自动运动模型标定web与单机交互步骤

0. 版本

版本	作者	审阅人	更新日期
v 1.0	王杰	焦健	2019-10-31

一、标定步骤

1、标定，分步骤，从目前10个步骤中进行，依次进行，目前定为顺序进行，无法跨序。

2、过程从web打开页面，访问web后台API开始，web服务从yaml中`general/use_calibration_step_vec`获取当前的标定进度，web页面跳转到对应的标定步骤。

3、读取`general/calibration_real_param_vec`默认参数，将`general/calibration_real_param_vec`默认参数写入到页面中。

4、用户修改确认完页面参数后，点击保存，将参数均写入到`general/calibration_real_param_vec`中去。

5、此时点击开始标定，web后台进行call service服务`/($AGENT_SERIAL)/calibration_confirmation_param_srv`进行标定，携带参数为`{param_name="start",id=1}`，返回值根据 confirm 进行判断call_service是否成功。

6、同时监听topic中`/($AGENT_SERIAL)/calibration_state`，然后根据返回的 id状态2确定标定完成。

7、完成后，从topic中字段<?>，进行弹窗展示，并给用户提示选择是重复标定还是下一步。

8、若下一步，则继续call service服务`/($AGENT_SERIAL)/calibration_confirmation_param_srv`进行标定，携带参数为`{param_name="<<calibration process parameters>>",id=1}`，并且更改yaml中`general/use_calibration_step_vec`标定步骤。

9、若重复，则call service服务`/($AGENT_SERIAL)/calibration_confirmation_param_srv`再次进行标定，携带参数为`{param_name="<<calibration process parameters>>",id=0}`

10、依次进行，直至完成标定结束。

二、标定交互，web部分与运动控制部分

1、用于确定标定结果，再次确认的service

```
ROSinterface:serviceserver
```

```
Servicename:/($AGENT_SERIAL)/calibration_confirmation_param_srv
Servicetype:bito_msgs/GetParamsSrv
Servicedescription:
Request:
string param_name
# 重新开始标定 0
# 继续进行下一步 1
int8 id
---
Response
bool confirm 返回true时，成功；返回false时，失败。
int8[] param_flag1
float64[] double_data 返回的标定数据，是一个10个元素的参数
int64[] int_data

```

2、用于监听的标定状态的topic

```
ROSinterface:publisher
Topicname:/($AGENT_SERIAL)/calibration_state
Servicetype:bito_msgs/AutomataStateMsg
Topicscription:
# Waiting 0
# RunCalibration 1
# DoneCalibration 2
# FatalStop 3
int32 id
```

3、配置yaml文件

```
# <<model tyep>>
# differential wheel: 0;
# mecanum wheel: 1;
# steering wheel(like forklift): 2;
general/model_type: 2

# <<User perference>>
# need time to steer the motor to the specific angle
general/use_motor_control: true
# use_confirmation: confirmation to start each unit calibration process
general/use_confirmation: false

# <<Steps user preference>> which calibration process step will be used
general/use_calibration_step_vec: [true, true, true, true, true, true, true, true,
false, false]

# <<motion params>>
# time when the robot doing linear motion
general/linear_motion_duration: 15
# time when the robot doing angular motion
general/angular_motion_duration: 15
# linear x velocity when the robot is doing linear x motion
```

```
general/linear_x_vel: 0.7
# linear y velocity when the robot is doing linear y motion
general/linear_y_vel: 0.5
# angular z velocity when the robot is doing angular z motion
general/angular_z_vel: 1.0
# acceleration for different calibration process
general/acceleration_vec: [0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1]

# <<calibration process parameters>>
# {
#    steering_angle_offset ,
#    steering_angle_scale,
#    steering_linear_velocity_scale,
#    steering_angular_velocity_scale,
#    odom_steering_angle_offset,
#    odom_steering_angle_scale,
#    odom_steering_linear_velocity_scale,
#    odom_steering_angular_velocity_scale,
#    steering_velocity_backward_scale,
#    odom_steering_velocity_backward_scale,
# }
# maximum count to know if the calibration converged
# this is a 8 elements vector for different calibration process
general/converge_max_count:  [3, 3, 3, 3, 3, 3, 3, 3, 3, 3]
# checker_max_count check calibration parameter for this times,
# if check times exceed this count, will stop the calibration process immediately,
# and start next calibration process.
general/checker_max_count: [10, 10, 10, 10, 10, 10, 10, 10, 10, 10]
# real calibration params writen currently in chassis, this param could be be read from
chassis as well,
# should be replaced soon.
general/calibration_real_param_vec: [0, 1, 1, 1, 0, 1, 1, 1, 1, 1]
# calibration_param_map_offset is the offset that
# calibration_real_param_vec-calibration_param_map_offset=[0]
general/calibration_param_map_offset: [0, 1, 1, 1, 0, 1, 1, 1, 1, 1]
# calibration step length is the step length which will be used in each calibratio
process gradient search process.
general/calibration_step_length: [150, 0.015, 0.011, 0.012, 100, 0.02, 0.1, 0.1, 0.012,
0.012]
# calibration threshold is the convergence threshold which will judge if the
calibration process converged in this threshold.
# calibration error should be in [0, threshold] range.
general/calibration_threshold_vec: [0.001, 0.001, 0.0001, 0.001, 0.001, 0.001, 0.001,
0.001, 0.001, 0.001]

```

三、web组内部分

web服务端提供接口给客户端

> 1、获取当前标定进度

> 2、获取标定参数

> 3、写入参数到yaml中

> 4、进行标定

> 5、提供topic的reply给前端

上述步骤应该在设置完车体的物理参数之后, 为了方便模块化测试可以在做完这个自动标定之后进行.