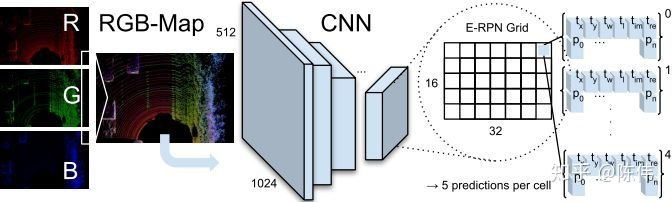
**Complex-YOLO**

**核心思想：**

本文基于YOLOv2版本的变异，通过把3D点云降维到2D鸟瞰图的方式，将图像检测的网络用于点云的目标检测中。出于降维后信息的损失考虑，本文采用点云多种特征综合起来填充输入通道，以达到目标信息的弥补。

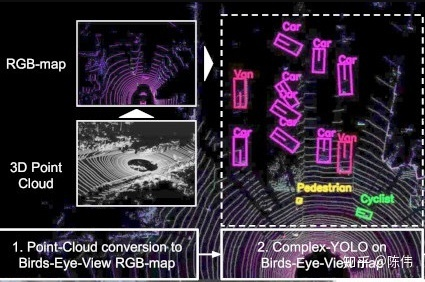


**框架结构：**

网络仍然以三通道作为输出，区别与图片中的RGB色系不同，这里先将三维点云进行栅格化，将点集分布到鸟瞰图空间的网格中，然后编码网格内点集的最大高度，最大强度，点云密度三种信息归一化后分别填充到R，G，B三个通道中形成RGB-Map，然后采用YOLOv2的Darknet19进行特征提取并回归出目标的相对中心点。

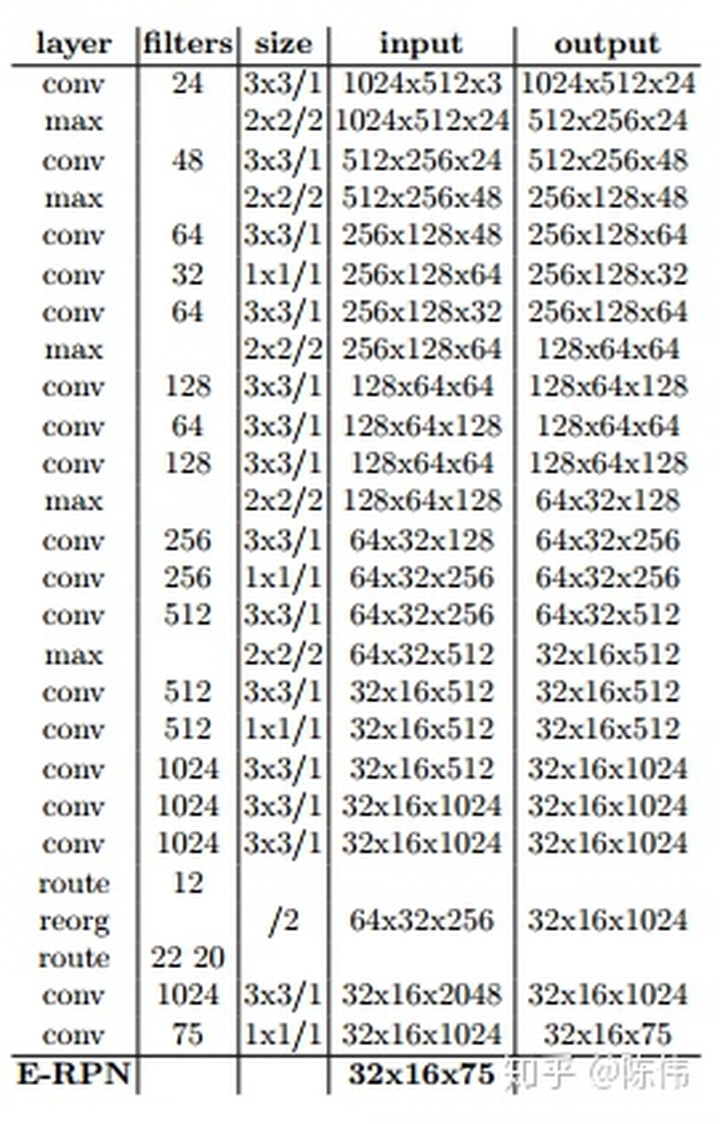
**实现细节：**

点云预处理：本文将单帧三维点云转换成一张俯视的三通道图片，帅选出传感器正前方ROI区域（80米x 40米）高度限定3米以内，并将点云栅格化到网格分辨率为8cm的二维网格图中：



三通道分别由**点云高度信息**、**点云强度信息**、**点云密度信息**编码所得

**网络结构：**本文的网络结构在YOLOv2版本的基础上使用E-RPN进行扩展。基本同Darket-19，只是在最后的输出层增加了两个复数角度的回归。



**锚点设计：**考虑在鸟瞰图视角下，同一类目标的长宽尺寸变化不大，但是目标存在方向信息，所以在设计锚点的时候，本文根据数据集内的外接框分布，定义了三种不同尺寸和两个角度方向：i）车辆尺寸（朝上）；i i）车辆尺寸（朝下）；i i i）自行车尺寸（朝上）；i v）自行车尺寸（朝下）；v）行人尺寸（朝左）。

**损失函数：**本文的损失函数在YOLO的基础上增加了欧拉角度回归损失，使用复数进行角度回归，总体上来说叠加的比较生硬，我们知道后面的版本将中心点和宽高放在一起保留bbox完整性然后计算损失，如果能把旋转角也和bbox其他属性整合在一个定位损失中会更好。

要点分析：

1、将图像检测网络YOLOv2应用到点云检测中，把三维点云转换成鸟瞰图的形式作为输入；

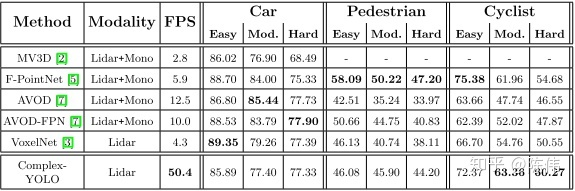
2、编码点云的高度，强度，密度信息到输入通道中；

3、在网络输出的位置信息，尺度信息，类别信息后增加了角度信息的输出；

4、采用复角的方式表征朝向角避免了单纯回归一个值所存在的奇异值问题（0°突变360°）；

**实验结果：**

从下表中可以明显的看出该方法的速度确实快，毕竟是基于单阶段的YOLO系列，但是这个速度应该是纯网络预测耗时，其点云转换成鸟瞰图部分还会消耗一部分时间：



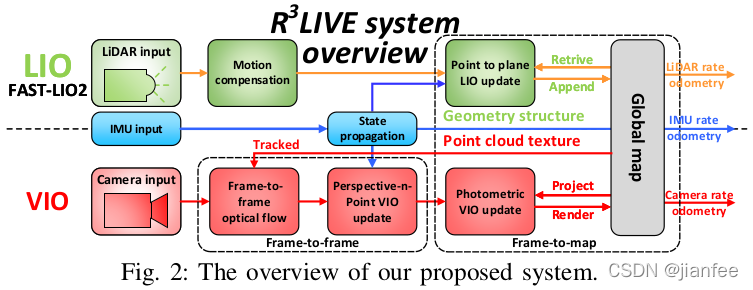
**思考与展望：**

1. 很多点云检测网络在其预处理部分需要消耗大量精力时间，本文也不例外，虽然网络的前向传播时效性比较好（或者升级到v5版本），但是对点云的预处理部分仍然拖累整体耗时；
2. 采用鸟瞰图形式的检测，由于点云近密远稀的特征，限制了其有效检测距离，所以本文只在40M以内的效果比较好；

**R3live**

[https://blog.csdn.net/handily\_1/article/details/122271243?ops\_request\_misc=%257B%2522request%255Fid%2522%253A%2522166796086316782425634405%2522%252C%2522scm%2522%253A%252220140713.130102334..%2522%257D&request\_id=166796086316782425634405&biz\_id=0&utm\_medium=distribute.pc\_search\_result.none-task-blog-2~all~top\_positive~default-1-122271243-null-null.142^v63^js\_top,201^v3^control\_1,213^v2^t3\_esquery\_v3&utm\_term=r3live&spm=1018.2226.3001.4187](https://blog.csdn.net/handily_1/article/details/122271243?ops_request_misc=%257B%2522request%255Fid%2522%253A%2522166796086316782425634405%2522%252C%2522scm%2522%253A%252220140713.130102334..%2522%257D&request_id=166796086316782425634405&biz_id=0&utm_medium=distribute.pc_search_result.none-task-blog-2~all~top_positive~default-1-122271243-null-null.142%5ev63%5ejs_top,201%5ev3%5econtrol_1,213%5ev2%5et3_esquery_v3&utm_term=r3live&spm=1018.2226.3001.4187)

R3LIVE(升级R2LIVE)：编译与运行



**整体框架：**

R3LIVE是一种新型的激光雷达-惯性-视觉传感器融合框架，它利用**激光雷达、惯性和视觉传感器**的测量来实现鲁棒性和准确性的状态估计。R3LIVE是在之前R2LIVE的基础上建立起来的，包含两个子系统:雷达 -惯性里程计(LIO)和视觉-惯性里程计(VIO)。LIO子系统(FAST-LIO)利用激光雷达和惯性传感器的测量，建立全局地图的几何结构(即三维点的位置)。VIO子系统利用视觉-惯性传感器的数据，呈现三维地图的纹理特征(即3D点的颜色)。

这个开源框架不仅生成了纹理贴合得不错的彩色点云，甚至还重建成了mesh带入到游戏中，使人身临其境。

1. **rviz分析**

Global Options坐标系：world

Grid：栅格

Axes：Fixed Frame的坐标系，不能添加相对于其他坐标系的Axes

LIO文件夹

pointcloud\_map： 订阅的是/cloud\_registered，这个就是处理后的全部点云了

current\_scan：当前扫描到的全部点，订阅的也是/cloud\_registered，Size是5，Alpha是1，Decay是0

feature\_map：特征点地图，订阅的是/Laser\_map

Path：r3live计算出来的轨迹，订阅的是/path

Odometry：/r2live/odometry 没用到

Odometry：建图的时候的位姿，订阅的是/aft\_mapped\_to\_init

Odometry：相机位姿，订阅的是/camera\_odom

Path：视觉计算出的轨迹，订阅的是/camera\_path

track\_pts：应该是视觉计算出的点云，跑代码的时候"/track\_pts"这个话题并没有数据输出

RGB\_map文件夹：这里面全是由不同的 /RGB\_map\_\*话题组成的一张RGB地图

Input image：rviz显示输入图像

Tracked points：rviz显示视觉追踪图像

1. **ros节点与话题**

/RGB\_map\_\*话题计较长，回头处理一下图片，其实r3live节点与话题和还是很清晰明了的，一共就/r3live\_LiDAR\_front\_end和/r3live\_mapping两个节点：

/r3live\_LiDAR\_front\_end节点算法来自于经典算法LOAM中的特征提取部分，其实也是R2live、fast-lio中都使用到的，他订阅/livox/lidar雷达话题信息，经过处理后发布点云/laser\_cloud，平面点/laser\_cloud\_flat和角点/laser\_cloud\_sharp信息。

/r3live\_mapping节点就订阅平面点/laser\_cloud\_flat、imu话题/livox/imu和相机（这里的图片是compressed格式）话题/camera/image\_color/compressed。发布的话题是前面rviz订阅的所有。

1. **launch文件分析**

以r3live\_bag.launch为例：

1. **<launch>**
2. <!-- Subscribed topics -->
3. **<param** name="LiDAR\_pointcloud\_topic" type="string" value= "/laser\_cloud\_flat" **/>**
4. **<param** name="IMU\_topic" type="string" value= "/livox/imu" **/>**
5. **<param** name="Image\_topic" type="string" value= "/camera/image\_color" **/>**
6. **<param** name="r3live\_common/map\_output\_dir" type="string" value="$(find r3live)/../r3live\_output" **/>**
7. **<rosparam** command="load" file="$(find r3live)/../config/r3live\_config\_r3live.yaml" **/>**
9. **<node** pkg="r3live" type="r3live\_LiDAR\_front\_end" name="r3live\_LiDAR\_front\_end"  output="screen"  required="true"**>**
10. **<remap** from="/livox/lidar" to="/livox/lidar" **/>**
11. **</node>**
13. **<node** pkg="r3live" type="r3live\_mapping" name="r3live\_mapping" output="screen" required="true" **/>**
15. **<arg** name="rviz" default="1" **/>**
16. **<group** if="$(arg rviz)"**>**
17. **<node** name="rvizvisualisation" pkg="rviz" type="rviz" output="log" args="-d $(find r3live)/../config/rviz/r3live\_rviz\_config.rviz" **/>**
18. **</group>**
19. **</launch>**

* 第一个参数是设置r3live处理的雷达话题，不是livox话题
* 定义imu、Image的topic和地图保存路径
* 加载配置文件r3live\_config\_r3live.yaml
* 启动r3live\_LiDAR\_front\_end节点和r3live\_mapping节点
* 启动rviz

**视觉-惯性里程计VIO部分**

<https://blog.csdn.net/handily_1/article/details/122377514>

（没看明白）

r3live的VIO子系统绘制了全局地图的纹理，通过最小化光度误差来估计系统的状态，即：

将全局地图中一定数量的点（即跟踪的点）投影到当前图像中，

然后最小化这些点的光度误差，在ESIKF框架内迭代估计系统状态。

为了提高效率，跟踪的地图点是稀疏的，这通常需要构建输入图像的金字塔。

然而，金字塔对平移或旋转都不是不变的，也需要被估计。

r3live框架中，利用单个地图点的颜色来计算光度误差,

因为颜色是地图点的固有属性，对相机的平移和旋转都是不变的。

**Avoiding dynamic small obstacles with onboard sensing and computating on aerial robots**

**摘要：**

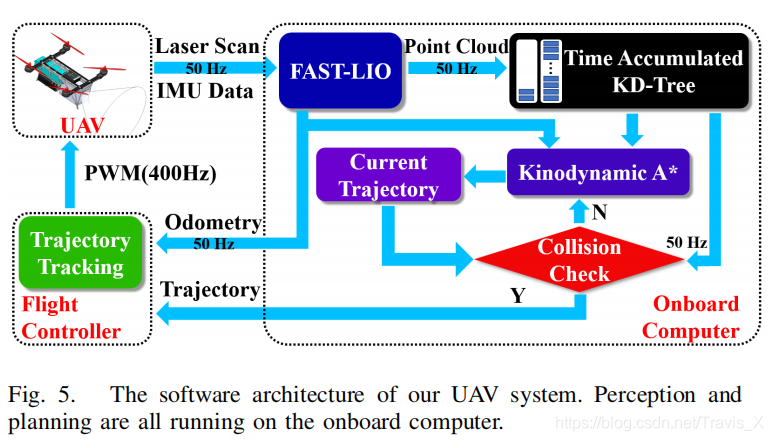
在实际应用中，自主四旋翼飞行器仍面临许多重大挑战，如检测和躲避非常小的或者是动态的障碍物（例如，树枝、电线等）。在本文中，我们提出了一个紧凑的，集成的、完全自主的四旋翼飞行器系统，它可以让飞行器在复杂环境中安全飞行的同时避免动态小障碍。

四旋翼平台搭配了机载电脑和三维激光雷达传感器来执行感知、位置估计、建图以及规划的任务。具体来说，机载电脑估计无人机的当前姿态维护局部地图（时间累积点云 KD-Trees），使用kinodynamic A\*算法搜索出一条到达目标点的安全轨迹。经过精心的优化，整个感知规划系统可以在机子上以 50Hz 的频率运行。通过各种室内外实验表明，该系统在复杂环境中以 2m/s 的速度飞行，同时可以避开动态的小障碍物（低至 20mm 直径的木棒）。

我注：一个真正的无人机避障系统是不需要区分室内还是室外避障的，因为执行同样的过程，在室内室外环境中都有可以实现避障。

**三、集成于机载计算机的感知与规划**

整个系统工作流程如图5所示。传统的自主无人机由三个功能部分组成：**导航、规划和控制**。 在我们的系统中，导航和规划模块都在机载计算机上运行，跟踪控制在机载飞行控制器 Pixhawk4-mini 上运行。 对于导航任务，我们使用 FAST-LIO 激光雷达 SLAM（Simultaneous Localization And Mapping）算法来计算无人机的位置并构建环境的点云图。



从激光雷达接收到**点云**和 **IMU** 数据的新扫描后，FAST-LIO 在**紧耦合迭代卡尔曼滤波器**中估计当前无人机状态。估计的状态又用于将点云的新扫描投影到世界坐标。然后将新扫描的点云添加到以两个时间累积的 K-D 树构建的局部地图中。更新后的局部地图会触发对当前跟踪轨迹的碰撞检查。如果存在发生碰撞的情况，则会进行重规划，并通过kinodynamic A\* 搜索生成新的安全轨迹来替换当前轨迹。否则，控制器对当前轨迹进行跟踪控制。轨迹跟踪控制器是在 Pixhawk4-Mini 上实现的级联 PID 控制器，我们事先适当调整了参数。此外，我们仔细优化了所有软件模块，以提高其实时性能。例如，我们在 FAST-LIO 中用增量式 KD 树结构替换 KD 树结构来增快它的计算速度。

**1. 时间累积KD树**

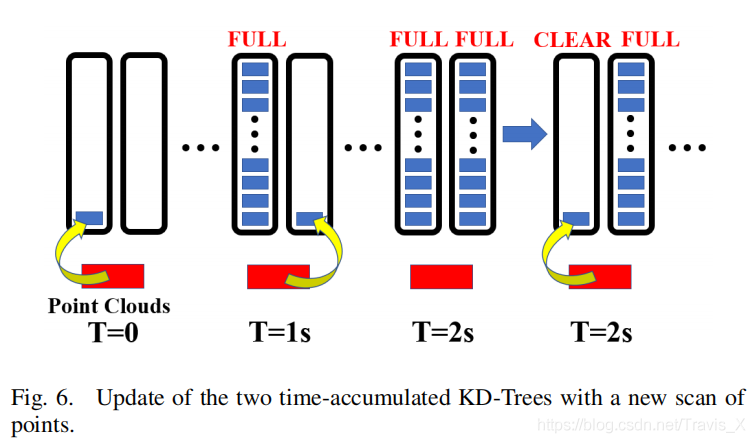
目前大多数的建图是依赖于占用网格图，为了滤除传感器带来的噪声，通常进行光线投射以估计沿点测量（例如，八叉树）的光线的每个网格的占用概率。 欧几里得有符号距离场 (ESDF) 地图建立在占用网格地图之上，并进一步保持每个网格与其最近的占用网格的有符号距离。这种距离提供了用于优化平滑轨迹的梯度信息。 然而，计算距离信息需要花费额外的计算时间。

由于我们的激光雷达的测量范围很长（450m），光线投射非常耗时，尤其是当网格的尺寸很小来容纳小物体时。 另一方面，激光雷达的噪声比非常低（低于 0.0003%），这使得在 八叉树 中使用的光线投射是不必要的。 在本文中，我们直接规划从估计和建图模块接收到的原始点云。 为了实现有效的碰撞检查，我们将原始点云装入成KD 树结构。

如果将所有历史点云都考虑进碰撞检测会导致过于保守的运动规划， 此外，由于点的数量太多，在单个 KD 树中维护所有点是非常耗时的。 另一方面，仅在考虑当前扫描的点将导致 FoV 覆盖不完整并错过轨迹上的潜在障碍物。

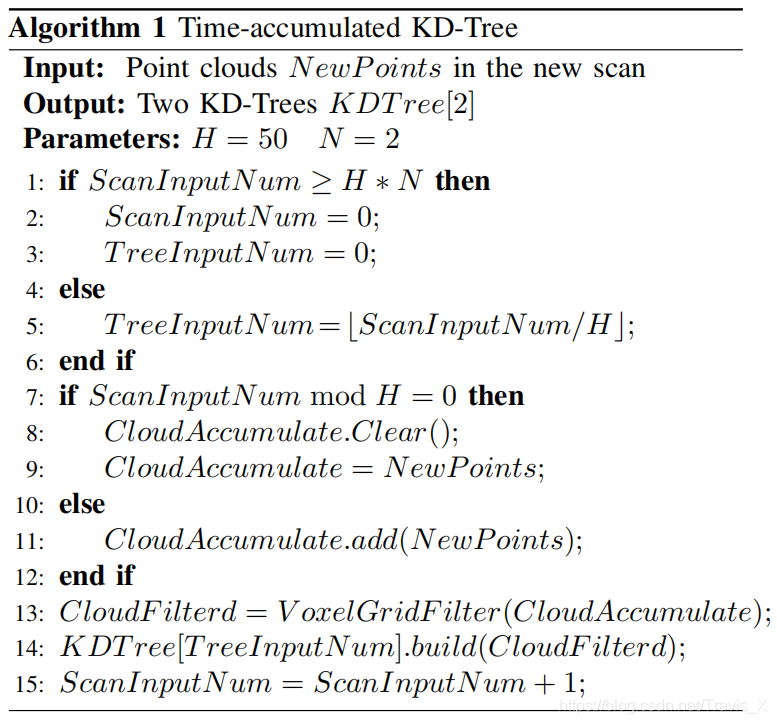
我们建议使用时间局部地图，即只使用最近的点云用于碰撞检测。我们在一定时间内累积点云，称为时间累积。 对于 livox AVIA 激光雷达，累积时间设置为一秒，从而产生足够高分辨率的点云（见图 4）。 我们以规定的分辨率（例如 10 厘米）进一步对原始点云进行下采样以降低计算量。 注意下采样分辨率，不像 Octomap 中的网格大小，不影响低于分辨率的物体的检测，但可能只会对障碍空间稍微进行膨胀。

局部地图被组成两棵KD树，一棵是静态的，一棵是动态生长的（见图6），每棵KD树都存储了累积时间的点云数据。 每个KD树的更新在算法 1 中有详细说明，其中 H = 50 表示存储在一个KD 树中的激光雷达最大扫描数，N = 2 表示 KD树的数量。 开始的时候，两棵 KD 树都是空的。 一旦新扫描的点到达，仍然装满的KD树被获得（第 1 - 6 。 然后，将新扫描的点添加到树上已有的点云中（第 11 行）。更新的点云以规定的分辨率（第 13 行）进行下采样，并用于构建一个新的 KD 树来替换现有的 KD 树（第 14 行）。 重复上述过程，直到当前的 KD树 已满（第 7 行），新的扫描从头开始累积（第 8-9 行）并保存到另一个 KD树（这将覆盖另一个 KD 上已有的点云）。



两棵KD树节省了两倍于累积时间的点云，足以覆盖激光雷达FoV。在进行碰撞检查时，应该同时使用两种KD树：静态的提供环境信息，而动态的则以激光雷达50Hz的扫描速率及时检测动态的障碍物。

备注：当添加新扫描的点云时，整个KD树使用标准 PCL 库 从头开始重新构建。 当KD树上新点的数量明显少于现有点时，这种重新构建可能非常低效。在这种情况下，可以使用增量KD树，它通过仅部分更新和重新平衡KD树来节省大量构建时间。



2. Kinodynamic A\* 搜索

一旦发现当前跟踪的轨迹上会发生碰撞（例如，由于动态或新检测到的障碍物），重规划模块被激活以基于保存在两个 KD 树中的最新局部地图和从FAST-LIO估计的当前无人机状态。为了避免动态障碍，重规划模块理应是轻量级的并且可以以高频率运行。 在我们的系统中，我们选择了基于运动原语的方法，该方法已成功用于高速规划和动态避障。

我们直接使用的是Kinodynamic A\* 搜索算法，其中通过 A\* 来构建和搜索图。该图使用 UAV 的起始状态（例如位置、速度等）进行初始化，我们将控制空间离散为一定数量的离散动作。 对于每个动作，我们在特定时间（即 T s T\_sT ）上对其进行整合，并获得动态可行的轨迹（称为运动原语）。 使用局部地图中的点（即两个时间累积的 KD 树）对每个运动基元进行碰撞检查。如果运动原语是无碰撞的并且满足运动动力学约束，则将其最终状态添加到图中。 这个过程称为扩展，不断重复直至到达目标点。

由于四旋翼无人机是微分平坦的，状态和实际控制可以从轨迹及其高阶导数（偏航角始终与速度方向对齐）唯一确定，这使我们能够使用高阶导数（例如 , 加速度, jerk, snap) 作为 Kinodynamic A\* 搜索中的虚拟控制。

**总结**

本文提出了一种自主无人机系统，状态估计、建图和轨迹规划都在机载电脑上执行，仅依靠激光雷达和惯性测量单元。介绍了硬件和软件系统的设计，并进行了大量的室内和室外实验。实验结果表明，所提出的系统能够在复杂环境中实现安全飞行，同时能避开动态小物体。所提出的系统使用 Kinodynamic A\*重规划的方法，其中控制（即加速度）空间被离散化产生运动基元。 搜索到的轨迹具有不连续的加速轨迹，防止无人机高速飞行，需要精确的轨迹跟踪。 少量的控制空间运动原语也限制了无人机在更紧凑的空间或以更高的重规划频率中运行。将来，我们会改进重规划模块，以实现更具挑战性的环境和更高的飞行速度。

