

map_listener_reference

功能：给机器人提供一个功能性质的 api

def decompression(data):

字符串转 list，消息解压缩

输入：str 数据

输出：list like [[],[],[[[]]]

def ort_pose(yaw,pose,r):

机器人朝向坐标

输入：yaw,pose,r (角度，坐标，半径)

输出：【x,y】机器人朝向角坐标

def quat_to_angle(quat):

四元组转成弧度

输入：四元组

输出：弧度

def angle_to_quat(angle):#in rad

角度转四元组

输入：弧度

输出：四元组

def map_center_cell(map_origin,height,resolution):

#real-world (0,0) in matrixs frame 即 cell 坐标

输入：map_origin(OccupancyGrid.info.origin),height,resolution

输出：现实坐标原点 (0,0) 在矩阵数据中的位置，centremap_cell=[x,y]，x,y 表示矩阵。

注释：# The origin of the map [m, m, rad]. This is the real-world pose of the cell (0,0) in the map。map_origin为 cell(0,0)的坐标

def robot_pose_cell(map_origin,odom_pose,height,resolution):

#返回机器人的 cell 坐标点格式为[x 格数,y 格数]，即机器人在矩阵中的位置

输入：map_origin,odom_pose,height,resolution

输出：机器在矩阵中的位置

def metrix_RealPose(map_origin,goal_cell,goal_ort,height,resolution):

#the real world position of a specific point in metrix 格式 pose()

输入：map_origin,goal_cell,goal_ort,height,resolution。(goal_cell,goal_ort 表示期望机器人将要到达地图上的位置，以及姿态)

输出：现实中的目标点坐标以及姿态

def

robot_goal_compensated(map_origin,goal_cell,goal_ort,height,resolution,data):

#figger out reasonabel goal for robot moving

输入：map_origin,goal_cell,goal_ort,height,resolution,data

输出：经过过滤后的坐标点，考虑到机器人的大小占位，以防止发布的坐标机器人无法到达，从而 plan Abortions。该 api 仅仅接受已知地图上的点，使用方法：先使用 edge_explorer 或者其他 api 获取目标点坐标，然后调用该 api，获取补偿坐标点。

def map_matrix_ranger(data):

#返回地图矩阵

输入：data (OccupancyGrid)

输出：map_matrix n*m 的地图矩阵，长：data.info.height。宽：data.info.width

```

def effective_point(data):
    #返回有效区域的坐标集
    输入：data (OccupancyGrid)
    输出：[clear_area,block_area],返回地图上所有有效点的点集，分别为 clear_area 和
    block_area。

def edge_explorer(data):
    #边界遍历程序，检测地图是否完整。
    输入：data (OccupancyGrid)
    输出：checker[[[]],[[]]]
    注释：#checker[[[]],[[]]]， checker[0]是用来检测所有的 y 坐标轴是否闭合，checker[1]是用
    来检测 x 坐标轴是否闭合。11 表示闭合，00 表示敞开，10 表示左边单边闭合，01 表示右边单边闭合。该
    api 可以返回某些行或者某些列的区间是打开的。

def effective_point_cell(data):
    #返回有效区域的 cell 坐标集 data 是订阅 map 的完整数据
    输入：data (OccupancyGrid)
    输出：[clear_area,block_area]

def nearest(goal,list_num):
    #输出一系列点中最近的一个
    输入：goal,list_num
    输出：距离 goal 最近的一个 num

def global_action_director(data):
    #全局行动方位决策
    输入：data (OccupancyGrid)
    输出：[[detector_area_x,detector_area_y],[goal_x,goal_y]]
        即应该扫描区域范围和目标 (x,y) 坐标
    注释：
        x 轴从第一行到最后一行都对应一个状态，x 轴的状态包括
        00：开环(包括 0000 以及 0110 两种情况)
        11：闭环(包括 1111 以及 1001 两种情况)
        01：左边单开的 x 轴
        10：右边单开的 x 轴
        -1：未探明区域
        y 轴从第一列到最后一列都对应一个状态，y 轴的状态包括
        00：开环(包括 0000 以及 0110 两种情况)
        11：闭环(包括 1111 以及 1001 两种情况)
        01：上边单开的 x 轴
        10：下边单开的 x 轴
        -1：未探明区域

        x_closed：闭环的 x 轴集合
        x_opened：开环的 x 轴集合
        x_ltopen：左边单开的 x 轴集合
        x_rtopen：右边单开的 x 轴集合
        x_undect：未探明区域
        y_closed：闭环的 y 轴集合
        y_opened：开环的 y 轴集合
        y_ltopen：左边单开的 y 轴集合
        y_rtopen：右边单开的 y 轴集合
        y_undect：未探明区域

def store_xlsx(data,high,width):

```

```
#地图数据存储 xlsx for testing convenient  
输入：data (OccupancyGrid)  
输出：'ok'  
将数据存入当前目录的 cellmap.xlsx 文件中
```

```
def store_txt(data):  
    #地图数据存储 txt for testing convenient  
    输入：data (OccupancyGrid)  
    输出：'ok'  
    将数据存入/home/%s/mapdata/test1.txt 文件中
```