该文件中表明了机器人底盘的尺寸参数和障碍物的膨胀参数 对于具体的机器人来说，要把相应的参数修改成机器人车体一致。 #传感器读数的最大有效高度，单位为 meters;

通常设置为略高于机器人的实际高度，高度是指包含机械臂打直情况下的最大高度。

max\_obstacle\_height: 0.60 # assume something like an arm is mounted on top of the robot

robot\_radius对一个圆形的机器人来说,这是机器人的半径,单位是米;对一个非圆形的机器人来说,

你可以用接下来说到的 footprint参数。这里用到的值是 Turtle的原始数据。

robot\_radius: 0.15 # distance a circular robot should be clear of the obstacle (kobuki: 0.18)

footprint: [[x0, y0], [x1, y1], ... [xn, yn]] # if the robot is not circular

在列表中的每一个坐标代表机器人的边上的一点,机器人的中心设为[0,0]。测量单位是米。

这在机器人周长上的点要不按照顺时针排列,要不按照逆时针排列。

当机器人非圆形时，先找机器人的旋转中心，即两个轮的中心点设置成(0,0)，然后确定机器人摆放方向，x,y为每个robot几何型的每条边的每个顶点。将所有顶点都列到其中。就完成了robot的footprint

footprint: [[0.55, 0.375], [0.55, -0.375],[-0.55, -0.375], [-0.55, 0.375]]

图类型，这里为voxel(体素地图)。另一种地图类型为costmap(代价地图)。

这两者之间的区别是前者是世界的3D表示，后者为世界的2D表示。

map\_type: voxel

障碍物层参数

obstacle\_layer:

enabled: true #启用

传感器读数的最大有效高度（单位：m）。 通常设置为略高于机器人的高度。 此参数设置为大于全局max\_obstacle\_height参数的值将会失效。

设置为小于全局max\_obstacle\_height的值将过滤掉传感器上大于该高度以的点。

max\_obstacle\_height: 0.6

min\_obstacle\_height: 0.0

cbination\_methoomd: 处理obstacle\_laye之外的其他层传入数据的行为方式，

枚举型(enum)。可能的取值有：覆盖已有值"Overwrite" (0), 取最大值"Maximum" (1), 什么也不干"Nothing" (99)。“覆盖”仅是“覆盖”其他层的数据，例如使得它们没有生效。 “取最大值”是多数时候需要的。 它提取obstacle\_layer或输入数据中提供的数据的最大值。 “Nothing”根本不会改变传入的数据。

# 请注意，这会极大地影响costmap的行为方式，具体取决于您对track\_unkown\_space的设置。

combination\_method: 1

track\_unknown\_space 如果为false，每个像素具有两种状态之一：

致命障碍(lethal)或自由(free)。 如果为true，则每个像素具有3种状态之一：

致命障碍(lethal)，自由(free)或未知(unknown)。

track\_unknown\_space: true #true needed for disabling global path planning through unknown space. 即true时，不能穿过未知区域；false时，穿过未知区域。

这两个参数用来设置代价地图的障碍信息的阈值。

“obstacle\_range”更新以机器人为中心，按此数值为半径内的障碍物信息。

将障碍物插入代价地图的最大范围，单位为 meters。

obstacle\_range: 2.0

“raytrace\_range”更新以机器人为中心，按此数值为半径内的自由空间。

从地图中扫描出障碍物的最大范围，单位为 meters 。

raytrace\_range: 5.0

#现在未使用

origin\_z z原点，单位为米，仅对voxel地图

#origin\_z: 0.0

#z\_resolution z分辨率，单位meters/cell；

#z\_resolution: 0.2

#z\_voxels 每个垂直列中的体素数目，ROS Nav功能包的默认值为10。

#z\_voxels: 2

# unknown\_threshold: 15 当整列的voxel是“已知”(``known’’)的时候，含有的未知单元(“unknown”)的最大数量

mark\_threshold: 0 整列voxel是“自由”(“free”)的时候，含有的已标记的cell(“marked”)的最大数目。

publish\_voxel\_map: 是否发布底层的体素栅格地图，其主要用于可视化。

publish\_voxel\_map: false

“observation\_sources”参数定义了一系列传递空间信息给代价地图的传感器。每个传感器定义在下一行。

观察源，我们这里是激光数据(scan)和凸点数据(bump)。观察源列表以空格分割表示，定义了下面参数中每一个 <source\_name> 命名空间。

observation\_sources: scan

scan:

data\_type: LaserScan

topic: "/scan"

“marking”和“clearing”参数确定传感器是否用于向代价地图添加障碍物信息，

#或从代价地图清除障碍信息，或两者都有。

marking: true

clearing: true

expected\_update\_rate: 0

注：关于Marking and Clearing：

marking和clearing参数用来表示是否需要使用传感器的实时信息来添加或清除代价地图中的障碍物信息）

代价地图自动订阅传感器主题并自动更新。

每个传感器用于标记操作（将障碍物信息插入到代价地图中），清除操作（从代价地图中删除障碍物信息）或两者操作都执行。

如果使用的是体素层，每一列上的障碍信息需要先进行投影转化成二维之后才能放入代价地图中。

#cost\_scaling\_factor and inflation\_radius were now moved to the inflation\_layer ns

#膨胀层参数

inflation\_layer:

enabled: true #启用膨胀地图

# cost\_scaling\_factor 在膨胀期间应用于代价值的尺度因子。默认值：10。对在内接半径之外的cells、以及在内接半径之内的cells这两种不同的cells, 代价函数的计算公式为：

exp(-1.0 \* cost\_scaling\_factor \* (distance\_from\_obstacle - inscribed\_radius)) \* (costmap\_2d::INSCRIBED\_INFLATED\_OBSTACLE - 1)

cost\_scaling\_factor: 10.0 # exponential rate at which the obstacle cost drops off (default: 10)

这里我们设置机器人的占用面积，按机器人坐标设置填入“footprint” 指定的footprint时,机器人的中心被认为是在(0.0,0.0),顺时针和逆时针规范都支持。

robot\_radius ：是机器人的占用面积为圆形，直接设置半径即可

inflation\_radius：设置代价地图膨胀半径。可以理解以机器人为中心膨胀半径为此数值内不能有障碍物出现。

地图上的障碍物的半径,单位为米。如果你的机器人不能很好地通过窄门或其他狭窄的地方,

则稍微减小这个值。相反的,如果机器人不断地撞到东西,则尝试增大这个值。

inflation\_radius: 0.1 # max. distance from an obstacle at which costs are incurred for planning paths.

static\_layer:

enabled: true

map\_topic: "/map"

# 附录：

bump:观察源之二：凸点数据。定义了：观察源的数据类型，发布话题，标记和添加障碍物功能及定义传感器源数值的有效范围

data\_type: PointCloud2 数据类型为点云

topic: mobile\_base/sensors/bumper\_pointcloud Topic为mobile\_base/sensors/bumper\_pointcloud

marking: true 启用标记障碍物功能

clearing: false 关闭清除障碍物功能

min\_obstacle\_height: 0.0 传感器最低有效读数，以米为单位。

max\_obstacle\_height: 0.15 传感器读数的最大有效高度，以米为单位。