

MaxL Controller Tuning Guide

This is a guide that will describe the steps to tune the MaxL controller for any deployment of the Mowito Navigation Stack. This guide shall provide all the parameters that are required to be tuned, their significance and description of what the parameters mean. This guide is typically meant for the end users who will be using the Mowito Navigation stack and have deployed the navigation stack on their respective hardware. This guide will only address the controller and obstacle avoidance functionality of the Mowito Navigation Stack.

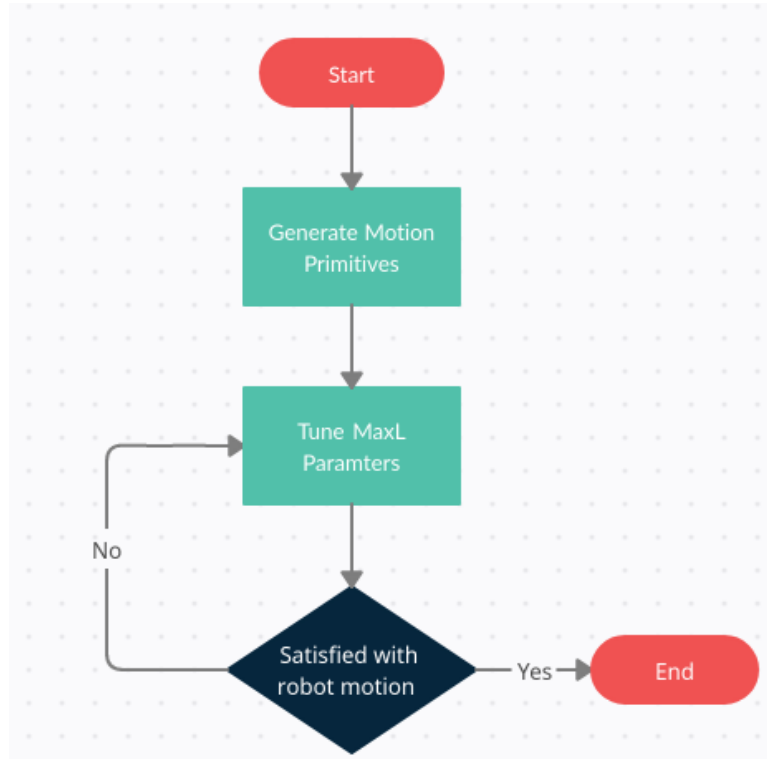
1.Contents

Sl. No	Topic	Page Number
1	Steps to tune the MaxL Controller	2
2	Step 1 : Generating Motion Primitives for the Controller	3
3	Step 2 : Configuring the MaxL parameters	9

2.Steps to tuning the MaxL Controller

The MaxL controller is a proprietary state of the art control and obstacle avoidance system that has the ability to process information and control the robot and avoid obstacles with a refresh rate as high as 50 Hz. In order to use the MaxL controller provided in the Mowito Navigation Stack, the controller is required to be tuned.

The following flow chart shall highlight the steps to follow for tuning the MaxL controller.



As seen in the flowchart, the process of tuning the controller involves two major steps :

- 1) Step 1 : Generating the motion primitives
- 2) Step 2 : Tuning the MaxL Parameters

Step 1 : Generating the Motion primitives involves generating a set of paths the robot can take during the robot motion. These set of paths are critical and depend on the robot dimensions. This is generally a one time step.

However, any change in the robot dimensions will require motion primitives to be regenerated.

Step 2 : Tuning the MaxL Parameters involves setting the MaxL parameters in the `mw_maxl_planner.yaml` file which is generally located in the `controller_config` folder in your repository. The values in this file can be modified based on the desired robot motion and can be set as many times until the desired robot motion is achieved.

Section 3 shall describe the method to generate the motion primitives and Section 4 shall describe the methods to tune the MaxL parameters located in the mw_maxl_planner.yml file.

3. Step 1 : Generating Motion Primitives for the Controller

As stated in section 2, the Motion Primitives are a set of precomputed paths that the robot can take while the robot is in motion. Whenever an obstacle confronts the robot, some of the precomputed paths are blocked and the controller chooses a path from the set of paths that are not blocked.

While tuning the controller generation of these precomputed paths is a mandatory first step. To generate the motion primitives, the following information is required :

- 1) Robot Length
- 2) Robot Width

The motion primitives would be generated using a tool provided by Mowito.

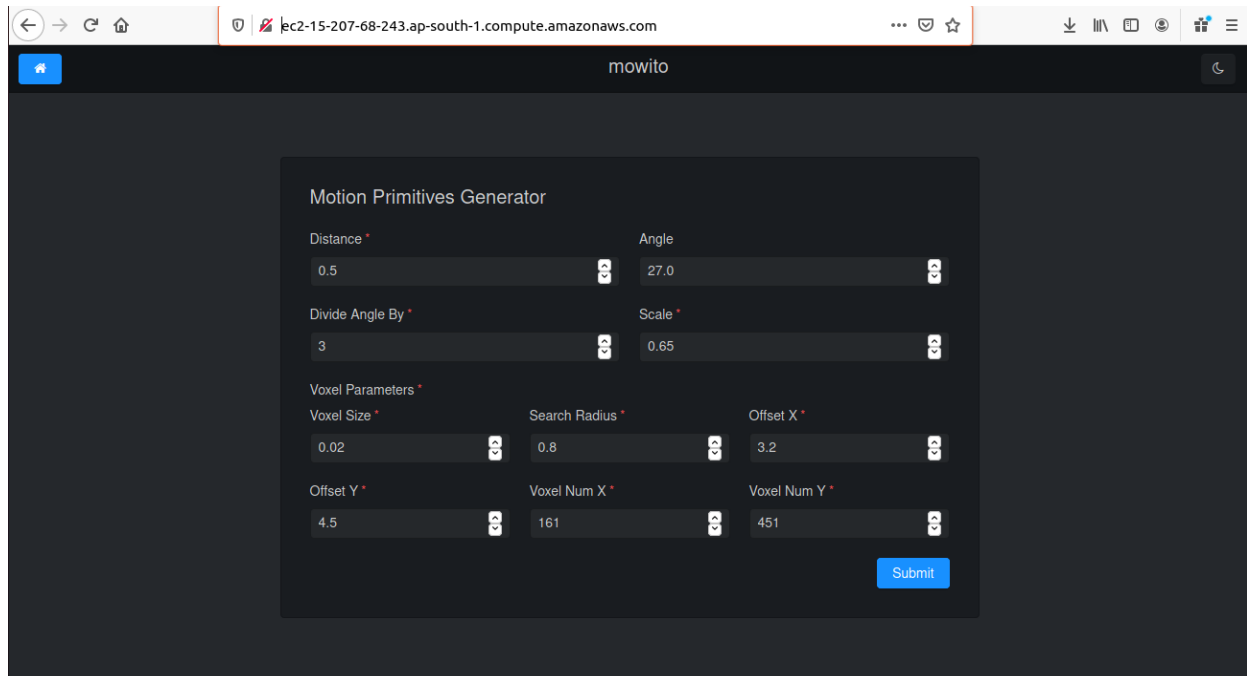
Here are the steps to generate the motion primitives:

1) Accessing the motion primitives generator web tool:

The motion primitives are generated using a web tool developed by Mowito. So in order to generate the motion primitives, the user must access the web tool.

The link to the web tool is : <http://ec2-15-207-68-243.ap-south-1.compute.amazonaws.com/>

Upon accessing the web tool, the user will land onto the following page :



2) Setting the motion primitive parameters to generate the motion primitives

In order to generate the motion primitives, certain parameters are required to be set. The parameters that are required to be set by the user are :

- a) Distance
- b) Search Radius

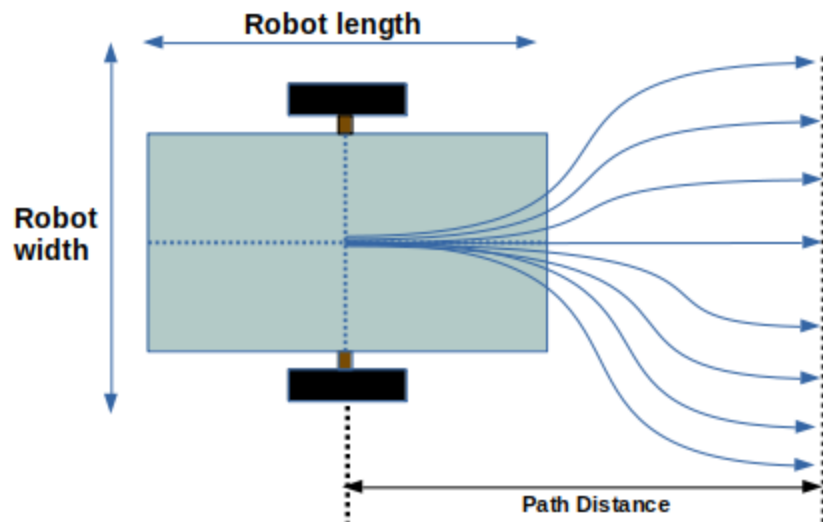
The aforementioned parameters are the **ONLY TWO PARAMETERS** that the **USER MUST SET**.

Tampering any other parameter shall generate wrong motion primitives.

The details of the two parameters are as follows :

- 1) Distance :

The distance basically indicates the length of the motion primitives from the center of the robot. The following diagram gives an illustration of the path distance.



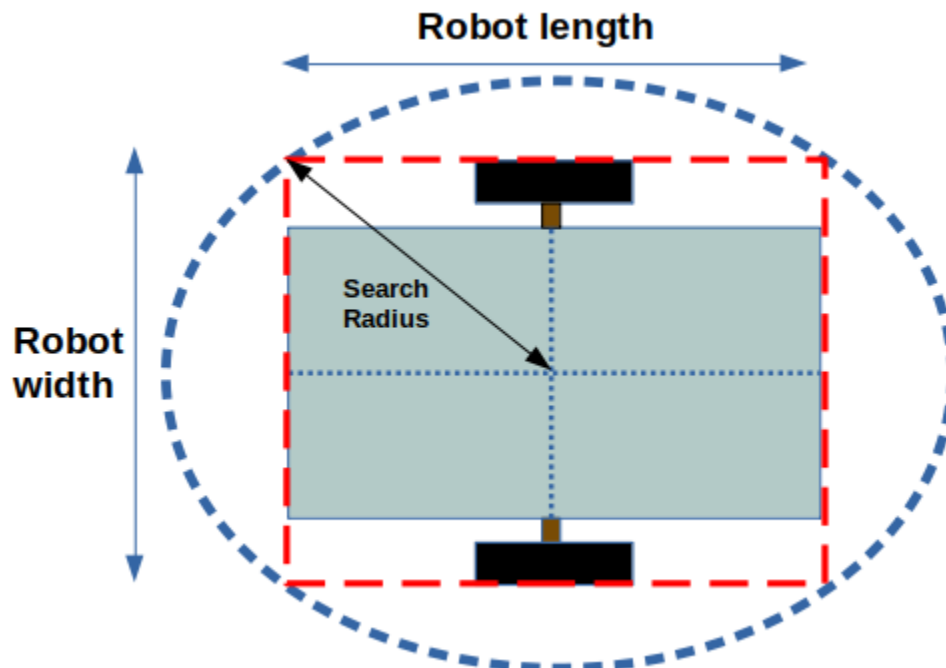
The path distance value shall remain within the following bounds :

Minimum path distance : $(\text{Robot Length})/2$

Maximum path distance : obstacle horizon distance (shall be explained in section 4)

2) Search radius

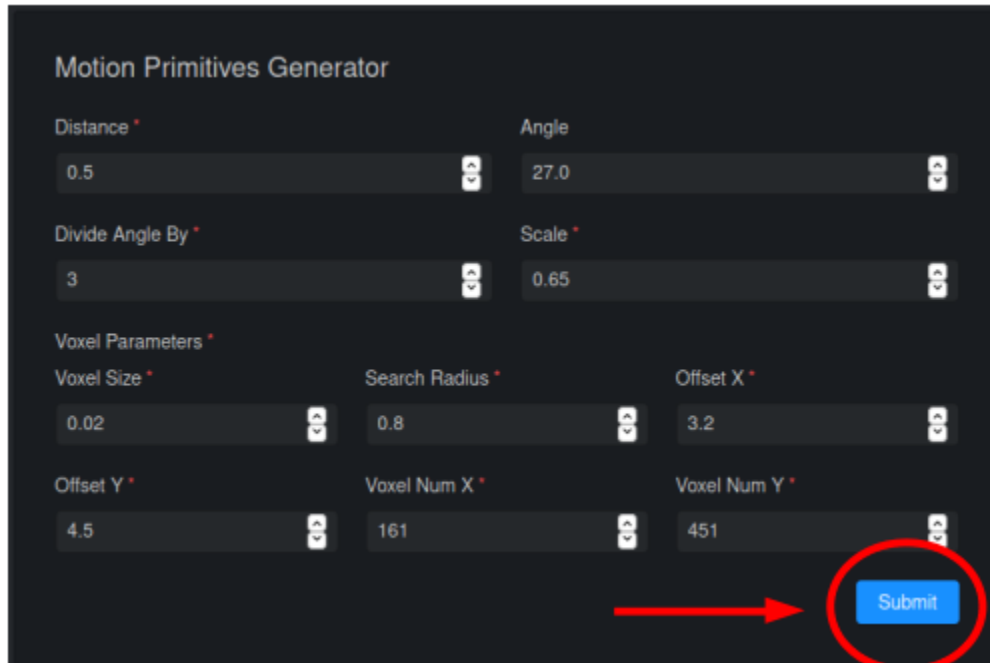
The search radius for the motion primitives shall be set a value equal to the radius of the circle that encircles the robot. The search radius parameter is illustrated in the following diagram.



Basically a higher search radius will provide a greater safety shield around the robot while the algorithm selects a path. However, a higher search radius will also lead to lesser free paths being available when the robot is confronted by an obstacle.

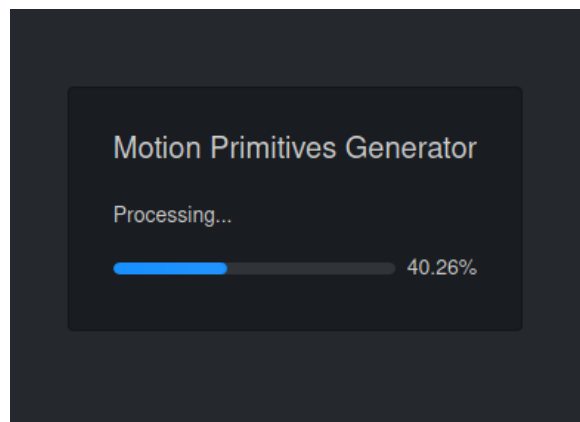
Thus it would be wise and apt to set the search radius to a value = radius of the circle encircling the robot.

3) Hit the Submit button

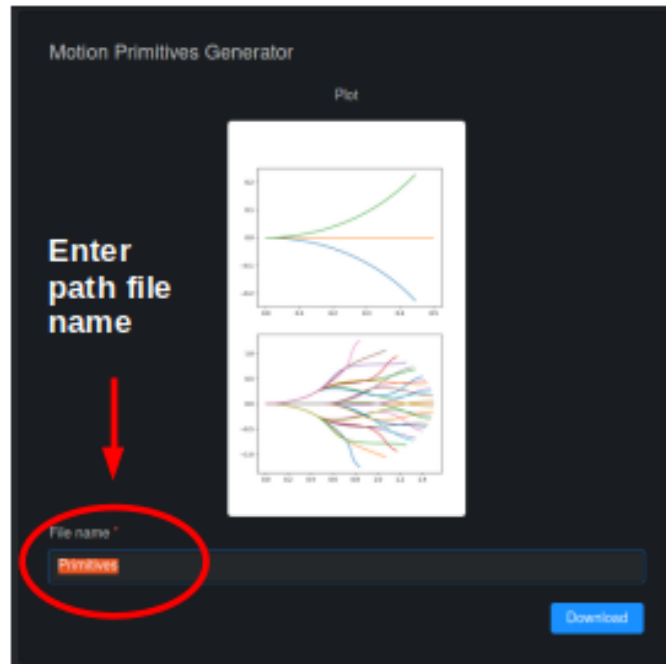


The screenshot shows the 'Motion Primitives Generator' interface. It features several input fields for parameters: Distance (0.5), Angle (27.0), Divide Angle By (3), Scale (0.65), Voxel Size (0.02), Search Radius (0.8), Offset X (3.2), Offset Y (4.5), Voxel Num X (161), and Voxel Num Y (451). A red arrow points to the 'Submit' button, which is circled in red.

4) The motion primitives will begin generation and a progress bar is displayed to track it



- 5) Upon completion, the web tool will display the motion primitives and will display the paths generated. Further the tool will prompt the user to enter the name for the paths that are generated



A general convention to name the motion primitive file is given below

mw_mprim_dxdd_rxrr

d = path distance
r = search radius

For example, the naming of the path file for motion primitives with **path distance = 1.2 m** and **search radius = 0.55 m** would be as follows :

mw_mprim_1x20_0x55

Another example, the naming of the path file for motion primitives with **path distance = 0.75 m** and **search radius = 0.65 m** would be as follows :

mw_mprim_0x75_0x65

- 6) Hit the download button, and uncompress the downloaded folder and place it in the active working directory in your robot workspace.

4. Step 2 : Configuring the MaxL parameters

The MaxL parameters are the parameters that help the algorithm decide what path to select during the robot motion when confronted by an obstacle and otherwise. This section shall describe all the parameters that the user must configure and will provide a description of these parameters and significance of these parameters.

The parameters are present in the `mw_maxl_planner.yml` file which is located in the `controller_config` folder.

The following image shows the `mw_maxl_planner.yml` file and the parameters available.

```
1 mw_maxl_planner:
2   use_laser: true
3   odomTopic: "/odom"
4   plot_path: true
5   # mprim_1 (1 m path 40 cm clearance)
6   # mprim_1_02 (20 cm clearance)
7   # mprim_07 (70 cm path 40 cm clearance)
8   # mprim_07_02 (20 cm clearance)
9   # mprim_05 (50 cm path 40 cm clearance)
10  # mprim_05_02 (20 cm clearance)
11
12  pathFolder: $(find mowito_rucha_humanoid)/mprints/mw_mprints_0x75_0x65_3
13  pathFile: "/paths"
14  autonomyMode: true
15  map_frame: "map"
16  robot_frame: "base_link"
17  velodyne_frame: "velodyne"
18  laser_frame: "base_link"
19
20  # pure pursuit
21  lookahead_goal_on_path: true
22  min_lookahead: 1.0 # 0.6 # 0.8
23  max_lookahead: 2.0 # 1.2
24  closest_point_index_search: 10
25  max_y_deviation: 1.0
26  min_radius: 0.1
27  max_radius: 1.0
28  max_omega_radius: 1.0
29  max_path_dev: 1.0
30  lookahead_point_distance: 0.1
31
32  lookahead_factor_val: 0.008 #controls the sensitivity of jumps. Lower the value, the lower the change in lookahead goal.
33  lookahead_jump_threshold: 0.05 #if the jump in lookahead values is greater than this value, it would be considered as an oscillation.
34
35
36
37  vehicleLength: 0.65 #0.45
```

```

38 vehicleWidth: 0.6 #0.5
39 sensorOffsetX: 0.1 # not having effect
40 sensorOffsetY: 0 # not having effect
41
42 maxSpeed: 0.5 #0.4/ 0.5
43 maxAccel: 0.5
44
45 yaw_gain: 0.3 # 0.6 # yaw gain used when robot is in motion
46 stop_yaw_gain: 0.6 # yaw gain used when robot is stopped/almost stopped
47 max_yaw_rate: 0.8 #0.5 # maximum yaw rate for the robot
48
49 direction_weight: 0.16 # 0.08 #0.02 # weight to change in direction
50 direction_threshold: 60 #120 # in degrees
51
52 #high accuracy multiplier for reaching the goal. Takes a value between (0, 1]
53 high_accuracy_multiplier: 0.4
54 #scoring params
55 in_place_rotation_penalty: 2.15 #higher value penalises in place rotation more
56 goal_direction_preference: 1.2 #0.80 #higher value means controller prefers paths oriented towards the goal
57 # for obstacle inflation
58 x_inflate: 0.05 #+- x_inflate / 2 meters in x-direction of path frame 0.15
59 y_inflate: 0.05 #+- y_inflate / 2 meters in y-direction of path frame 0.10
60 # enable visualisation of detailed data (pointcloud data)
61 vis_pointcloud: true
62 # the obstacle horizon - used for cropping the pointcloud
63 # it should be more than the length of the first part of the motion primitives
64 # adjacent range should be more than min path range
65 obstacle_horizon: 3.0 #2.0
66 min_path_range: 0.5
67 # scales the paths and distances. low pathScale means path elongation and vice-versa.
68 # for particular local goal, pathScale starts with initial value, finds a ptha,
69 # then value of path scale is decreased find a longer solution path. till it hits th minPathScale.
70 initial_path_scale: 1.0
71 min_path_scale: 0.75
72 path_scale_step: 0.25
73
74 use_odom_velocity: false

```

The following are the **ONLY** parameters that the user **MUST MODIFY OR TUNE**. Please **DO NOT MODIFY ANY OTHER PARAMETERS IN THE FILE**.

Sl. No.	Parameter
1	pathFolder
2	maxSpeed
3	maxAccel
4	min_lookahead
5	max_lookahead
6	lookahead_factor_val
7	lookahead_jump_threshold
8	vehicleLength
9	vehicleWidth
10	max_yaw_rate
11	in_place_rotation_penalty
12	goal_direction_preference
13	x_inflate

14	y_inflate
15	obstacle_horizon

The description and significance of these parameters is given below :

1) pathFolder:

This parameter specifies the path for the motion primitives folder where path files are located.

2) maxSpeed:

This parameter specifies the maximum speed the robot can operate at.

Units : m/s
Nominal Value : 2 m/s

Typically a higher speed would help achieve the robot reach the target point faster. On the flipside, a higher speed can induce a higher load on the motor in stopping the robot. Set this value as per the stopping capabilities of the motor in use and loads the motor can handle.

3) maxAccel:

This parameter specifies the maximum acceleration the robot can operate at.

Units : m/s²
Nominal Value : 0.5 m/s²

Typically a high acceleration value can lead to a jerky motion and can stress the motor operating the robot. Hence it is advised to have the acceleration values set on the lower side. Preferably the acceleration values can range anywhere between 0.5 m/s² and 1.5 m/s².

4) min_lookahead:

This parameter specifies the minimum lookahead point the robot must reach on the global path when the robot is in motion .

Units : m
Nominal value : (Robot length / 2) * 1.1

Typically this parameter provides the smallest lookahead target the robot must achieve on the global path during the robot motion. The actual lookahead point shall be a value that would be within the bounds of the min_lookahead and the max_lookahead.

5) max_lookahead:

This parameter specifies the maximum lookahead point the robot must reach on the global path when the robot is in motion .

Units : m

Nominal value : -

Typically this parameter provides the farthest lookahead target the robot must achieve on the global path during the robot motion. The actual lookahead point shall be a value that would be within the bounds of the min_lookahead and the max_lookahead.

It is generally advisable to have the min_lookahead fixed and vary the max_lookahead to achieve the desired motion.

Generally the user is advised to set a larger max_lookahead value in floorspaces where the floor space is large and has **dimensions greater than the max range of the LiDAR being used**.

Thus, having a robot with a higher max_lookahead in larger floor spaces will generate a smoother motion than a robot with smaller max_lookahead value.

However, in smaller floor spaces where the **dimensions of the floor space are less than the max range of the LiDAR being used**, it is advisable to have a smaller max_lookahead value and the max_lookahead in such cases can be set to the path distance parameter value specified while generating the motion primitives.

More specifically, it is advised to have a smaller max_lookahead value for robots attempting to travel through constrained door passages.

6) lookahead_factor_val:

This parameter specifies the factor by which the lookahead goal will be incremented.

Nominal value : - 0.088

The robot follows the lookahead goal. This param helps to avoid large fluctuations/jumps in the movement of the lookahead goal.

If the value of this param is decreased, the lookahead goal's movement on the global path will reduce and vice-versa. Generally it is advisable to keep a low value as this ensures smooth movement of the lookahead goal on the global path as it approaches the final pose.

<TBD>

Unsure if User should have control on deciding this value of this parameter

7) lookahead_jump_threshold:

This parameter defines the value of fluctuation/jump in lookahead goal. That is, as the lookahead goal changes, what value of change in the lookahead goal can be considered as a fluctuation/jump.

Units : m

Nominal value : - 0.05

Generally it is advised to keep a low value as this ensures smoother movement of lookahead goal on the global path

<TBD>

Unsure if User should have control on deciding this value of this parameter

8) vehicleLength:

This parameter specifies the robot length.

Unit : m

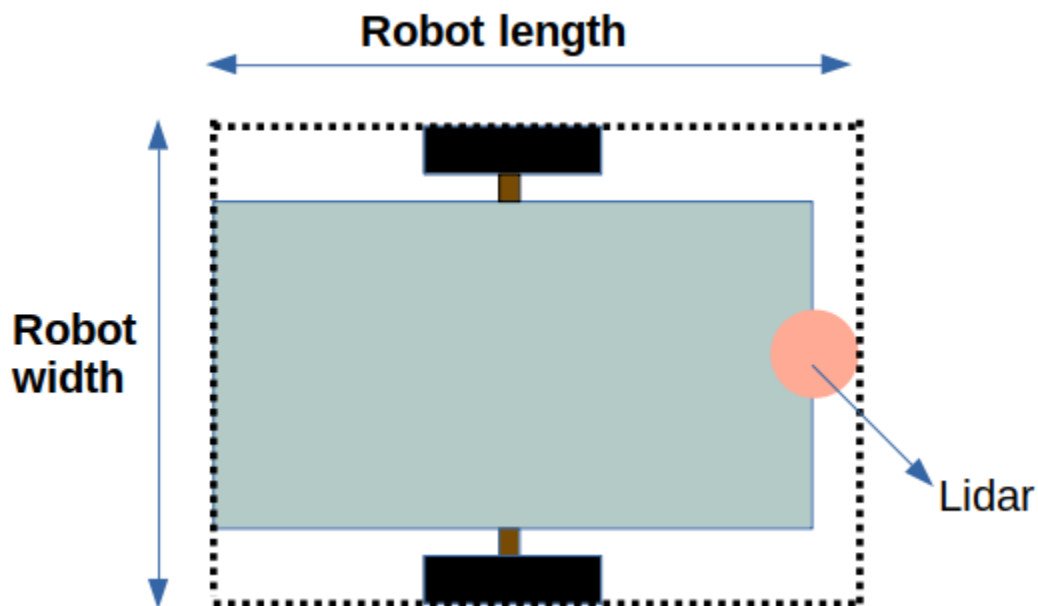
9) vehicleWidth:

This parameter specifies the robot width.

Unit : m

The robot length and width must be calculated taking into account all the auxiliary devices connected to the robot that are protruding outside the robot chassis.

The following diagram illustrates the calculation of the robot length and width.



10) max_yaw_rate:

This parameter specifies speed at which the robot performs on spot turn.

Units : rad/s

Nominal value : 0.5

Generally, it is advisable to have a low max_yaw_rate as the robot, during the path selection when confronted by an obstacle will react slower to the MaxL algorithm when the algorithm is oscillating between potential paths. This can significantly reduce the odometry and localization errors that are caused by aggressive robot oscillations.

11) in_place_rotation_penalty

This parameter specifies the weight factor to be used while scoring the different free paths available when the robot is confronted by an obstacle.

Setting a high value for this parameter reduces the in place rotations of the robot and prevents the robot from oscillating when confronted by obstacles.

Nominal Value : 2.15

12) goal_direction_preference

This parameter specifies the weight factor to be used while scoring the different free paths available when the robot is confronted by an obstacle.

Setting a high value to this parameter shall set the robot to choose a path closer to the target goal point.

Nominal value : 0.8

It is generally advisable to have a lower goal_direction_preference value in cluttered environments. This allows the robot to choose paths farther from the goal and still be successful in reaching the target goal point. A higher goal_direction_preference in a cluttered environment will prevent the robot from taking a father path and would lead to the robot not being able to reach the target goalpoint.

13) x_inflate

This parameter specifies the inflation around the obstacle in the longitudinal direction.

Basically this parameter specifies the region of influence the obstacle has for the robot to compute its local path.

Units : m

Nominal value : 0.1 m

14) y_inflate

This parameter specifies the inflation around the obstacle in the lateral direction.

Basically this parameter specifies the region of influence the obstacle has for the robot to

compute its local path.

Units : m

Nominal value : 0.1 m

15) obstacle_horizon

This parameter specifies the distance to which the robot must look in order to detect an obstacle.

Units : m

Nominal value : 1.5 m

It is advisable to have this parameter to be set to a higher value in order to have a smoother robot motion.

Further, it is **MANDATORY** to have the obstacle_horizon value **to be greater than the path distance of the motion primitives.**