移动机器人轨迹规划及运动控制

Michael Gao

实验目标

- 填补global planner节点,将所规划路径发布使其能在rviz上显示
- 结合laser信息,填补local planner节点,使其可以避障

实验步骤

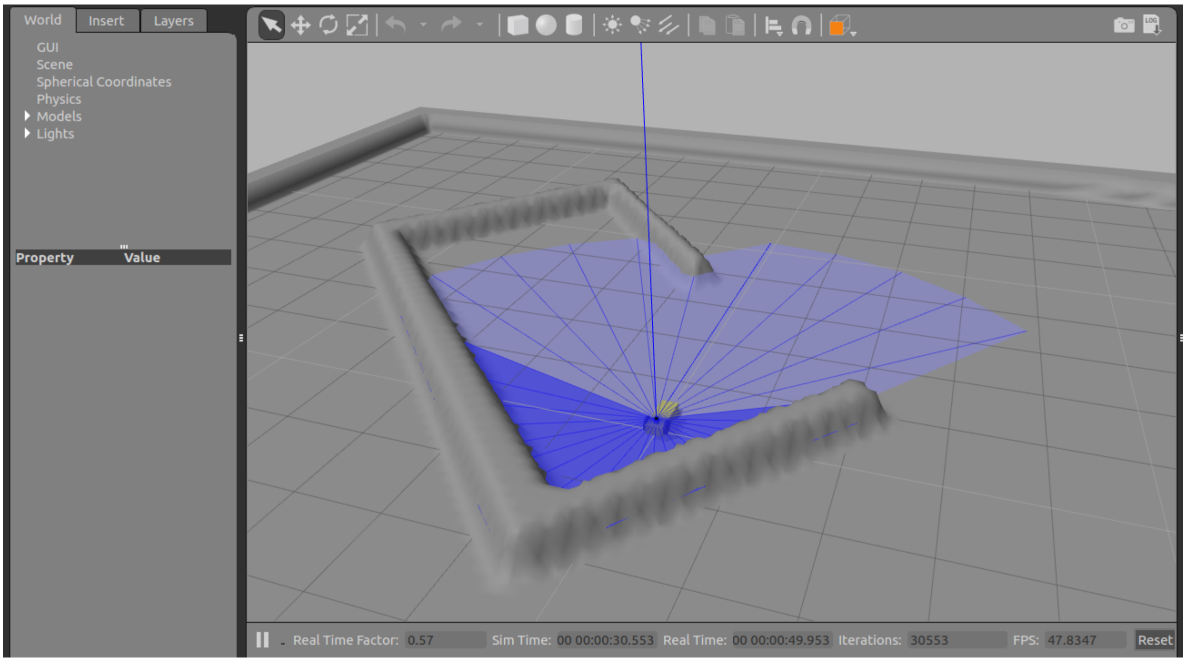
(1) Laser 配置修改

由于电脑GPU配置不支持`libgazebo_ros_gup_laser.so`插件，故使用CPU版本，将`urdf/course_agv.gazebo`文件中的激光雷达配置部分修改如下：

```
<!-- hokuyo -->
<gazebo reference="course_agv__hokuyo__link">
    <sensor type="ray" name="head_hokuyo_sensor">
        <pose>0 0 0 0 0 0</pose>
        <visualize>true</visualize>
        <update_rate>3</update_rate>
        <ray>
            <scan>
                <horizontal>
                    <samples>30</samples>
                    <resolution>1</resolution>
                    <min_angle>-3.14159</min_angle>
                    <max_angle>3.14159</max_angle>
                </horizontal>
            </scan>
            <range>
                <min>0.10</min>
                <max>6.0</max>
                <resolution>0.01</resolution>
            </range>
            <noise>
                <type>gaussian</type>
                <mean>0.0</mean>
                <stddev>0.01</stddev>
            </noise>
        </ray>
        <plugin name="gazebo_ros_head_hokuyo_controller" filename="libgazebo_ros_laser.so">
            <topicName>/course_agv/laser/scan</topicName>
            <frameName>course_agv__hokuyo__link</frameName>
        </plugin>
    </sensor>
</gazebo>
```

将laser插件更改为了CPU版本，并降低了激光束范围和分辨率，否则CPU无法负载.

修改后的激光范围如图所示：



(2) Weighted A-star

补全`a_star.py`文件中的部分，A*代码结构如下：

- Maintain a priority queue to store all the nodes to be expanded
- The heuristic function h(n) for all nodes are pre-defined
- The priority queue is initialized with the start state X_S
- Assign $g(X_S)=0$, and g(n)=infinite for all other nodes in the graph
- Loop
  - If the queue is empty, return FALSE; break;
  - Remove the node “n” with the lowest f(n)=g(n)+h(n) from the priority queue
  - Mark node “n” as expanded
  - If the node “n” is the goal state, return TRUE; break;
  - For all unexpanded neighbors “m” of node “n”
    - If g(m) = infinite
      - Push node “m” into the queue
    - If $g(m) > g(n) + C_{nm}$
      - $g(m)= g(n) + C_{nm}$
  - end
- End Loop

为了方便调整A*中已经走过路径cost：g(x) 和 heuristic-cost：h(x) 权重，在`AStarPlanner`类中加入两个变量：

```
self.g_w = 1
self.h_w = 0.5
```

补全部分如下：

```
while 1:
```

```python

    # 1. Find the smallest in open set & Remove it
    min_f = 1e8
    for tinx, tnode in open_set.items():
        if tnode.f < min_f:
            min_f = tnode.f
            min_inx = tinx

    min_node = open_set.pop(min_inx)

    # 2. Add it to closed set
    closed_set[self.calc_grid_index(min_node)] = min_node

    # 3. Is Goal?
    if min_node.x == ngoal.x and min_node.y == ngoal.y:
        print("Search arrive goal!")
        ngoal = min_node
        break

    # 4. Handle all neighbor - Relaxation
    for motion_i in self.motion:
        tnode = self.Node(min_node.x + motion_i[0], min_node.y + motion_i[1], 0.0, 0.0, self.calc_grid_index(min_node))
        tnode.g = min_node.g + motion_i[2]
        tnode.f = self.g_w * tnode.g + self.h_w *self.calc_dis(tnode, ngoal)
        tnode_inx = self.calc_grid_index(tnode)

        # check obstacle & if in closed set
        if self.verify_node(tnode) and not(closed_set.has_key(tnode_inx)):
            # if not in open set -> it's a new point
            if not open_set.has_key(tnode_inx):
                # add it to openset
                open_set[tnode_inx] = tnode

            # if in open set
            else:
                if open_set[tnode_inx].g > tnode.g:
                    open_set.pop(tnode_inx)
                    open_set[tnode_inx] = tnode
```

(3) DWA 动态窗口法

机器人运动模型

$$
\begin{aligned}
&x=x+v\Delta t\cos(\theta_t)\\
&y=y+v\Delta t\sin(\theta_t)\\
&\theta_t=\theta_t+w\Delta t
\end{aligned}
$$

速度采样

范围限制：

$$V_m=\{v\in[v_{\min},v_{\max}],w\in[w_{\min},w_{\max}]\}$$

$$V_d = \left\{ (v, \omega) \middle| \begin{array}{l} v \in [v_c - \dot{v}_b \Delta t, v_c + \dot{v}_a \Delta t] \wedge \\ \omega \in [\omega_c - \dot{\omega}_b \Delta t, \omega_c + \dot{\omega}_a \Delta t] \end{array} \right\}$$

在此范围内以一定分辨率进行速度采样.

评价函数

$$G(v,\omega) = \sigma(\alpha \cdot \mathrm{heading}(v,\omega) + \beta \cdot \mathrm{dist}(v,\omega) + \gamma \cdot \mathrm{velocity}(v,\omega))$$

<u>方位角评价</u>:

计算达到轨迹末端时的朝向与目标之间的角度差值

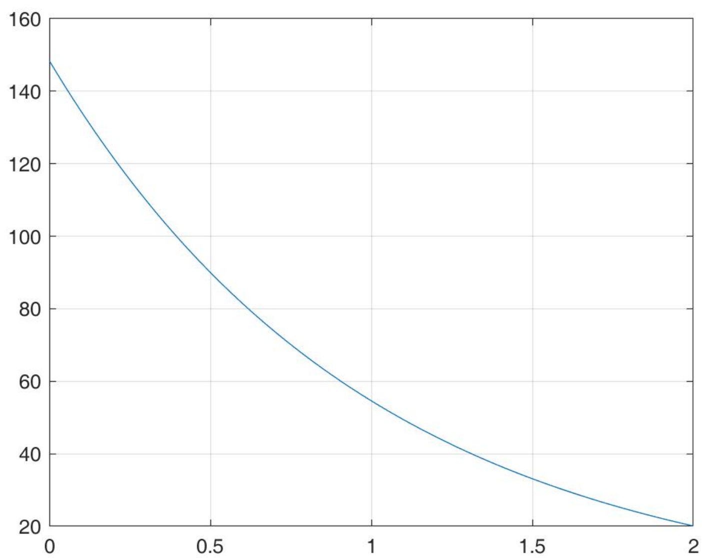
$$Cost = |\theta_{traj} - \theta_{aim}|$$

```python
def calc_to_goal_cost(trajectory, goal):
    """
        calc to goal cost with angle difference
    """
    goal_angle = math.atan2(goal[1], goal[0])

    if trajectory[-1,3] >= 0:
        cost = math.fabs(goal_angle - trajectory[-1, 2])
    else:
        diff = math.fabs(goal_angle - (trajectory[-1, 2]+math.pi))
        if diff > math.pi:
            diff = 2*math.pi - diff
        cost = math.fabs(diff)

    return cost
```

<u>障碍物距离评价</u>:

当障碍物距离< 1时，有如下形式的障碍物cost:

$$Cost = e^{-d+5}$$




```python
def calc_obstacle_cost(trajectory, ob, config):
    """
        calc obstacle cost inf: collision
```

```python
    """
    if ob.size == 0:
        cost = 0
        return cost

    ox = ob[:, 0]
    oy = ob[:, 1]
    dx = trajectory[:, 0] - ox[:, None]
    dy = trajectory[:, 1] - oy[:, None]
    r = np.hypot(dx, dy)
    cost = 0
    mind = r.min()

    if mind < 1:
        cost = math.exp(-(mind-1))

    return cost
```

速度评价:

$$Cost = Maxv - |\bar{v}|$$

```python
v_mean = np.mean(trajectory[:,3])
v_cost = 0.8 - math.fabs(v_mean)
```

结果:

(见 `vedio/dwa.mp4`)

从视频和命令行输出显示可以看到，小车速度已达上限，在平滑路段已以最大速度行驶，整体路径跟踪性能较好.

图中添加了2个动态障碍物，小车具有动态避障能力.