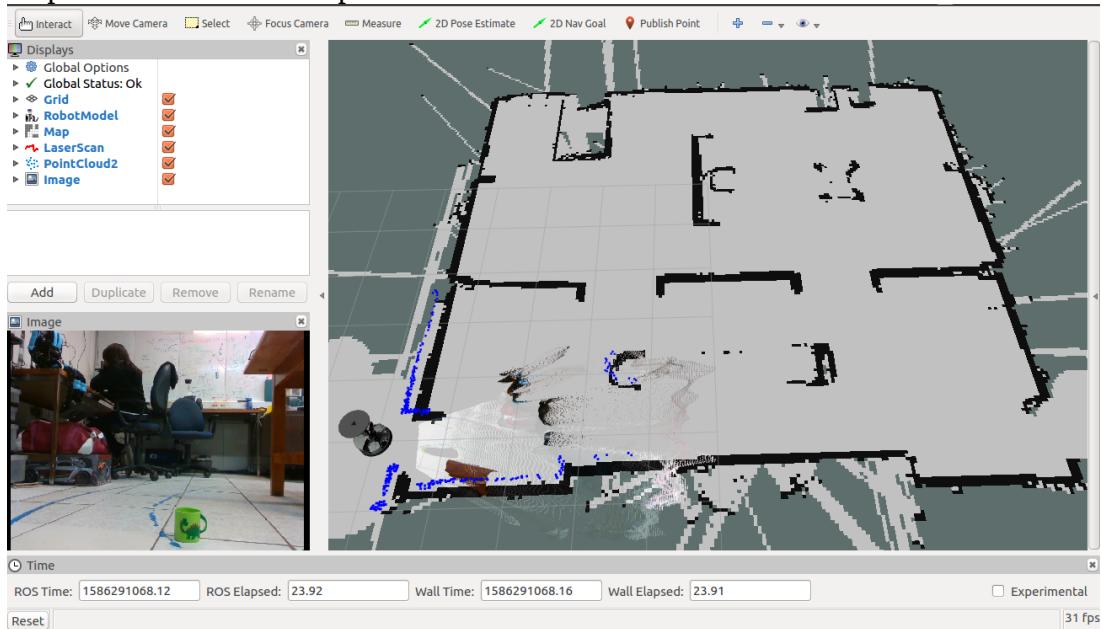
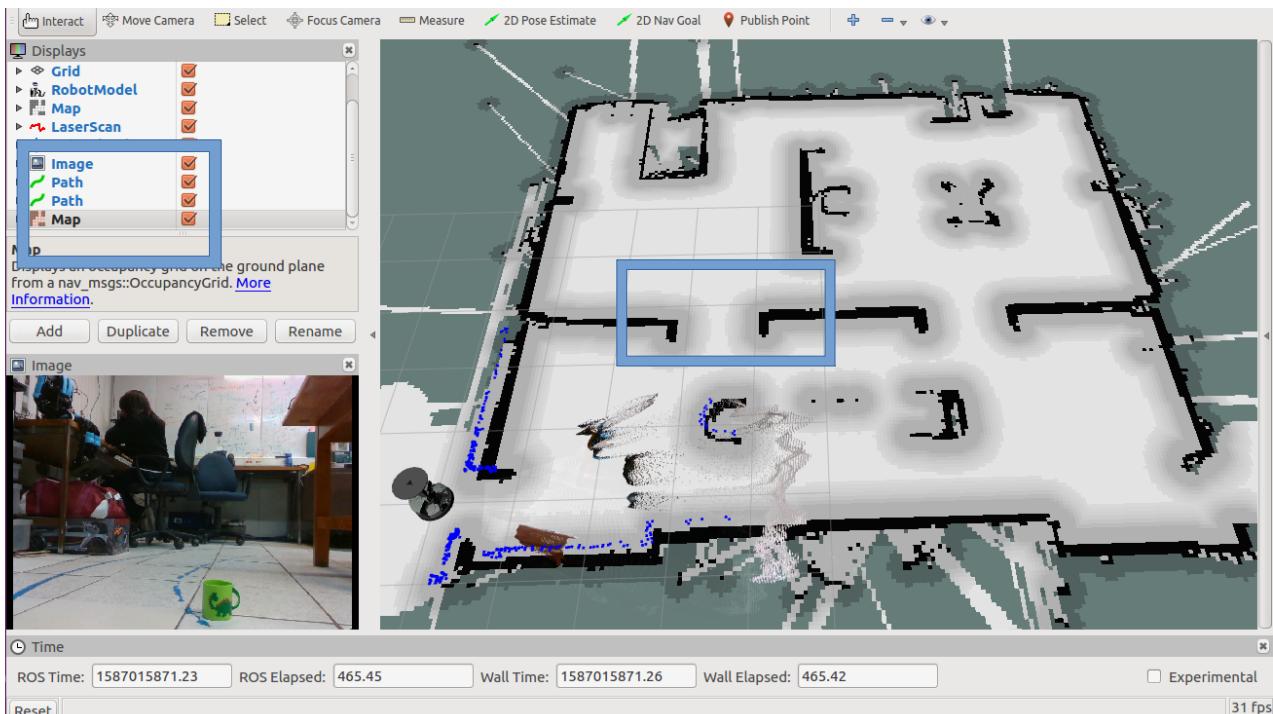


Práctica 3: Uso del navigation stack para navegación en 2D.

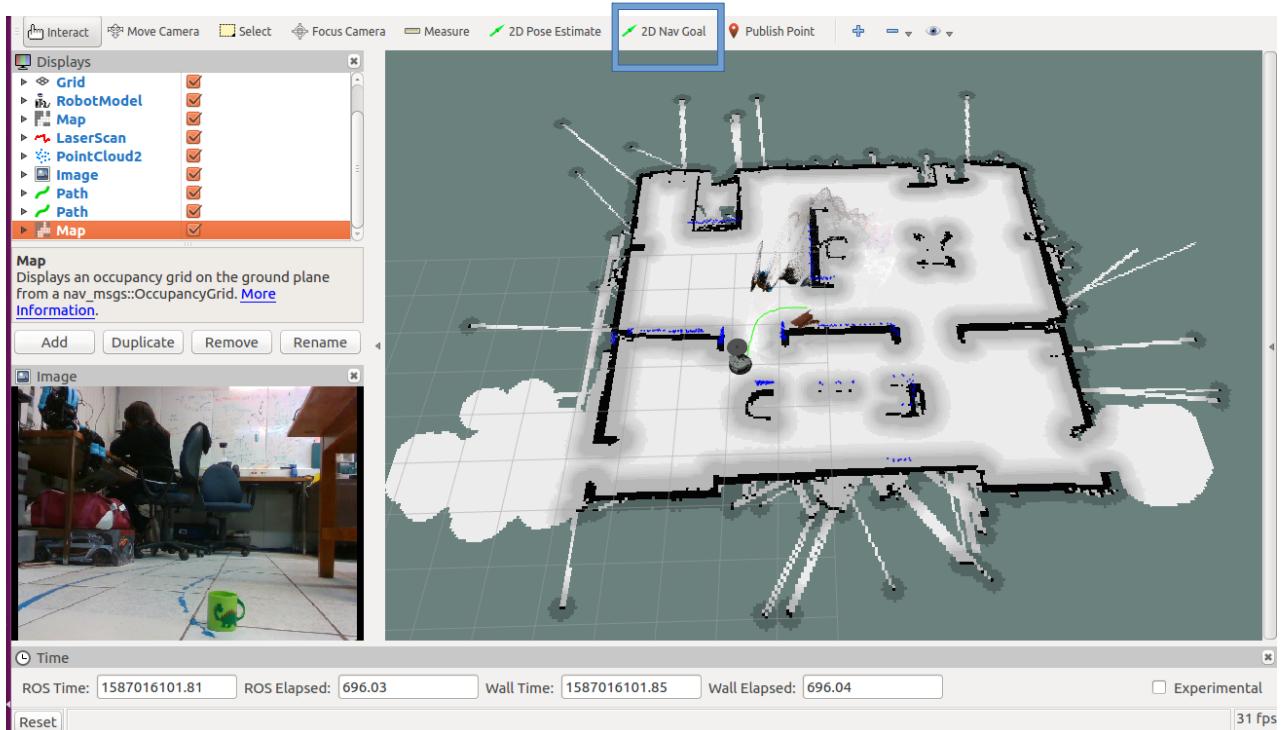
-Captura de pantalla al inicio de la práctica.



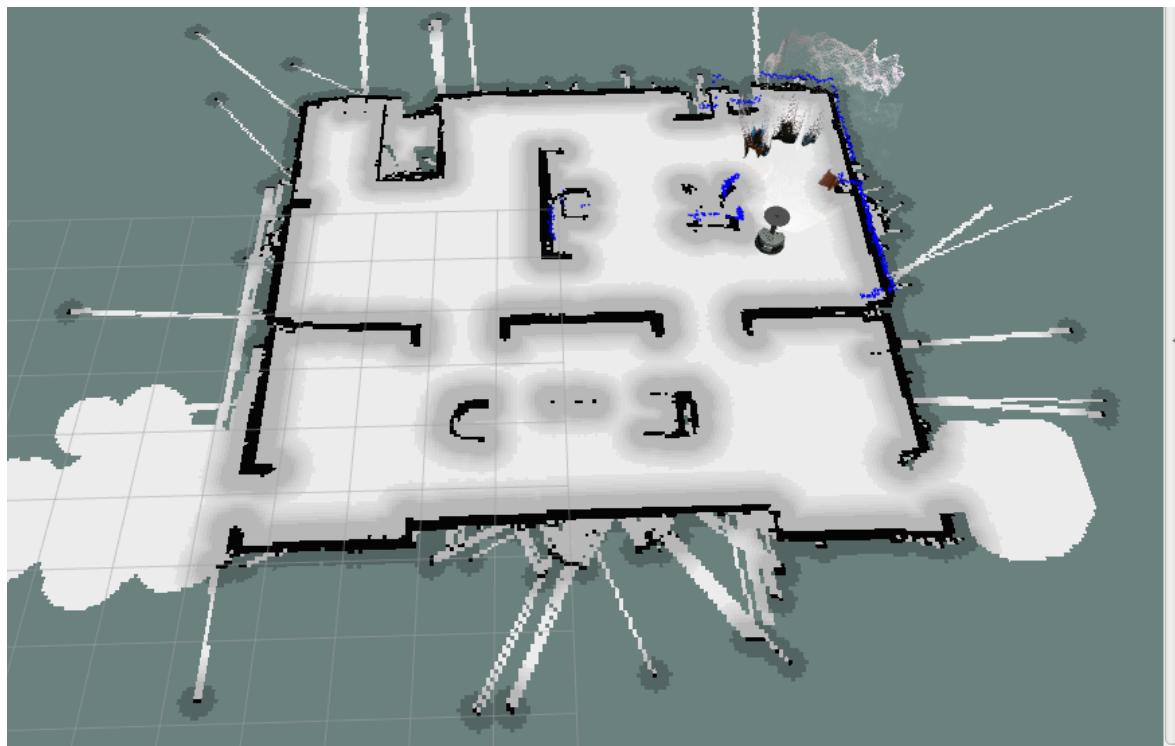
-Captura de pantalla (varias) donde se observe la ruta planeada y el movimiento del robot.



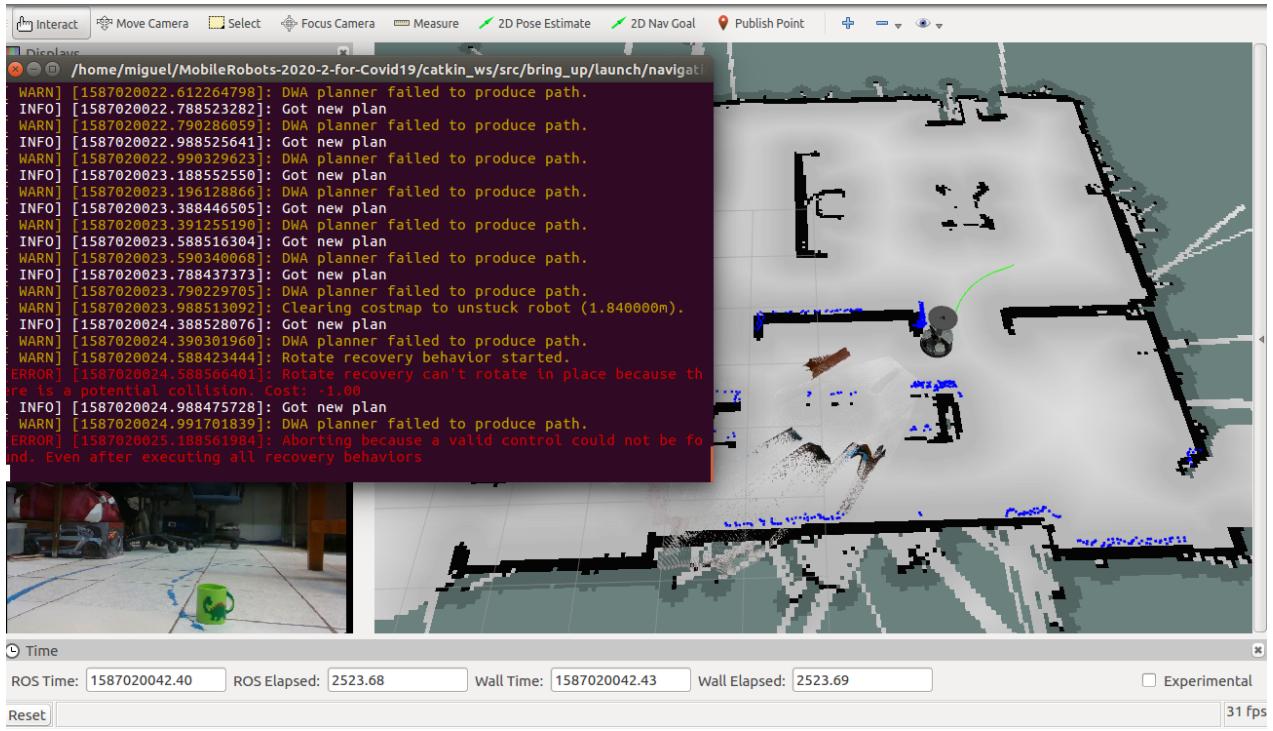
En la imagen anterior se muestra lo que sucede cuando se agregan los tópicos de “global_plan”, “local_plan” y el de “costmap”. La gran diferencia que se puede notar es que alrededor de los obstáculos aparece una sombra en color gris.



Para esta imagen se activa la opción “2D Nav Goal”, donde se selecciona una ubicación del mapa, después el robot empieza su camino hacia dicho punto, donde se puede notar un trazo en color verde, el cual el robot va siguiendo.



En esta imagen, el robot ya llegó al punto indicado por el “2D Nav Goal”, durante su recorrido la cámara le ayuda a llegar a la meta evitando los obstáculos.



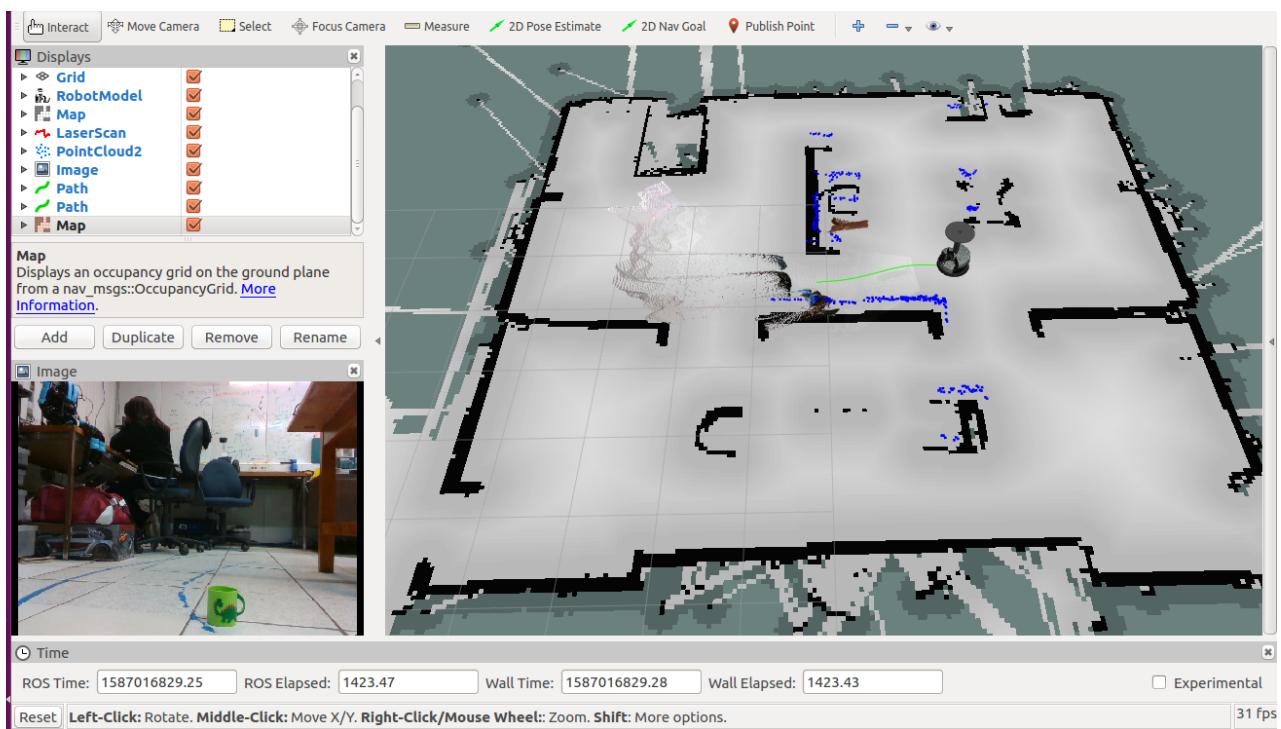
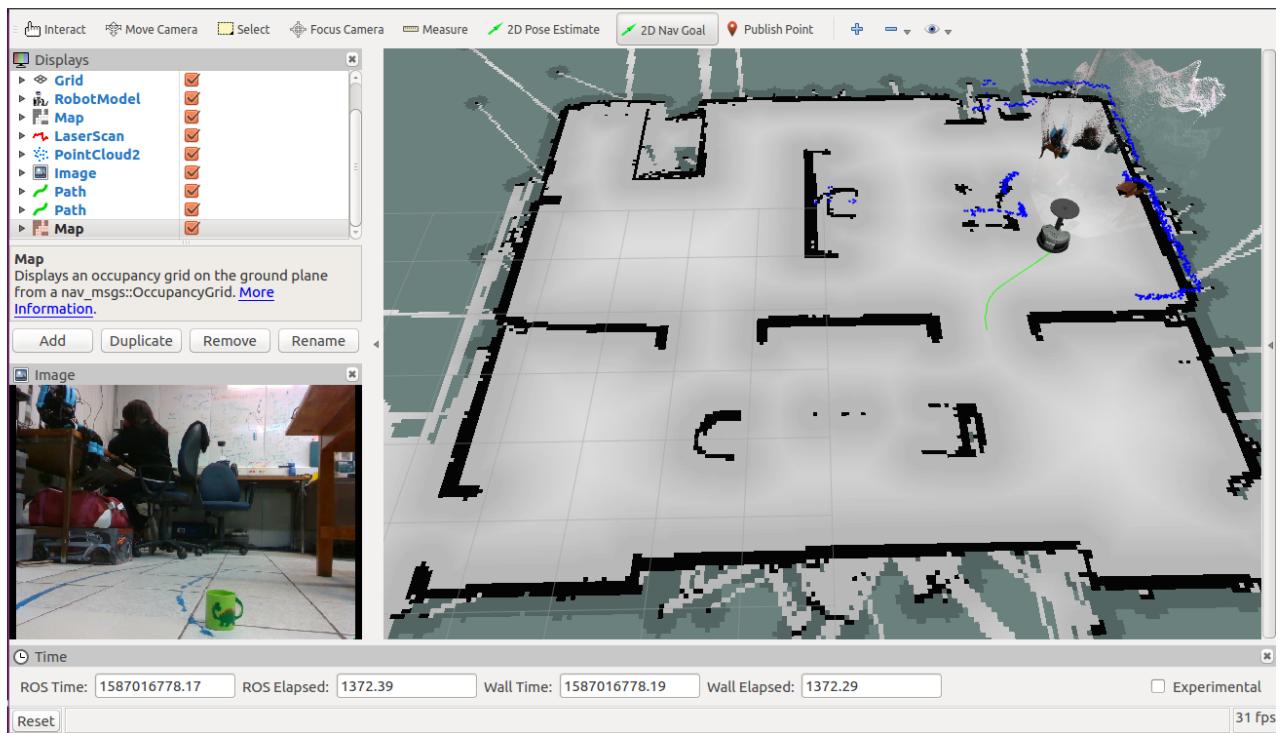
Un punto muy importante a destacar es que en algunas rutas dadas, el robot no logra llegar al punto deseado, se traba y se detiene el robot, mientras que en la terminal, aparece que se ha cometido un error.

```

data_type: LaserScan
topic: scan
marking: true
clearing: true
min_obstacle_height: 0.25
max_obstacle_height: 0.35
bump:
  data_type: PointCloud2
  topic: mobile_base/sensors/bumper_pointcloud
  marking: true
  clearing: false
  min_obstacle_height: 0.0
  max_obstacle_height: 0.15
# for debugging only, let's you see the entire voxel grid
#cost_scaling_factor and inflation_radius were now moved to the inflation_layer ns
inflation_layer:
  enabled: true
  cost_scaling_factor: 1.0 # exponential rate at which the obstacle cost drops off (default: 10)
  inflation_radius: 2.5 # max. distance from an obstacle at which costs are incurred for planning paths.
static_layer:
  enabled: true

```

Configurando los valores de “cost_scaling_factor” a 1.0 y el valor de “inflation_radius” a 2.5 se puede notar que la sombra en color gris alrededor de los obstáculos, se ha modificado, pasando a un color más claro en comparación con el mapa antes de las modificaciones, como se muestra en la siguiente imagen.



```

Robot Configuration Parameters - Kobuki
max_vel_x: 2.0 # 0.55
min_vel_x: 0.0

max_vel_y: 0.0 # diff drive robot
min_vel_y: 0.0 # diff drive robot

max_trans_vel: 0.5 # choose slightly less than the base's capability
min_trans_vel: 0.1 # this is the min trans velocity when there is negligible rotational velocity

# Robot Configuration Parameters - Kobuki
max_vel_x: 2.0 # 0.55
min_vel_x: 0.0

max_vel_y: 0.0 # diff drive robot
min_vel_y: 0.0 # diff drive robot

max_trans_vel: 2.0 # choose slightly less than the base's capability
min_trans_vel: 0.1 # this is the min trans velocity when there is negligible rotational velocity
trans_stopped_vel: 0.1

# Warning!
#   do not set min_trans_vel to 0.0 otherwise dwa will always think translational velocities are non-negligible and small in place rotational velocities will be created.

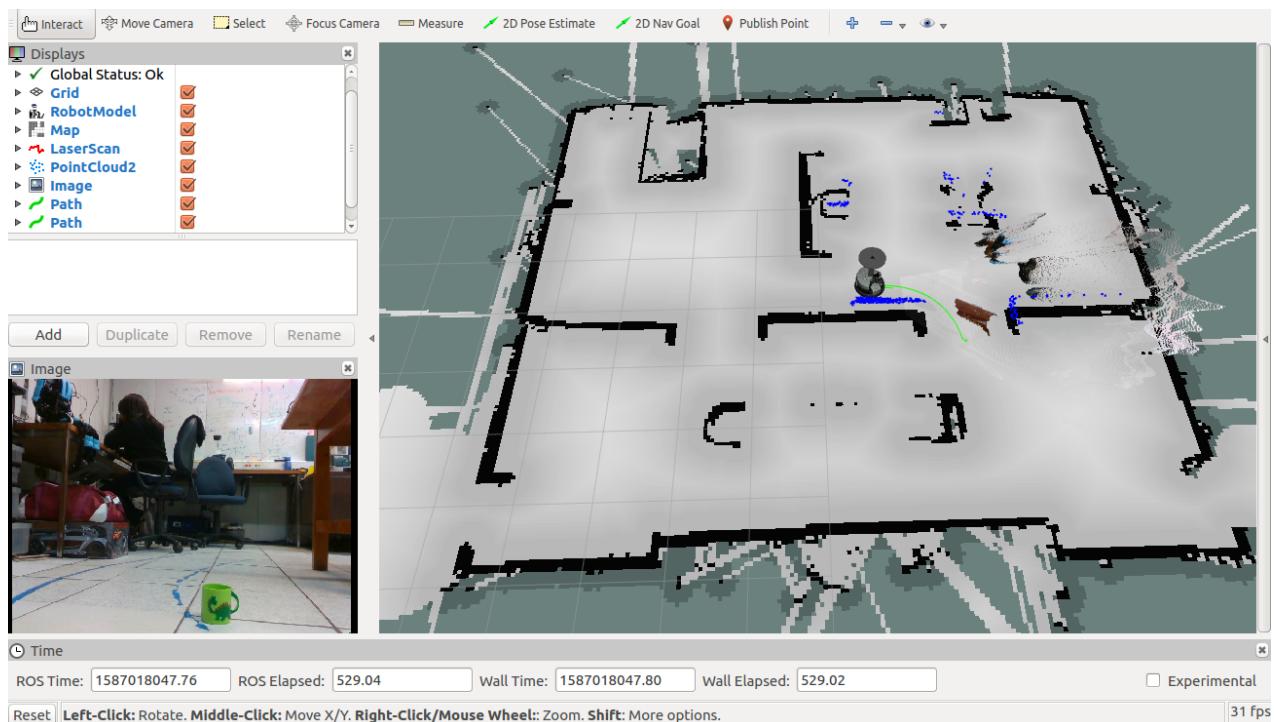
max_rot_vel: 5.0 # choose slightly less than the base's capability

# Warning!
#   do not set min_trans_vel to 0.0 otherwise dwa will always think translation velocities are non-negligible and small in place rotational velocities will be created

max_rot_vel: 5.0 # choose slightly less than the base's capability
min_rot_vel: 0.4 # this is the min angular velocity when there is negligible rotational velocity
rot_stopped_vel: 0.4

acc_lim_x: 2.0 # maximum is theoretically 2.0, but we
acc_lim_theta: 2.0
acc_lim_y: 0.0      # diff drive robot

```



Configurando los valores de “max_vel_x”, “max_trans_vel” y “acc_lim_x” a 2.0, se puede notar que el robot llega al punto meta con mayor velocidad en comparación a la simulación llevada a cabo antes de modificar los valores, de igual forma la velocidad con que gira la cámara para evadir los obstáculos es mayor, esto ayuda a que el robot presente su trayectoria de forma más satisfactoria con la velocidad elevada.

