占用栅格地图在线构建

Michael Gao

实验目标

基于在线SLAM轨迹，以及Laser Scan，构建Grid Map.

实验内容

1.TF与Laser Scan听取

监听ekf节点的结果，获得机器人坐标：

```
/* 1.监听机器人局部坐标系 */
try{
    listener.lookupTransform(world_frame, sensor_frame,
                                    ros::Time(0), transform);
}
catch (tf::TransformException &ex) {
    ROS_ERROR("%s",ex.what());
    ros::Duration(1.0).sleep();
    return;
}
```

2.二值贝叶斯滤波

占用栅格地图将空间划分为有限多个栅格单元，此处采用**对数占用概率**表示该栅格有障碍的概率：

$$l = log\frac{p}{1-p}$$

其中，p是该栅格有障碍物的概率.

此处采用对数占用概率将原本取值在0-1的有障碍物概率映射到了$[-\infty, \infty]$.

用二值贝叶斯滤波的方法对对数占用概率进行更新：

$$\textbf{Algorithm binary\_Bayes\_filter}$$

$$l_t = l_{t-1} + \log\frac{p(x|z_t)}{1-p(x|z_t)} - \log\frac{p(x)}{1-p(x)}$$
$$\text{return } l_t$$

在当前的情境下，每获得一帧新的观测雷达数据，就对所有栅格单位进行遍历：

- 计算当前帧激光数据下的对数占用概率
- 用贝叶斯二值滤波更新

算法伪代码如下：

Algorithm occupancy_grid_mapping

for all cells m_i do

  if m_i in perceptual field of z_t

    $l_{t,i} = l_{t-1,i} + \text{inverse\_sensor\_model}(\mathrm{m}_i, x_t, z_t) - l_0$

  else

    $l_{t,i} = l_{t-1,i}$

  endif

endfor

return $\{l_{t,i}\}$

完整代码:

```cpp
/* 2.遍历每个栅格，进行二值贝叶斯滤波 */
for(int i = 0; i < map_height; i++){
    for(int j = 0; j < map_width; j++){
        // 2.1 将当前栅格中心坐标变换到机器人局部极坐标系下，寻找离其最近的一束激光
        Vector2d tgrid;
        tgrid << i*map_res + map_res/2.0, j*map_res + map_res/2.0;

        if(sqrt(pow(tgrid(0) - roboPose(0), 2) + pow(tgrid(1) - roboPose(1), 2)) > sensor_range + occu_dis_thres)
            grid_map.data[j*grid_map.info.width+i] = 100.0 * (1 - 1.0 / (1 + exp(map_log(i, j))));
            continue; //不在激光范围内

        Vector2d rgrid;
        rgrid = R.inverse()*(tgrid - roboPose);
        double rangle = angleNorm(std::atan2(rgrid(1), rgrid(0)));
        int inx = rangle /input.angle_increment + laser_num/2;

        if(input.ranges[inx] > input.ranges[inx+1]){
            inx++;
        }
        range = input.ranges[inx]+0.1;

        // 2.2 计算栅格占用概率
        map_log(i, j) = map_log(i, j) + inverseSensorModel(tgrid, roboPose, range) - lprob_init;

        // 2.3 计算grid_map值 0-100
        grid_map.data[j*grid_map.info.width+i] = 100.0 * (1 - 1.0 / (1 + exp(map_log(i, j))));
    }
}
```

反演测量模型

即计算当前帧激光数据下的每个栅格对数占用概率.

在机器人局部极坐标系下分析:

变量:

`dis`: 该栅格中心与机器人原点距离

`occu_dis_thres`: 判定为有障碍物的半径

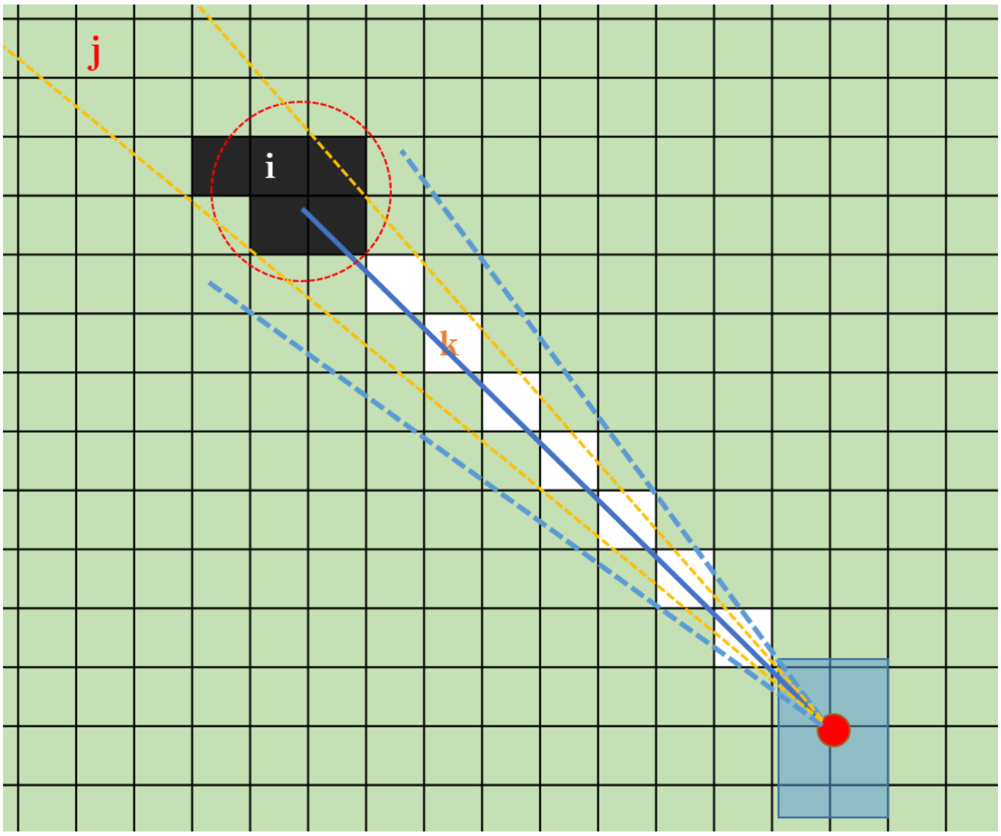
`range`：激光束长度

步骤：

- 根据角度找到：距离当前栅格角度最近的激光束
- 根据该栅格位置和对应激光点位置赋予该栅格占用概率，分为3种情况赋予概率：
  - a.栅格位置超出该激光长度 `range + occu_dis_thres` 距离 -> 不更新
  - b.栅格位置距离激光点小于 `occu_dis_thres` -> 有障碍
  - c.非a,b情况 -> 无障碍

如下图所示，红点为机器人原点，蓝色为激光束，黄色为两个激光束之间的角度等分线，红圈为 `occu_dis_thres` 判定有障碍物半径；

栅格j对应情况a，栅格i对应情况b，栅格k对应情况c：



具体代码如下：

```
double mapping::inverseSensorModel(Vector2d tgrid, Vector2d roboPose, double range)
{
    double dis = sqrt(pow(tgrid(0) - roboPose(0), 2) + pow(tgrid(1) - roboPose(1), 2));
    if(dis > min(sensor_range, range + occu_dis_thres)){
        return lprob_init;
    }
    if(range < sensor_range && abs(dis - range) < occu_dis_thres){
        return lprob_occu;
    }
    if(range < sensor_range){
        return lprob_free;
```

```
    }
    return lprob_init;
}
```

3.grid_map二值化

设置一个人为阈值，根据贝叶斯滤波的结果构建二值化地图：

```
if(grid_map.data[j*grid_map.info.width+i] >=  prob_thres*100){
    grid_map.data[j*grid_map.info.width+i] = 100;
}
else{
    grid_map.data[j*grid_map.info.width+i] = 0;
}
```

实验结果

视频结果见`vedio/res.mp4`，视频中展示了：

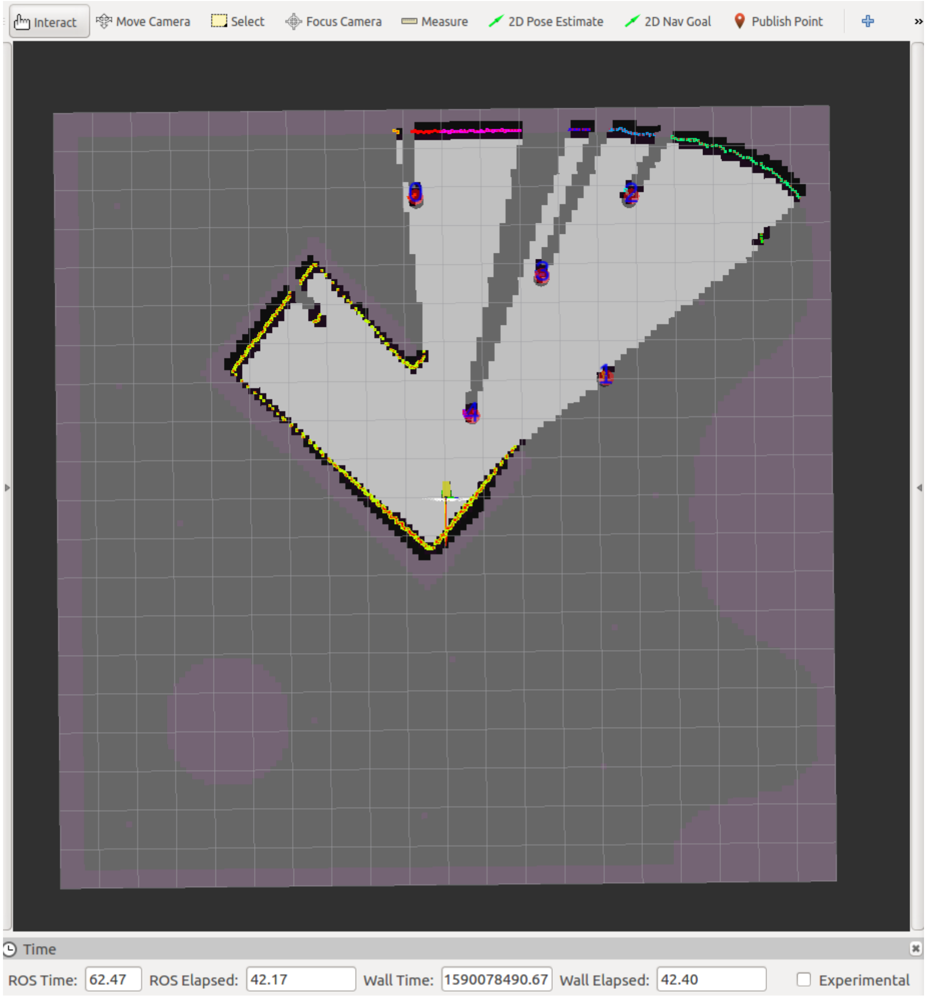
- 动态障碍物
- 二值化地图显示
- 贝叶斯地图显示

视频播放速度为8倍速.

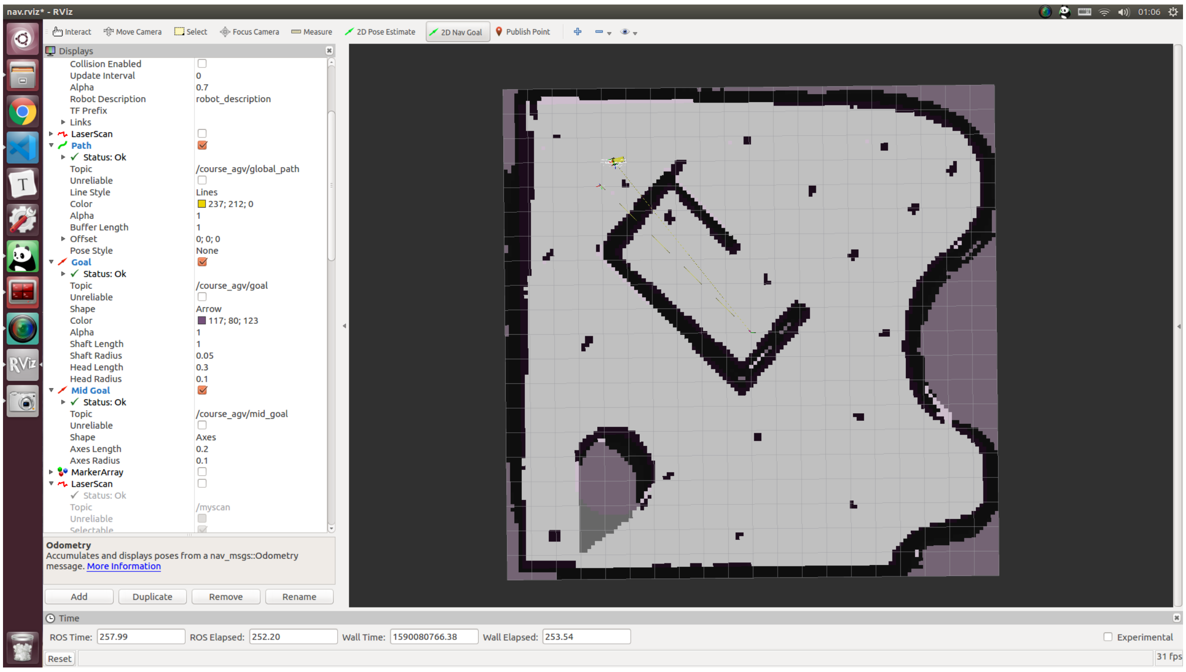
运行方法：

```
roslaunch course_agv_slam_task icp_all.launch
roslaunch course_agv_slam_task ekf.launch
roslaunch course_agv_nav nav_for_all.launch
roslaunch course_agv_slam_task mapping.launch
rosrun course_agv_control keyboard_velocity.py
```

第一帧截图：



最终建图结果：



误识率

定义误识率：

$$err = Num(right\_grid)/Num(all\_grid)$$

误识率为所有二值化后的栅格中误识别的比例，注意，由于边缘墙太厚(即上图中边缘粉色部分)，使得误识率不能降至0，误识率图像随建图变化如下：