规则，对于堆体场景，可以直接在堆体周围放置二维码标签，这样做就可以的得到一个由二维码构成的地图。但是这样做就需要在场景中布置足够多的二维码视觉标签，那么在大尺度的堆体场景中，这样做就会提高实验的难度，因为在相机移动的过程中，要保证每帧图像中至少有两张二维码出现。

基于上述两种情况，本文提出了一种融合自然特征点和人工放置的二维码的SLAM方法来解决各自的问题和限制。对于实际堆体场景中，结合了自然点和二维码的SLAM方法可以只检测自然点，或者只检测二维码来进行位姿估计。因此对于大多数场景，都可以有更好的鲁棒性，此外，每一个二维码都有一个单独准确得到的ID值，在高度重复的堆体环境中，也能保证匹配的正确性，最后对于大尺度的堆体场景，可以结合自然点进行跟踪，二维码进行重定位来保证SLAM系统的长期稳定。

如图~\ref{fig:map}所示，对自然场景进行建图，图~\ref{fig:map}为自然区域的原图，在该区域中，部分区域纹理丰富，其余部分则为贫纹理的白墙面，则在白墙面上人为布置二维码标志，图~\ref{fig:map}为根据该区域重建出的3D地图，可以发现，在丰富纹理的区域，自然点和人为布置的二维码都可以检出，但在贫纹理区域则，则基本只有二维码被检出，因此在贫纹理的区域就可以依靠二维码发挥作用。

\begin{figure}[htbp]

\centering

\subcaptionbox{自然场景}{\label{fig:cross}

\includegraphics[width=12cm]{cross.png}\hskip2cm}

\vskip0.5cm

\subcaptionbox{根据自然场景建图}{\label{fig:cross\_map}

\includegraphics[width=12cm]{cross\_map.png}\hskip2cm}

\caption{包含二维码的构建地图示意图}\label{fig:map}

\end{figure}

2.3.2包含二维码的地图描述

在SLAM运行的过程中，可以生成一套地图系统，对于同一场景，该地图可以作为先验复用于后续的建图和定位，通过这样的方式可以添加对系统的约束，提高待估计量的精度。传统SLAM系统组成地图的元素主要包括关键点和关键帧，通过对关键点的描述符进行匹配，可以较好的的估计出相机的位姿。但关键点描述符的计算和匹配过程一般都比较耗时，而且对于堆体场景中，极易出现重复，容易出现出现匹配错误的情况，考虑到这一情况，本文在此基础上，又添加二维码信息作为构成地图的元素，来进一步优化地图以获取更加准确得到相机位姿和地图信息。

本文中地图的构成包括以下3个集合：关键点集合\mathbf{p}，关键帧集合f和二维码m集合，每个集合之间的数据元素耦合关联。其中关键点集合

\begin{equation} \mathbf{p}=\{\mathbf{x}, \mathbf{v}, \hat{\mathbf{d}}\}\end{equation}

每个元素代表三维空间中的一个点，该点的描述包括在地图坐标系中的三维坐标\{\mathbf{x}，观测方向\mathbf{v}以及该点的描述符信息\hat{\mathbf{d}}，考虑到SLAM系统的实时性，选择BRIEF描述符来加快匹配过程。关键帧集合

\begin{equation}\mathbf{f}=\{t, \mathbf{T}, \delta\}\end{equation}

其中每一个关键帧包含一个外参矩阵矩阵\mathbf{T},该外参矩阵是全局参考坐标系到相机参考坐标系的转化，\delta\是相机的内参矩阵，该参数包括相机焦距，光学中心以及畸变参数，这些参数提前标定。二维码集合

\begin{equation}m =\mathbf{m}=\left\{s, M, \mathbf{x}^{1}, \mathbf{x}^{2}, \mathbf{x}^{3}, \mathbf{x}^{4}\right\}\end{equation}

其中的每一个二维码包括边长s（需要保证场景中的所有二维码的为正方形且所有二维码的尺度完全一致），二维码由其自身坐标系到全局坐标系的转换矩阵M，$\mathbf{x}^{1}$,$\mathbf{x}^{2}$,$\mathbf{x}^{3}$,$\mathbf{x}^{4}$分别代表二维码四个角点在其自身坐标系下的坐标。通过对包含二维码的自然场景进行离线建图，可以得到如图~\ref{fig:Ucoslam\_map}所示的地图，其中正方形代表检出的二维码，蓝色矩形代表检出关键帧，红色点代表检出关键点。

上述地图元素之间还具备一定耦合关系：\\

1. 对于任意一个关键点，除了自身的属性外，还包括被观测帧序列，即所有可以观测到该特征点的关键帧的集合，以及在这些帧中出现的像素坐标位置
2. 对于任意一个关键帧，还包与其关联的其他关键帧序列，以及在该关键帧中所有观测到的关键点和坐标。

通过这些约束可以使得估计结果有更好的鲁棒性。

2.3.3联合二维码的SLAM系统描述

本文所提出的基于二维码的SLAM系统运行框图如图~\ref{fig:VSLAM \_pipeline}所示，与一般SLAM系统相比较，本文主要提出了添加二维码视觉标签的SLAM系统。该系统一直对地图进行维护，有新的信息加入时，则会对更新地图。在对堆体场景运行SLAM时，该地图为空，所以需要对其初始化。

\begin{figure}[H] %

\centering

\includegraphics[height=6.7cm,width=14cm]{SLAM\_ucoslam\_pipeline.png}

\caption{基于二维码的SLAM流程图}

\label{fig:SLAM\_ucoslam\_pipeline}

\end{figure}

\textbf{地图初始化：}在该SLAM中对地图的初始化可以获取相对尺度信息和参考坐标系的信息，可以依靠图像序列中的关键帧或者二维码来完成，在一般情况下初始化的流程为:首先选择第一和第二帧（$f\_0$，$f\_1$）运行两套初始化方法，如果任意一套方法初始化成功，那么就直接进入跟踪环节；如果两套方案都初始化成功，则优先考虑使用二维码的初始化方式；如果都没有成功，那么则固定第一帧，换取第三帧重复上述过程（$f\_2$，$f\_3$...）。

对于基于二维码的初始化方法，帧之间需要足够的视差，随后得到帧和二维码在地图中的位姿信息，当在某一帧中检测到多个二维码标志时，则可以通过选择在该帧中最小化投影误差最小的二维码来估计初始化信息。使用二维码来进行初始化的一个优点就是，地图中的参数都具备正确的尺度，因此就可以直接用来为无人机的定位提供信息。

\textbf{跟踪模式：}如果在某一帧中估计出相机的位姿，则SLAM系统就会以前一帧作为起始点来估计当前帧的相机位姿，相机的位姿估计是将当前帧中观测到的一组地图三维点和二维码角点进行重投影误差最小化的过程，因此在该过程中主要包括两个步骤：寻找地图的对应关系和通过最小化投影误差获取相机位姿。

在寻找地图点对应关系环节，首先，查找在前一帧中观察到的地图点与参考关键帧之间的匹配，由于该帧很可能再次出现在当前帧中，通过这些匹配可以解算出当前相机的初始估计；然后，通过将地图点投射到参考关键帧的相邻帧中来找到额外的匹配。对于每个地图点，首先需要检查其视角与相机方向之间的夹角是否小于阈值，若大于则丢弃该点，然后，检查相机与地图点之间的欧氏距离是否在关键点的尺度不变性区域内，随后，计算其投影，最后，将地图点的描述符与图像中搜索半径内的关键点进行比较。选择搜索半径中距离最小的两个关键点，且仅当描述符之间的距离比大于0.8,以及最小的描述符距离低于最低阈值$\tau\_d$, 则与匹配成功。一旦所有的匹配都被计算出来，重复的匹配就会被移除。最终，关键点结合检测到的二维码标记获得最终的精确位姿。

对于相机的位姿估计，以

\begin{equation}

\Upsilon\_{p}^{f}=\{(\mathbf{p}, \mathbf{f}, \mathbf{g})\}

\end{equation}

和

\begin{equation}

\Upsilon\_{p}^{\mathrm{f}}=\left\{\left(\mathbf{m}, \mathbf{f}, \mathbf{c}\_{\mathbf{m}, \mathbf{f}}\right)\right\}

\end{equation}

表示通过上述步骤得到的$\mathrm{f}$帧中地图点和二维码标记的观测值，对于该帧的位姿$\mathbf{f}\_{\mathrm{T}}$可以通过最小化重投影误差来估计：

\begin{equation}

\mathbf{f}\_{\mathrm{T}}=\underset{\mathrm{T}}{\operatorname{argmin}}\left(\mathbf{w}\_{\mathbf{p}}^{\mathrm{f}} H\left(\Upsilon\_{p}^{\mathrm{f}}, \mathrm{T}\right)+\mathbf{w}\_{\mathbf{m}}^{\mathrm{f}} H\left(\Upsilon\_{m}^{\mathrm{f}}, \mathrm{T}\right)\right)

\label{equ:weight}

\end{equation}

其中

\begin{equation}

H\left(\Upsilon\_{m}^{\mathrm{f}}, \mathrm{T}\right)=\sum\_{\left(\mathrm{m}, \mathrm{f}, \mathrm{c}\_{\mathrm{m}, \mathrm{f}}\right) \in \Upsilon\_{m}^{\mathrm{f}}} \sum\_{i=1}^{4}\left\|e\left(\mathrm{T} \cdot \mathbf{m}\_{\mathrm{M}}, \mathbf{m}\_{\mathrm{x}^{i}}, \mathbf{f}\_{\delta}, \mathbf{c}\_{\mathrm{m}, \mathrm{f}}^{i}\right)\right\|\_{2}^{2}

\end{equation}

是当前帧观测到的二维码标记的重投影误差之和，其中

\begin{equation}

H\left(\Upsilon\_{p}^{\mathrm{f}}, \mathrm{T}\right)=\sum\_{(\mathbf{p}, \mathbf{f}, \mathbf{g}) \in \Upsilon\_{p}^{\epsilon}} h\_{\alpha}\left(e\left(\mathrm{T}, \mathbf{p}\_{\mathbf{x}}, \mathbf{f}\_{\delta}, \mathbf{g}\_{u}\right) \Omega\_{\mathbf{g}} e\left(\mathrm{T}, \mathbf{p}\_{\mathbf{x}}, \mathbf{f}\_{\delta}, \mathbf{g}\_{u}\right)^{\top}\right)

\end{equation}

是当前帧观测到的地图点的重投影误差之和。对于Huber损失函数

\begin{equation}

h\_{\alpha}(a)=\left\{\begin{array}{ll}{\frac{1}{2} a^{2}} & {\text { for }|a| \leq \alpha} \\ {\alpha\left(|a|-\frac{1}{2} \alpha\right)} & {\text { otherwise }}\end{array}\right.

\end{equation}

是为了降低优化过程中异常值的相对权重。

对于公式~\ref{equ:weight},二维码视觉标签和地图点的权重分别由

$\mathbf{w}\_{\mathbf{m}}^{\mathrm{f}}$

和$\mathbf{w}\_{\mathbf{p}}^{\mathrm{f}}$决定，因为同一帧中地图点的数量远远多于二维码的个数，所以必须平衡两者之间的权重，

定义

\begin{equation}

\mathbf{w}\_{\mathbf{p}}^{\mathbf{f}}=1-\mathbf{w}\_{\mathbf{m}}^{\mathbf{f}}

\end{equation}

和

\begin{equation}

\mathbf{w}\_{\mathbf{m}}^{\mathbf{f}}=\frac{1}{2} \min \left(1, \frac{\mathbf{n}\_{\mathbf{f}}}{\tau\_{m}}\right)

\end{equation}

其中$\mathbf{n}\_{\mathbf{f}}$代表追踪过程中该帧有效的二维码标记的个数，${\tau\_{m}}$代表数量阈值，

$\mathbf{w}\_{\mathbf{m}}^{\mathbf{f}}$的值将介于$\left[0, \frac{1}{2}\right]$之间，假设在追踪的过程中，没有检测到有效的二维码，则$\mathbf{n}\_{\mathbf{f}}$的值将为0，即该帧的位姿的结果将全部由地图点来估计。

\textbf{插入关键帧：}关键帧只有在向系统添加新信息时才会被添加到地图中，以便实现平稳可靠的跟踪。因此，必须设定一些规则来确定是否需要将当前帧添加至关键帧的集合。本文提出以下关键帧的提取规则:\\

1.如果当前帧至少有一个新的二维码标记出现(该标记没有出现在原本的地图中)，则将该二维码和该帧添加到地图中。二维码是第一次在该帧中被发现时添加的，因此也不能在帧中明确地估计该二维码的姿态。在这种情况下，标记的位姿被设置为无效，当观察标记的其他关键帧被添加到地图中时，再对其进行估计。\\

2. 如果该帧包含至少一个二维码标记，且该二维码标记在地图中具有无效位姿，如规则1所描述，则该二维码的位姿可以在该帧中进行估计，则添加该帧并更新该二维码在地图中的位姿。\\

3.如果帧包含至少一个二维码标志,并且距当前帧最近的关键帧的距离大于一个阈值$\tau\_b$,该帧也会被添加进关键帧。\\

4 如果当前帧中匹配的映射点数量低于在参考关键帧中检测到的映射点总数的百分比$\tau\_k$，该帧也会被添加进关键帧。

前三条规则的目的是考虑到当前关键帧中的可见二维码视觉标签来检查是否添加关键帧，由于这些规则而没有被认为是关键帧的帧则通过最后一条规则来添加。

本文中的SLAM系统在添加地图点时会遵循一种强鲁棒的策略，当一个新的关键帧被添加到地图时，系统有机会添加新的地图点，以及增加对现有地图点的支持。对于每个关键点，搜索参考关键帧的相邻关键帧之间可能的对应，由于关键帧的姿态是已知的，所以采用了外极限制来减少误检。一旦在地图上添加了一个点，将以一下原则验证该点是否会被加入地图基本思想是，该点必须至少在接下来的三分之二帧中可见，直到再向地图添加两个关键帧。

\textbf{地图优化：}在添加新的关键帧或检测到闭环时，需要更新地图，以便加入新的信息，并剔除不正确的地图点。需要优化的元素包括关键帧位姿$k\_T$、地图点位置$p\_x$和二维码位姿$m\_M$。全局优化是一个缓慢耗时的过程，在SLAM中一般会启用多线程的方式来管理地图。

\textbf{筛选关键帧：}与地图点的保存策略相似，关键帧剔除策略避免了关键帧的无限增长。但是，相似处理关键帧的同时必须同时处理关键点和二维码。其主要流程是，每个二维码和地图点都应至少被三个关键帧所观测到，以便在进行优化过程时实现良好的三角化。对于每一个二维码标记$\mathbf{m}$，从关键帧集合中找到可以观测的关键帧，并筛选出三个和该关键帧距离最近的关键帧，这些关键帧不会被筛除。对于剩余的关键帧，在不显著影响观察到的地图点的情况下会筛选掉，即某个关键帧中的地图点如果存在$\tau\_c$的比例以上会被至少三个其他关键帧观测到，那么该帧会被被认为是冗余的，而使用其他三个关键帧代替。

\textbf{回环检测和校准：}在视觉SLAM中，对于相机位姿的解算都是由上一帧来计算的，那么这样就会造成误差的一直传递，产生累积误差，回环检测的提出就是通过建立当前帧和历史帧之间的关系以减少累积误差。词袋模型的提出可以快速简单地完成回环检测，词袋模型使得每一帧都可以用单词来描述，表达为：

\begin{equation}

F=1 \cdot \omega\_{1}+0 \cdot \omega\_{2}+0 \cdot \omega\_{3}+\ldots+1 \cdot \omega\_{n-1}+0 \cdot \omega\_{n}

\end{equation}

每一帧$q$都可以由单词$w$和权重$\eta$构成：

\begin{equation}

A=\left\{\left(w\_{1}, \eta\_{1}\right),\left(w\_{2}, \eta\_{2}\right), \ldots,\left(w\_{N}, \eta\_{N}\right)\right\} \triangleq v\_{A}

\end{equation}

可以通过计算$q$,$d$两帧之间字典的相似度判断是否构成回环:

\begin{equation}

\begin{aligned} s\left(v\_{q}, v\_{d}\right) &=1-\frac{1}{2}\left|\frac{v\_{q}}{\left|v\_{q}\right|}-\frac{v\_{d}}{\left|v\_{d}\right|}\right| \\ &=\frac{1}{2} \sum\_{i | q\_{i} \neq 0, d\_{i} \neq 0}\left(\left|v\_{q i}\right|+\left|v\_{d i}\right|-\left|v\_{q i}-v\_{d i}\right|\right) \end{aligned}

\end{equation}

以上是对关键帧的回环处理，对于二维码的回环，必须立即检出由二维码标记引起的闭环，如果在跟踪之前不纠正回环漂移，会导致相机的跟踪问题，因为未纠正的帧位姿可能与二维码所计算出来的位姿差异较大。例如在某一个场景中，系统从一个二维码开始初始化，相机在场景中移动并回到最初的起始位置，在这个过程中，会不断地叠加漂移误差，当相机第二次到达初始位置并再次观察二维码时，估计的相机姿态与实际会较大，当使用二维码标记或关键点检测到循环闭合时，必须纠正漂移。

\textbf{重定位：}如果在跟踪的过程中失败，则必须进行重定位，重定位也是首先考虑使用二维码来完成，如果在某一帧中明确检测到二维码标记，则利用该二维码来重新计算位姿，随后，通过查找地图对应关系和重新估计相机姿态来优化位姿。如果当前帧位姿不能从二维码标记中准确地估计出来，就使用关键帧结合数据库进行查询，查询与当前帧相似的关键帧，并分析最佳关键帧。在回环检测出中，将当前帧的关键帧与所选关键帧进行匹配，并采用RANSACPnP法估计相机的位姿T,如果内置点的数量足够高，则认为重新定位成功，系统重新进入跟踪模式。

2.4 面向堆体无人机定位系统设计

2.4.1 无人机相机坐标变换数学描述

人机上携带的相机可以看成三维空间中的刚体（保证了同一个向量在各个坐标系下的长度和夹角都不会发生变化），对于相机的平移和旋转，平移可以利用向量来表示，旋转则利用旋转矩阵，旋转向量，欧拉角，四元数等方式来描述，本文也将主要说明旋转过程的数学描述。

\textbf{旋转矩阵:}对于任意向量{~p~}，在两个坐标系的描述为：

\begin{equation}

\left[\mathbf{e}\_{1}, \mathbf{e}\_{2}, \mathbf{e}\_{3}\right]\left[\begin{array}{l}{a\_{1}} \\ {a\_{2}} \\ {a\_{3}}\end{array}\right]=\left[\mathbf{e}\_{1}^{\prime}, \mathbf{e}\_{2}^{\prime}, \mathbf{e}\_{3}^{\prime}\right]\left[\begin{array}{c}{a\_{1}^{\prime}} \\ {a\_{2}^{\prime}} \\ {a\_{3}^{\prime}}\end{array}\right]

\end{equation}

等式两边同时左乘$\left[\begin{array}{ccc}{\mathbf{e}\_{1}^{T},} & {\mathbf{e}\_{2}^{T},} & {\mathbf{e}\_{3}^{T}}\end{array}\right]^{T}$可得

\begin{equation}

\left[\begin{array}{l}{a\_{1}} \\ {a\_{2}} \\ {a\_{3}}\end{array}\right]=\left[\begin{array}{lll}{e\_{1}^{T} e\_{1}^{\prime}} & {e\_{1}^{T} e\_{2}^{\prime}} & {e\_{1}^{T} e\_{3}^{\prime}} \\ {e\_{2}^{T} e\_{1}^{\prime}} & {e\_{2}^{T} e\_{2}^{\prime}} & {e\_{2}^{T} e\_{3}^{\prime}} \\ {e\_{3}^{T} e\_{1}^{\prime}} & {e\_{3}^{T} e\_{2}^{\prime}} & {e\_{3}^{T} e\_{3}^{\prime}}\end{array}\right]\left[\begin{array}{c}{a\_{1}^{\prime}} \\ {a\_{2}^{\prime}} \\ {a\_{3} ;}\end{array}\right]=\mathbf{R a}^{\prime}

\end{equation}

对于旋转矩阵R，可描述旋转过程，并且R可以定义为特殊正交群：

\begin{equation}

S O(n)=\left\{\mathbf{R} \in R^{n \times n} | \mathbf{R} \mathbf{R}^{T}=\mathbf{I}, \operatorname{det}(\mathbf{R})=1\right\}

\end{equation}

\textbf{旋转向量：}旋转矩阵在描述旋转时存在参数冗余和自身约束的问题，因此提出一个更加紧凑的数学描述$\theta\_{n}$，旋转向量方向为旋转轴{~n~}，

大小为旋转角$\theta$。旋转向量和旋转矩阵的转换可以用罗德里格斯公式表示：

\begin{equation}

\mathbf{R}=\cos \theta \mathbf{I}+(1-\cos \theta) \mathbf{n} \mathbf{n}^{T}+\sin \theta \mathbf{n}^{\wedge}

\end{equation}

反之则有

\begin{equation}

\operatorname{tr}(\mathbf{R})=\cos \theta \operatorname{tr}(\mathbf{I})+(1-\cos \theta) \operatorname{tr}\left(\mathbf{n} \mathbf{n}^{T}\right)+\sin \theta \operatorname{tr}\left(\mathbf{n}^{\wedge}\right)=1+2 \cos \theta

\end{equation}

\textbf{欧拉角：}将旋转运动分解成分别绕三个坐标轴的旋转（$[r, p, y]^{T}$）来表示的，三个旋转的总和即为总的旋转，表述最为直观。

\textbf{四元数：}由于欧拉角和旋转向量存在奇异性，且不存在不带奇异性的三维向量描述方式，因此提出四元数这种紧凑且没有奇异性的数学描述。四元数可以描述为：

\begin{equation}

\mathbf{q}=q\_{0}+q\_{1} i+q\_{2} j+q\_{3} k=\{s, \mathbf{v}\}

\end{equation}

可以用单位四元数来描述空间中的任意旋转，假设旋转的描述为$\theta\_{n}$，则：

\begin{equation}

\mathbf{q}=\left[\cos \frac{\theta}{2}, n\_{x} \sin \frac{\theta}{2}, n\_{y} \sin \frac{\theta}{2}, n\_{z} \sin \frac{\theta}{2}\right]^{T}

\end{equation}

旋转描述为R时，则：

\begin{equation}

\mathbf{R}=\left[\begin{array}{ccc}{1-2 q\_{2}^{2}-2 q\_{3}^{2}} & {2 q\_{1} q\_{2}-2 q\_{0} q\_{3}} & {2 q\_{1} q\_{3}+2 q\_{0} q\_{2}} \\ {2 q\_{1} q\_{2}+2 q\_{0} q\_{3}} & {1-2 q\_{1}^{2}-2 q\_{3}^{2}} & {2 q\_{2} q\_{3}-2 q\_{0} q\_{1}} \\ {2 q\_{1} q\_{3}-2 q\_{0} q\_{2}} & {2 q\_{2} q\_{3}+2 q\_{0} q\_{1}} & {1-2 q\_{1}^{2}-2 q\_{2}^{2}}\end{array}\right]

\end{equation}

反之则有：

\begin{equation}

q\_{0}=\frac{\sqrt{\operatorname{tr}(\mathbf{R})+1}}{2}, q\_{1}=\frac{m\_{23}-m\_{32}}{4 q\_{0}}, q\_{2}=\frac{m\_{31}-m\_{13}}{4 q\_{0}}, q 3=\frac{m\_{12}-m\_{21}}{4 q\_{0}}

\end{equation}

通过以上方式都可以描述刚性运动中的旋转过程R，结合物体的平移t，则变化后的坐标为：

\begin{equation}

\mathbf{a}^{\prime}=\mathbf{R} \mathbf{a}+\mathbf{t}

\end{equation}

在实际的使用中，由于以上等式非线性，可以引入齐次坐标T，可以将上式改成：

\begin{equation}

\left[\begin{array}{l}{\mathbf{a}^{\prime}} \\ {1}\end{array}\right]=\left[\begin{array}{ll}{\mathbf{R}} & {\mathbf{t}} \\ {\mathbf{0}^{T}} & {1}\end{array}\right]\left[\begin{array}{l}{\mathbf{a}^{\prime}} \\ {1}\end{array}\right]=\mathbf{T}\left[\begin{array}{l}{\mathbf{a}} \\ {1}\end{array}\right]

\end{equation}

对于变换矩阵T，可以定义为特殊欧氏群：

\begin{equation}

S E(3)=\left\{T=\left[\begin{array}{cc}{\mathbf{R}} & {\mathbf{t}} \\ {\mathbf{0}^{T}} & {1}\end{array}\right] \in R^{4 \times 4} | \mathbf{R} \in S O(3), t \in R^{3}\right\}

\end{equation}

对于针孔相机模型，如图~\ref{fig:2VSLAM\_pinehole}所示，（其中f为相机焦距）存在4个坐标系分别为：世界坐标系，相机坐标系，图像坐标系和像素坐标系，对于真实世界中的空间点$P\_w$（$X\_w$，$Y\_w$，$Z\_w$），其对应的相机坐标系坐标为$P\_c$（$C\_w$，$C\_w$，$C\_w$）,对应的图像坐标系为$P^{'}$（$X^{'}$,$Y^{'}$），对应的像素坐标系为p（u，v）。

\begin{figure}[H]

\centering

\includegraphics[height=6cm]{2VSLAM\_pinehole.png}

\caption{针孔相机模型}

\label{fig:2VSLAM\_pinehole}

\end{figure}

对于世界坐标系到相机坐标系之间的转化：

\begin{equation}

\left[\begin{array}{l}{X c} \\ {Y\_{C}} \\ {Z\_{C}}\end{array}\right]=R\left[\begin{array}{l}{X w} \\ {Y w} \\ {Z w}\end{array}\right]+t

\label{equ:world2cam}

\end{equation}

对于相机坐标系到图像坐标系之间的转化，可以利用如图~\ref{fig:2VSLAM\_similartri}所示的相似三角形来求解:

\begin{equation}

\begin{split}

\frac{Z c}{f}=\frac{X c}{X^{'}}=\frac{Y c}{Y^{'}}\\

\left\{\begin{array}{l}{X^{\prime}=f \frac{X c}{Z c}} \\ {Y^{\prime}=f \frac{Y c}{Z c}}\end{array}\right.

\end{split}

\label{equ:cam2photo}

\end{equation}

对于图像坐标系到像素坐标系的转化，如图~\ref{fig:2VSLAM\_pixeltrans}所示:

\begin{equation}

\begin{split}

\left\{\begin{array}{l}{u=\frac{X^{\prime}}{d\_{x}}+u\_{o}} \\ {v=\frac{Y^{\prime}}{d\_{y}}+v\_{o}}\end{array}\right.\\

\left[\begin{array}{l}{u} \\ {v} \\ {1}\end{array}\right]=\left[\begin{array}{ccc}{\frac{1}{d\_{x}}} & {0} & {u\_{o}} \\ {0} & {\frac{1}{d\_{x}}} & {v\_{o}} \\ {0} & {0} & {1}\end{array}\right]\left[\begin{array}{c}{X^{\prime}} \\ {Y^{\prime}} \\ {1}\end{array}\right]

\end{split}

\label{equ:photo2pixel}

\end{equation}

\begin{figure}[H]

\centering%

\subcaptionbox{相机坐标系到图像坐标系转换\label{fig:2VSLAM\_similartri}}{%

\includegraphics[height=4cm]{2VSLAM\_similartri.png}}\hspace{6em}%

\subcaptionbox{图像坐标系到像素坐标系转换\label{fig:2VSLAM\_pixeltrans}}{%

\includegraphics[height=4cm]{2VSLAM\_pixeltrans.png}}

\caption{坐标系转化示意图}

\label{fig:trans}

\end{figure}

其中$d\_x$,$d\_y$分别表示沿着x,y轴的实际物理尺寸，（$u\_o$,$v\_o$）表示光心对应到像素坐标系得到坐标.由公式~\ref{equ:cam2photo}和

~\ref{equ:photo2pixel}可以转化得到相机坐标系到像素坐标系的关系：

\begin{equation}

Z c\left[\begin{array}{l}{u} \\ {v} \\ {1}\end{array}\right]=\left[\begin{array}{ccc}{\frac{f}{d\_{x}}} & {0} & {u\_{0}} \\ {0} & {\frac{f}{d\_{y}}} & {v\_{0}} \\ {0} & {0} & {1}\end{array}\right]\left[\begin{array}{c}{X\_{c}} \\ {Y\_{c}} \\ {Z\_{c}}\end{array}\right]=\left[\begin{array}{ccc}{f\_{x}} & {0} & {u\_{0}} \\ {0} & {f\_{y}} & {v\_{0}} \\ {0} & {0} & {1}\end{array}\right]\left[\begin{array}{c}{X\_{c}} \\ {Y\_{c}} \\ {Z\_{c}}\end{array}\right]=K\left[\begin{array}{c}{X\_{c}} \\ {Y\_{c}} \\ {Z\_{c}}\end{array}\right]

\label{equ:cam2pixel}

\end{equation}

由公式~\ref{equ:world2cam}和~\ref{equ:cam2pixel}得到世界坐标系到像素坐标系的转化关系为:

\begin{equation}

Z c\left[\begin{array}{l}{u} \\ {v} \\ {1}\end{array}\right]=K(R P w+t)=K\left(R\left[\begin{array}{l}{X w} \\ {Y w} \\ {Z w}\end{array}\right]+t\right)

\end{equation}

其中K表示相机内参矩阵，R，t表示相机外参。

2.4.2 获取真实世界坐标系方法

在SLAM系统中所描述的定位信息都是基于地图坐标系或者相机坐标系，但是在实际的应用中，就需要将所有的定位信息全部转换至真实世界坐标系下，才能为无人机后续的移动提供控制方法。传统的SLAM方法，只能够得获取相机在地图坐标系下的位姿，本文结合二维码自带坐标系的特性，可以经过一定的转换获取相机在真实世界坐标系下的位姿，为后续控制提供真实世界下的位姿信息。

首先，声明所有可以得到的坐标信息，二维码在SLAM地图坐标系下的位姿$T\_{marker}^{c}$，相机在SLAM地图坐标系下得到位姿$T\_{camera}^{c}$，二维码在真实世界坐标系的位姿$T\_{marker}^{w}$。

经过坐标变化，可以的得到相机在二维码下的位姿

\begin{equation}

T\_{camera}^{m} =\begin{bmatrix}T\_{marker}^{c}\end{bmatrix}^{-1}T\_{camera}^{c}

\end{equation}

真实世界坐标系在二维码坐标系下的位姿

\begin{equation}

T\_{world}^{m} =\begin{bmatrix}T\_{marker}^{w}\end{bmatrix}^{-1}

\end{equation}

那么相机在真实世界坐标系下的位姿

\begin{equation}

\begin{array}{l}T\_{camera}^w=

\begin{bmatrix}T\_{world}^m\end{bmatrix}^{-1}T\_{camera}^m\\=\;T\_{marker}^w

\begin{bmatrix}T\_{marker}^c\end{bmatrix}^{-1}T\_{camera}^c\end{array}

\end{equation}

其中，二维码在真实世界坐标系的位姿可以按照实际场景进行设置，为了计算方便，直接设定第一个二维码的中心点为世界坐标系三维坐标的中心点，且x轴朝右，y轴朝上，z轴垂直于二维码平面向外，如图~\ref{fig:2VSLAM\_global}所示，那么第一个二维码的在世界坐标系下的坐标为（0,0,0），其余二维码则可以按照真实尺度获取。

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

\centering

\includegraphics[height=6cm]{Aruco\_global.png}

\caption{二维码布置示意图}

\label{fig:Aruco\_global}

\end{figure}

在SLAM实际运行前，应该保证二维码的实际布置坐标和设定的二维码在实际世界坐标系下的坐标一致，避免SLAM过程中位姿的抖动问题，先验地图可以以配置文件的形式记录，这样可以针对不同的场景和不同的二维码布置情况，直接修改配置文件即可。在SLAM运行过程中，应该尽可能保证每一帧都可以看到至少一个二维码，，对于观测不到二维码的图像帧，SLAM算法将利用特征点匹配进行定位，当观测的图像帧中有多个二维码时，那么就需要考虑到多个二维码的协同影响，本文提出以下几种处理形式：

1 提前选择某一个参考二维码，无论相机观测到哪一个二维码，都根据观测到的二维码和参考二维码之间的位姿关系，求解相机在世界坐标系下的位姿。

2.对于观测到的多个二维码标记，根据上一帧计算出来的位姿，可以计算出当前帧距离最近的一个二维码，以该二维码作为参考，计算出此时相机在世界坐标系下的位姿。

3. 对于观测到的多个二维码标记，分别结合各个二维码计算出此时相机在世界坐标系下的位姿，对于所有位姿，求取平均值即可。

以上三种方法都可以解算出相机在真实世界坐标系下的位姿，其中方法1的计算过程最为简略，并且全程只利用一个参考二维码可以避免在频繁更换参考二维码的过程中造成的位姿抖动问题，但是会存在线性放大相机位姿误差的情况，该方法难以对误差进行优化。方法2会应用到历史位姿信息，可以有效降低相机位姿估计的误差，但会频繁更换参考二维码，这样也会造成相机位姿估计过程的抖动问题。方法3.和方法2类似，在处理误差时会结合多个参考二维码的信息，使得估计位姿的系统鲁棒性提升。针对不同的场景，需要结合具体的方法，对于二维码远距离线性摆放的场景，应该避免使用方法1,优先考虑方法2和方法3；对于实时性要求较高的场景，则优先考虑方法1。

2.4.3 面向堆体的无人机自主定位设计

在根据以上章节，可以确定出在真实世界下相机的位姿结果，在具体的工程实践中，除了结合二维码的SLAM算法的研究，还需要考虑到先验地图的设定，数据的传输，相机的设定等问题，具体流程如图\ref{fig:2VSLAM\_getPose\_pipeline}所示。

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

\centering

\includegraphics[height=14cm]{getPose\_pipeline\_1.png}

\caption{获取真实位姿完整流程图}

\label{fig:getPose\_pipeline}

\end{figure}

\textbf{地图管理:}在无人估计位姿时，如果有先验地图的加入，则可以添加约束进一步优化位姿结果。在载入地图的过程中，如果该场景是第一次测量，没有先验的地图信息，则创建一个空的globalmap来存储地图信息，并将MapFlag字段设定为NotFirstMap；反之，则直接载入历史先验地图。在整个定位流程结束后，将更新后的地图保存在globalmap中,代码如~\ref{code: Map\_Process}所示。

\begin{lstlisting}[

language=C++,

numbers=left,

numberstyle=\footnotesize,

frame=single,

basicstyle=\small\tt,

escapeinside = '',

caption={地图管理~C++~实现},

label={code: Map\_Process}]

'//载入地图'

if (Map\_Flag == NotFirst\_Map)

{

globalmap->readFromFile(map\_path);

}

else

{

globalmap->readFromFile(map\_path);

Map\_Flag == NotFirst\_Map;

}

'//保存地图'

if (Map\_Flag == Save\_Map)

{globalmap->saveToFile(map\_path);}

\end{lstlisting}

\textbf{载入先验二维码地图：}首先准备先验地图配置文件，包括二维码的真实尺度，所有二维码的ID值，以及每一个二维码特定的id值和在真实世界坐标系下的位姿（R，t）。如代码~\ref{code:loadMarkermap}所示，需要依次遍历每个二维码的信息，最终将所有数据都存储在markermap中。

\begin{lstlisting}[

language=C++,

numbers=left,

numberstyle=\footnotesize,

frame=single,

basicstyle=\small\tt,

escapeinside = '',

caption={载入先验二维码地图~C++~实现},

label={code:loadMarkermap}]

'先验地图配置文件：'

%YAML:1.0

markersize: 0.73

markers\_id: [0,1]

id\_0: !!opencv-matrix

rows: 4

cols: 4

dt: d

data: [1.0, 0.0, 0.0, 0.0,

0.0, 1.0, 0.0, 0.0,

0.0, 0.0, 1.0, 0.0,

0.0, 0.0, 0.0, 1.0]

id\_1: !!opencv-matrix

rows: 4

cols: 4

dt: d

data: [1.0, 0.0, 0.0, 4.0,

0.0, 1.0, 0.0, 0.0,

0.0, 0.0, 1.0, 0.0,

0.0, 0.0, 0.0, 1.0]

'先验地图载入代码：'

void loadMarkermap(string path){

cv::FileStorage fs(path, cv::FileStorage::READ);

fs["markersize"] >> markersize;

fs["markers\_id"] >> markers\_id;

for (std::size\_t i =0;i<markers\_id.size();i++){

cv::Mat cvmarker ;

Eigen::Matrix<double,4,4> marker;

fs["id\_"+to\_string(markers\_id[i])] >> cvmarker;

cv2eigen(cvmarker,marker);

markermap[markers\_id[i]] = marker;

}

}

\end{lstlisting}

\textbf{设定SLAM参数：}对于结合二维码的SLAM系统，需要对系统中的参数进行设定，主要包括SLAM的运行模式（SLAM模式，重定位模式），特征点的提取方式（orb、brief、SIFT等），二维码的检测类别（ARUCO\\_MIP\\_25h7、ARUCO\\_MIP\\_36h12等），关键帧筛选阈值，线程的数量，是否复用地图等等，这些SLAM相关的参数都需要在运行前设定,具体内容如代码~\ref{code:setSlam}所示。

\begin{lstlisting}[

language=C++,

numbers=left,

numberstyle=\footnotesize,

frame=single,

basicstyle=\small\tt,

escapeinside = '',

caption={slam启动和参数设置~C++~实现},

label={code:setSlam}]

void setSlam(int argc){

UcoSlamParams.runSequential=false;

UcoSlamParams.detectMarkers=true;

UcoSlamParams.aruco\_markerSize=markersize;

UcoSlamParams.aruco\_Dictionary="ARUCO\_MIP\_36h12";

UcoSlamParams.nthreads\_feature\_detector =1;

}

\end{lstlisting}

\textbf{设定相机参数：}相机的选择对基于纯视觉的无人机定位任务也很重要，作为SLAM系统的输入，必须选择一台成像质量高，稳定性强的相机。在实际工程中选择PointGrey （CM3-U3-31S4M-CS）工业相机来接受视觉信息，外形如图~\ref{fig:2VSLAM\_pointgrey}所示，具体参数如表~\ref{tab:pointgrey}所示。

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

\centering

\includegraphics[height=6cm]{getPose\_pt.png}

\caption{PointGrey工业相机示意图}

\label{fig:getPose\_pt}

\end{figure}

\begin{table}[h]

\centering

\caption{pointgrey相机参数表}

\label{tab:pointgrey}

\begin{tabular}{C{1.6cm}C{2.4cm}C{2.6cm}C{6.4cm}}

\toprule

\textbf{参数} & \textbf{数值} & \textbf{参数} & \textbf{数值} \\

\midrule

Frame Rate & 30 FPS & Megapixels & 1.3 MP \\

Lens Mount & CS-mount & Readout Method & Global shutter \\

Sensor Format & 1/3" & Chroma & Color \\

Sensor Type & CCD & Part Number & CM3-U3-13S2C-CS \\

Pixel Size & 3.75 µm & Sensor Name & Sony ICX445 \\

Resolution & 1288 × 964 & Required Accessories & Lens, Cable, Host Adapter (USB 3.1 Gen 1), Tripod Mount Adapter (ACC-01-0003) \\

\bottomrule

\end{tabular}

\end{table}

相机在使用过程中需要设定的参数一般包括相机内参，以及曝光，快门时间和增益等，对于相机内参需要提前进行标定，后面三个参数需要根据实际场景调试后确定出最佳数值,代码如~\ref{code: setCamera}所示

\begin{lstlisting}[

language=C++,

numbers=left,

numberstyle=\footnotesize,

frame=single,

basicstyle=\small\tt,

escapeinside = '',

caption={相机参数设定~C++~实现},

label={code:setCamera}]

int SHUTTER=7 ; int BRIGHTNESS=0.5; float GAIN=20.0;

void setCamera(){

for (int frameNumber = 0; ;frameNumber++) {

char key = static\_cast<char>(cv::waitKey(1));

if ( key =='w' || key == 'W') {break;}

if ( key =='u' || key == 'U'){SHUTTER =SHUTTER+1;}

if ( key =='j' || key == 'J'){SHUTTER =SHUTTER-1;}

if ( key =='i' || key == 'I'){BRIGHTNESS =BRIGHTNESS+1;}

if ( key =='k' || key == 'K'){BRIGHTNESS =BRIGHTNESS-1;}

if ( key =='o' || key == 'O'){GAIN =GAIN+1;}

if ( key =='l' || key == 'L'){GAIN =GAIN-1;}

mypointgrey->pointgreyInit(SHUTTER,BRIGHTNESS,GAIN);

mypointgrey->getData();

}

}

\end{lstlisting}

\textbf{设定数据传输系统：}在工程实践中，一般都采用ROS（机器人操作系统）来对数据进行传输，其用于编写机器人软件程序的一种具备高灵活轻度的软件架构，可以使得数据在多个进程中进行传输，本文主要涉及SLAM进程和无人机飞控进程，这两个进程分别对同一个节点进行数据得到发布和订阅以完车数据传输,代码如~\ref{code: setROS}所示。

\begin{lstlisting}[

language=C++,

numbers=left,

numberstyle=\footnotesize,

frame=single,

basicstyle=\small\tt,

escapeinside = '',

caption={ROS初始化设定和数据传输~C++~实现},

label={code:setROS}]

void publishPose(ros::Publisher odom\_pub){

Eigen::Quaterniond q = Eigen::Quaterniond(t.block<3,3>(0,0));

nav\_msgs::Odometry odometry;

ros::Time current\_time= ros::Time::now();

odometry.header.stamp = current\_time;

odometry.header.frame\_id = "world";

odometry.pose.pose.position.x = t(0,3);

odometry.pose.pose.position.y = t(1,3);

odometry.pose.pose.position.z = t(2,3);

odometry.pose.pose.orientation.x = q.x();

odometry.pose.pose.orientation.y = q.y();

odometry.pose.pose.orientation.z = q.z();

odometry.pose.pose.orientation.w = q.w();

odom\_pub.publish(odometry);

ros::spinOnce();

}

for (int frameNumber = 0; keyPressed!=27;frameNumber++) {

...\dots

publishPose(odom\_pub);

}

ros::init(argc, argv, "talker");

ros::NodeHandle n;

ros::Publisher odom\_pub =

n.advertise<nav\_msgs::Odometry>("odom", 1000);

ros::Rate loop\_rate(10);

\end{lstlisting}

根据以上流程，可以针对堆体，在不适用GPS的条件下进行无人机的自主定位和控制，无人机所拍摄的堆体场景如图所示。二维码视觉标签位于图像左侧，连续放置，为无人机的自主定位提供世界坐标系得到真实位姿。

\begin{figure}[h]

\centering

\subcaptionbox{Frame1}{\label{fig:2VSLAM\_big1}

\includegraphics[width=6cm]{2VSLAM\_big1.jpg}\hskip2cm}

\subcaptionbox{Frame2}{\label{fig:2VSLAM\_big2}

\includegraphics[width=6cm]{2VSLAM\_big2.jpg}\hskip2cm}

\vskip0.5cm

\subcaptionbox{Frame3}{\label{fig:2VSLAM\_big3}

\includegraphics[width=6cm]{2VSLAM\_big3.jpg}\hskip2cm}

\subcaptionbox{Frame4}{\label{fig:2VSLAM\_big4}

\includegraphics[width=6cm]{2VSLAM\_big4.jpg}\hskip2cm}

\vskip0.5cm

\subcaptionbox{Frame5}{\label{fig:2VSLAM\_big5}

\includegraphics[width=6cm]{2VSLAM\_big5.jpg}\hskip2cm}

\subcaptionbox{Frame6}{\label{fig:2VSLAM\_big6}

\includegraphics[width=6cm]{2VSLAM\_big6.jpg}\hskip2cm}

\vskip0.5cm

\subcaptionbox{Frame7}{\label{fig:2VSLAM\_big7}

\includegraphics[width=6cm]{2VSLAM\_big7.jpg}\hskip2cm}

\subcaptionbox{Frame8}{\label{fig:2VSLAM\_big8}

\includegraphics[width=6cm]{2VSLAM\_big8.jpg}\hskip2cm}

\vskip0.5cm

\subcaptionbox{Frame9}{\label{fig:2VSLAM\_big9}

\includegraphics[width=6cm]{2VSLAM\_big9.jpg}\hskip2cm}

\subcaptionbox{Frame10}{\label{fig:2VSLAM\_big10}

\includegraphics[width=6cm]{2VSLAM\_big10.jpg}\hskip2cm}

\vskip0.5cm

\subcaptionbox{Frame11}{\label{fig:2VSLAM\_big11}

\includegraphics[width=6cm]{2VSLAM\_big11.jpg}\hskip2cm}

\subcaptionbox{Frame8}{\label{fig:2VSLAM\_big12}

\includegraphics[width=6cm]{2VSLAM\_big12.jpg}\hskip2cm}

2.5 本章小结

本章阐述了二维码的检测和识别原理，通过线条和角点的检测，判断区域中二维码的坐标位置以及每一个二维码对应的ID值，推导了通过二维码解算相机位姿的流程。

然后重点研究二维码对视觉SLAM带来的优化作用以及SLAM地图的描述情况，包括关键点，关键帧以及二维码的属性等。说明引入二维码对视觉SLAM各个环节的影响，包括地图初始化、跟踪情况、关键帧的插入筛选、地图优化、回环检测和重定位等。解决了针对堆体场景下，单目视觉SLAM在无人机自主定位工作中，无法确定尺度和不能确定坐标系的问题。

最后本文针对具体堆体场景，进行软件设计，提出了一种为无人机在封闭环境中依靠纯视觉进行自主定位的方案，包括地图管理，图像处理，无人机位姿信息传递等模块，模块之间相互协作，为无人机提供准确的位姿数据，供其后续进行循迹飞行。

在无人机飞行的过程中，会携带图像采集装置，在自主飞行的过程中会不断采集堆体的图像信息，为后续的堆体三维重建以及体积测量提供支持。