2024赛季视觉组第二轮考核部分题目

本次第二轮考核题目汇总如下，请大家务必仔细阅读，题目基础分值有100分，设立有相关加分项，总分可超过100分。

考核截止时间为8月29日24:00，以下所有题目需全部按要求打包，并汇总上传至个人GitHub或Gitee，提交时需告知仓库链接并公开访问。

提示：大小超过100m的文件需额外安装**Git LFS**进行上传。

相关题目附件或链接不再重复放置于此，务必仔细翻阅群文件。

1. C++

本题基础分值为10'

编写一个定时器函数，定时每隔一段时间，对输入的数据（需要使用 `std::vector<>` 作为函数参数，输入的数据类型和具体数据由自己而定）进行不定的操作，对数据的操作作为定时器函数的参数传入。例如

提示：这里作为函数的参数的操作也就是回调函数需要使用 `std::function`。
功能固定：函数定时周期为 `500ms` ，将外界随机生成的一百个数封装至`std::vector`作为参数传入此函数，然后将**求和、求平均值、求方差、求累乘乘积**四种操作(普通函数)一起作为参数传入此函数。在函数中随机选择一种操作方式，对传入的被封装好的100个数求值，将结果打印到终端上验证是否正确。相当于这是个死循环线程，每500ms就执行一次随机操作。
提示：`float createTimerLoop(int period, std::vector<float>& data, std::vector<std::function<float (std::vector<float>&)>>& operations)`
将上述功能实现。提供源码，然后也可以想想，如果使用`std::vector`的话，`data`和`operation`是只能举例要求的100和4个吗，是否数据和操作都是可动态扩展修改的。仅做思考，增加对`stl`和自己设计函数和类时对可扩展性的理解。

提交源码并在代码中加入相关功能说明注释。

2. ROS基础应用

本题基础分值为30'

黄金矿工

我们来写一个简单的黄金矿工采矿的游戏，这里有个三维空间具有未知数的矿石，此空间我们称为`矿界`，`矿界`向外部**发布**矿石信息(例如矿石编号、价值、所在位置信息等)，`采矿人`需要**订阅**这些矿石信息并进行采集。规定每一步`采矿人`都选择离自己欧式距离最近的矿石作为目标矿石，采取机制就是需要采矿人向`矿界`发送一个采矿**请求**，`矿界`里有一个`server`，负责处理`采矿人`发送的请求，如果`矿界`收到了人给他的请求(`request`)后，`矿界`收到请求后就将`采矿人`移动到那个位置并让他采集到**(P.S.这个就是虚拟移动和采矿，矿界直接将采矿人传送到选择的矿石处，然后瞬间采矿完毕)**，然后`矿界`做出回应

`response`，`response`内容数据为`采矿人`当前位置和`采矿人`已经取得所有矿石的价值和。采矿人初始化时默认在`(0，0，0)`处。

现在需要实现这个功能，一个`矿界node`包含发布矿石信息的`topic`和处理采矿人的`request`信息。一个`采矿人node`负责发送`request`信息和接收`response`并存储采矿人自己的位置和已经获取的矿石总价值。

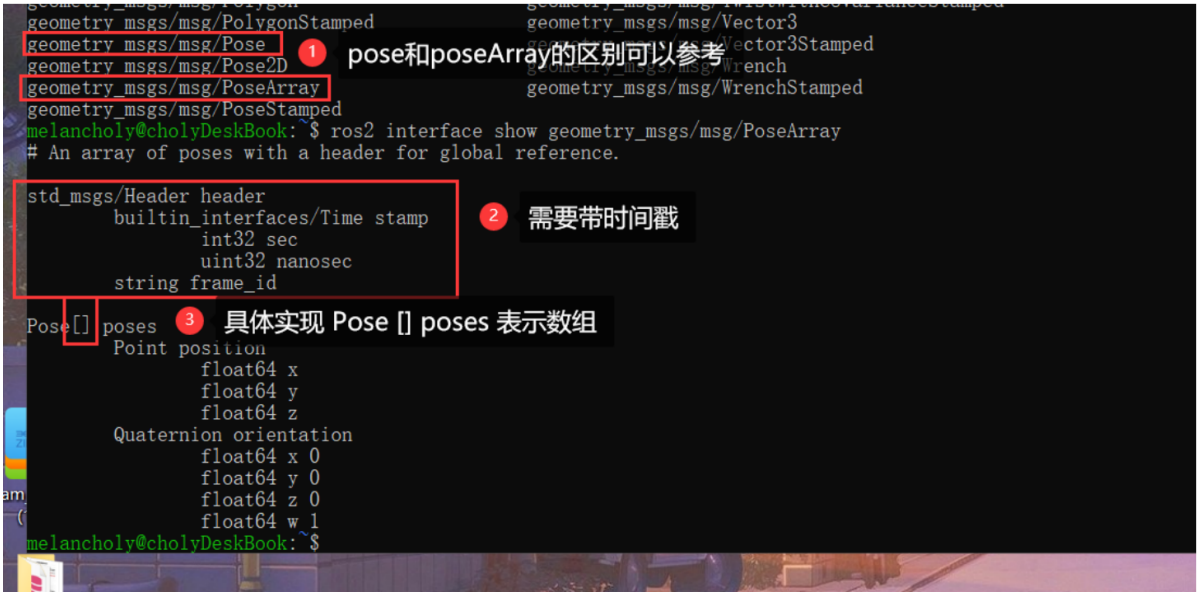
2.1 第一步：自定义`interface`矿石信息

现在假设在一个`矿界`(三维空间)存在`n`个金矿，其中`n`需要声明为`parameter`，默认值设置为小于10的数，可使用`params.yaml`和`launch`传入(实际也传少一点，意思意思就行了)。然后这个`矿界`需要使用`topic`功能向外发布(`publish`)矿石信息，其中矿石信息需要自定义`interface`，其中矿石信息需要包含以下几类数据(--->的后侧内容 表示提示的ros标准interface)。

- 矿石编号（**int类型**）-->std_msgs
- 矿石类型（**金矿 or 银矿string**）-->std_msgs
- 矿石所处位置（**position: x,y,z 坐标**）-->geometry_msgs
- 矿石价值（**金矿-80.8，银矿-40.4，double类型**）-->std_msgs

因为`矿界`实际发送时发送矿石数组，因此需要再定义一个`interface`，就是使用已经定义好的`矿石信息`，然后定义一个矿石信息数组的`interface`。需自行学习如何自定义`interface`和其数组。

给出下方案例可供学习。可用`ros2 interface show ...`来查看具体`Pose`和`PoseArray`的区别。



2.2 第二步：矿界发送矿石信息

要求创建一个`矿界`ros node。写成规范类格式继承`rclcpp::node`的节点
生成`n`个矿石信息，`n`由外部`parameter`决定，这`n`个矿石信息按`1~n`进行编号，n个矿石的位置由随机数生成，要求限制在离原点`(0,0,0)`圆心距离`10`的球形范围内，然后`矿界`创建`publisher`发布消息。

2.3 第三步：采矿人订阅、发请求和构建服务端

创建一个`矿界`ros node。写成规范类格式继承`rclcpp::node`的节点 。

`采矿人`订阅话题，然后选出最近的矿石，并选择采该矿石。因此需要向`矿界`发请求响应，`矿界`节点需要在新建一个`服务器server`来接受请求并反馈`response`。`srv`的通信`interface`需要自行定义，`请求`为想要采集的矿石编号，回应`response`的是`采矿人`当前的位置(被传送到矿石处并立即采矿成功)和已经获得的矿石的总价值，并且当采矿人采了矿后，矿界停止发布该矿石信息。

例如`interface`可定义信息：

```
float64 id
---
Point position
    float64 x
    float64 y
    float64 z
float64
```

2.4. （加分项）第四步：循环往复直到取得所有矿石，计算总收益

本题加分10'

本部分为扩展题目属于加分项，若上述步骤均已实现，可尝试使用Rviz将题目所述过程可视化，即将行走路线用箭头标出来方向。

提交源码并在代码中加入相关功能说明注释。

3. 坐标转换

本题基础分值为10'

3.1 启动培训给定的urdf文件

本小题——2'

- **提示**： 给定文件中已有launch文件
- 使用`rviz2`打开，打开`TF`插件，看一下有几个坐标系，拖动小滑块让`yaw`轴和`pitch`轴动一下
- 使用`tf2_tool`s或`rqt`查看`tf`坐标转换树并截图
    - `tf2_tools`

      ```
      ros2 run tf2_tools -h
      ```

    - `rqt` 安装`rqt`插件

      ```
      sudo apt install ros-humble-rqt-tf-tree
      ```

      开启方式:`plugin-->visualization-->TF tree`

3.2 入门tf

本小题——2'

- 学会使用`static_publisher`，并演示效果，使用Rviz可视化。
- 学会使用`tf2_echo`

  ```
  ros2 run tf2_ros tf2_echo -h
  ```

  使用命令`echo`读取上面启动的`urdf`中的`tf`树中的两个坐标系之间的变换。

3.3. 代码编写

3.3.1. `lookTransform`

本小题——2'

使用此API得到`camera_optical_link`到`gimbal_odom`之间的转换。并将数据使用`RCL_CPP`打印到终端输出。

3.3.2. `tf2_ros::Buffer::transform`

本小题——4'

要求使用此API将某坐标在两个不同坐标系之间进行转换。

现假设三维空间中有一点`Point`，它在`camera_optical_link`下坐标为`(3，4，5)`，使用API将此空间点转换到`gimbal_odom`坐标系下。两个坐标系之间的转换由上方启动的`urdf`文件提供。需要再把得到的坐标`(x，y，z)`从`gimbal_odom`下转换到`camera_optical_link`下，如果得到的是`(3，4，5)`，说明`transform`成功。

提交源码并在代码中加入相关功能说明注释。

4. OpenCV

本题基础分值为25'，可额外加5'

编程实现简单的能量机关识别，读入大能量机关旋转视频，并进行能量机关待击打装甲板的识别。要求如下：

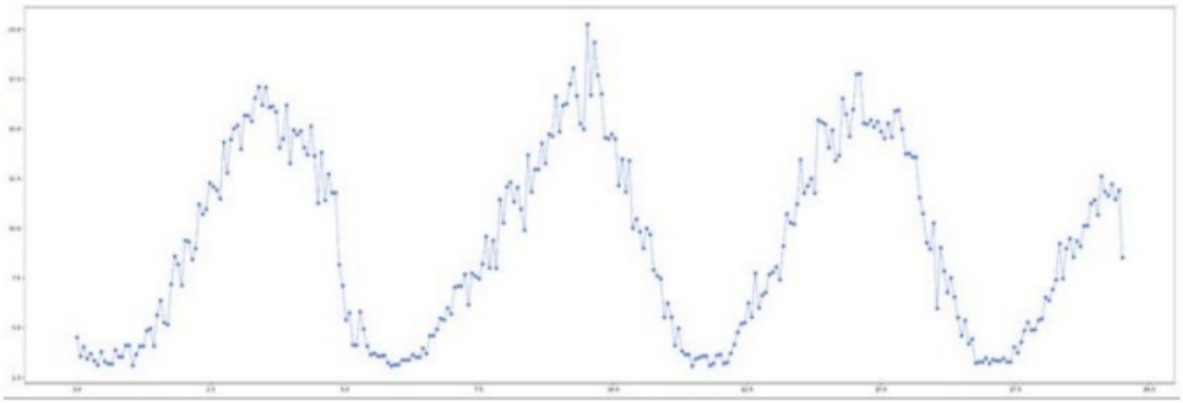
- 编程环境：OpenCV 4.5+，Python 3.9+ or GCC支持C++11及以上。
- 代码规范：使用面向对象编程方法，须有统一的代码规范（可参考谷歌代码规范），有适量的清晰的注释。——3'
- 功能实现：
  - 识别待击打装甲板——6'
  - 识别旋转圆心——3'
  - 旋转方向判断——3'
  - 旋转预测

    > 基础：设定能量机关转速为10rpm，弹丸飞行时间为1s——3'
    >
    > 进阶：实时监测能量机关转速，弹丸飞行时间为1s——加额外5'
- 输入：能量机关视频（文件中提供的北理珠大能量机关视频）

    - 读取视频时。同意在循环最后执行`cv::waitKey(1)`
    - 顺利读取视频——1.5’
- 输出：目标装甲板矩形框四点在图像中的坐标信息——1.5’
- 运行效率：主要运算模块耗时<=60ms，需输出耗时——2’
- 调试输出：用矩形框实施绘制出目标装甲板在原图像中的位置——1‘、用圆形绘制圆心的位置——1’

（说明：若实现了识别待击打装甲板功能，则目标装甲板为待击打装甲板轮廓；若完成了旋转预测，则目标装甲板为待击打装甲板轮廓金国旋转预测后得到的矩形。）

需提交源代码（带注释）和调试输出录像，若实现进阶版旋转预测，则附上旋转出的转速虽时间变化折线图（体现正弦规律）



5. Rviz2 和 Gazebo使用

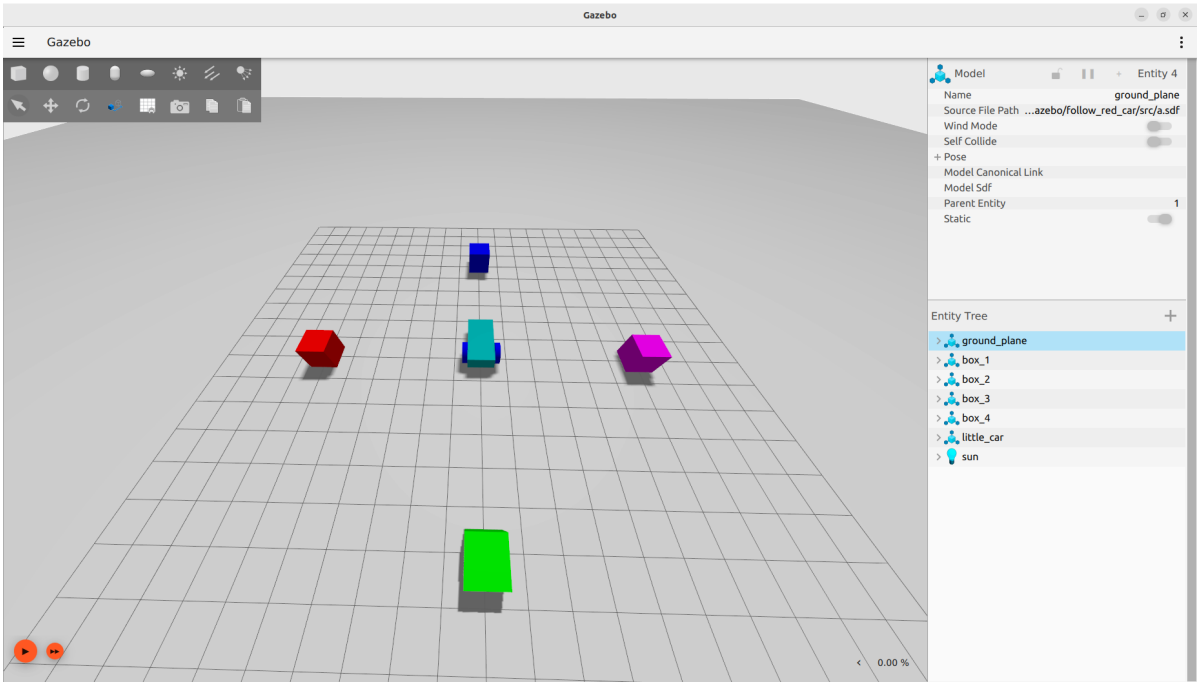
本题基础分值为15’

自行在Ignition Gazebo中搭建如下场景。

场景要求：小车在世界中心，四个不同颜色的正方体正对小车的前后左右方向，且四个正方体距离小车的位置一致，颜色可自行拟定。小车至少添加一个相机或其他传感器。

题目要求：使用ROS2编写程序驱动小车寻找指定颜色的正方体，同时小车可自动行驶至对应颜色正方体的旁边，颜色可自行制定，不可写死程序使小车只往一个颜色的正方体行驶，同时Rviz2可订阅节点看到小车的相机画面。

录屏小车运行时行驶画面和Rviz2小车相机画面的视频，代码文件必须加上相关函数的功能解释注释，并打包提交。



6. 卡尔曼滤波

分为基础题目和进阶题目，二选一即可，或两个都做（时间充裕的话）。

基础题目：运用卡尔曼滤波器对直线轨迹滤波(给一个真值加噪声再过滤)

本题基础分值为5'

要求如下：

- 数据点至少大于200个
- 添加噪声1% - 10%
- 可根据情况对QR矩阵进行尝试调参，并附上不同情况下的滤波效果
- 要求能看出滤波效果与真值的匹配情况
- 不限制编程语言C++、python、matlab均可。

进阶题目：二阶卡尔曼滤波器

自编写二阶卡尔曼滤波器类，不限编程语言，不可使用现成的库函数。

使用OpenCV检测素材视频中的小球并预测其运动轨迹，要求在原图像中绘制小球当前的中心点和预测得到的中心点。

预测效果参考视频：卡尔曼滤波预测坤坤运球

打包提交代码（需带有相关功能说明的注释）和输出的滤波效果图片或视频。

7. YOLO

本题基础分值为5'，可额外加5'

使用YOLOV5-Face自训练2023赛季新能量机关检测模型，详细要求如下：

- 使用下方链接自行搜集大符数据集，数据量要求2000张以上。（建议大家组队收集）
- 需要按照一定比例区分数据集为验证集和训练集。
- 自行搭建环境或租赁算力平台进行训练。（自行配置环境较为麻烦，需要搭建Anaconda+cudnn+cuda+Pytorch，若时间匆忙建议租赁算力平台）
- 训练出来的模型对素材视频中扇叶和R标的置信度均在0.9以上。

加分项：

- 自行编写程序区分验证集和训练集。——加3’
- 使用YOLOv8进行训练。——加2’

提交包含检测结果的输出视频，附上调参过程记录（记录参数在YOLO的哪个代码文件和第几行代码中，调整的数值为多少。）若有自行编写的区分数据集程序，则一同提交，并在代码中简要注释相关函数的功能。