# eai\_robot\_control ros 功能包接口说明

## 1.Eai\_robot\_control 功能包实现功能

在功能包是在eai平台下，实现精准走直线，转特定角度，带参考校准走直线，两点之间走直线且激光会检测前方是否有障碍物，有则停止。

## 具体接口说明

### 2.1 move\_rotate 接口

作用： 做动特定角度，误差在1.5度以内

参数： angle ：需要转动的角度，单位为弧度，例如3.14为180度

返回值： 正常执行完，返回true

注意：1.执行该函数时，会阻塞等待执行完转动返回结果

2.转动时，会依靠陀螺仪的值来进行判断是否转动指定角度，因此必须有陀螺仪

bool move\_rotate(float angle);

### 2.2 move\_linear 接口

作用： 不带校准，直线走指定距离，误差在1.5cm左右

参数： linear\_distance ：需要行走的距离，单位为m

返回值： 正常执行完，返回true

注意：1.执行该函数时，会阻塞等待执行完行走返回结果

2.行走时，依靠odom计算行走距离

bool move\_linear(float linear\_distance);

### 2.3 move\_linear\_with\_adjust 接口

作用： 带校准，直线走指定距离，误差在1.5cm左右

参数： linear\_distance ：需要行走的距离，单位为m

angle\_adjust : 行走时，当前的参照角度值，单位为弧度，例如以当前imu值

(假如为0.0)为参考，误差在正负0.05弧度(约3度)，当陀螺仪值超过0.05时，

就会通过设置角速度转动回来

返回值： 正常执行完，返回true

注意：1.执行该函数时，会阻塞等待执行完行走返回结果

2.行走时，依靠odom计算行走距离，校准时，依靠陀螺仪的值

bool move\_linear\_with\_adjust(float linear\_distance,float angle\_adjust);

### 2.4 run\_straight\_line 接口

作用： 设定起点和目标点(位置和方向)，两点之间带校准，走直线，激光检测前方范围是

否有障碍物，如果有就会暂停，并每5s 检测前方是否还有障碍物，如果没有会继

续行走，否则会继续停止。

参数： start\_pose : 起点坐标位置和方向

target\_pose : 目标点坐标位置和方向

返回值： 正常执行完，返回true

注意：1.执行该函数时，会阻塞等待执行完行走返回结果

2.行走时，依靠odom计算行走距离，校准时，依靠陀螺仪的值

3. 激光会检测前方范围是否有障碍物，判断是否需要停止，具体检测范围在

dashgo\_tools/conf/box\_filter.yaml 中，(之后会改)

bool run\_straight\_line(geometry\_msgs::Pose start\_pose,geometry\_msgs::Pose target\_pose);

### 2.5 getCurrentImuAngle 接口

作用： 获取当前机器人陀螺仪的值(角度方向)，单位弧度

参数： 无

返回值： 正常执行完，返回当前imu 的值，单位弧度

注意：

float getCurrentImuAngle();

## 接口使用范例

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

{

ros::init(argc,argv,"eai\_robot\_control");

RobotControl rc;

//原地转动,此函数会阻塞，直到转动完成返回结果

if(rc.move\_rotate(3.14))

{

cout<<"excute move\_rotate ok"<<endl;

}

//不带校准的直线行走

if(rc.move\_linear(1.0))

{

cout<<"excute move\_linear ok"<<endl;

}

//带校准的直线行走，0.0为参照的角度值，

if (rc.move\_linear\_with\_adjust(1.0,0.0))

{

cout<<"excute move\_linear\_with\_adjust ok"<<endl;

return 0;

}

//两点之间走直线

start\_pose.position.x=0.0;

start\_pose.position.y=0.0;

start\_pose.position.z=0.0;

start\_pose.orientation=tf::createQuaternionMsgFromYaw(0);

target\_pose.position.x=1.0;

target\_pose.position.y=0.0;

target\_pose.position.z=0.0;

target\_pose.orientation=tf::createQuaternionMsgFromYaw(0);

if(rc.run\_straight\_line(start\_pose,target\_pose))

{

cout<<"excute run\_straight\_line ok"<<endl;

}

ros::AsyncSpinner spinner(boost::thread::hardware\_concurrency());

spinner.start();

}