通过pibot_simulator我们看下地图与坐标的关系以及如何获取机器人在地图中的实时坐标

1.PIBOT模拟器

1.1模拟器使用

启动模拟器 roslaunch pibot_simulator nav.launch 启动Rviz roslaunch pibot_navigation view_nav.launch

1.2地图

```
roscd pibot_simulator/launch
cat nav.launch
```

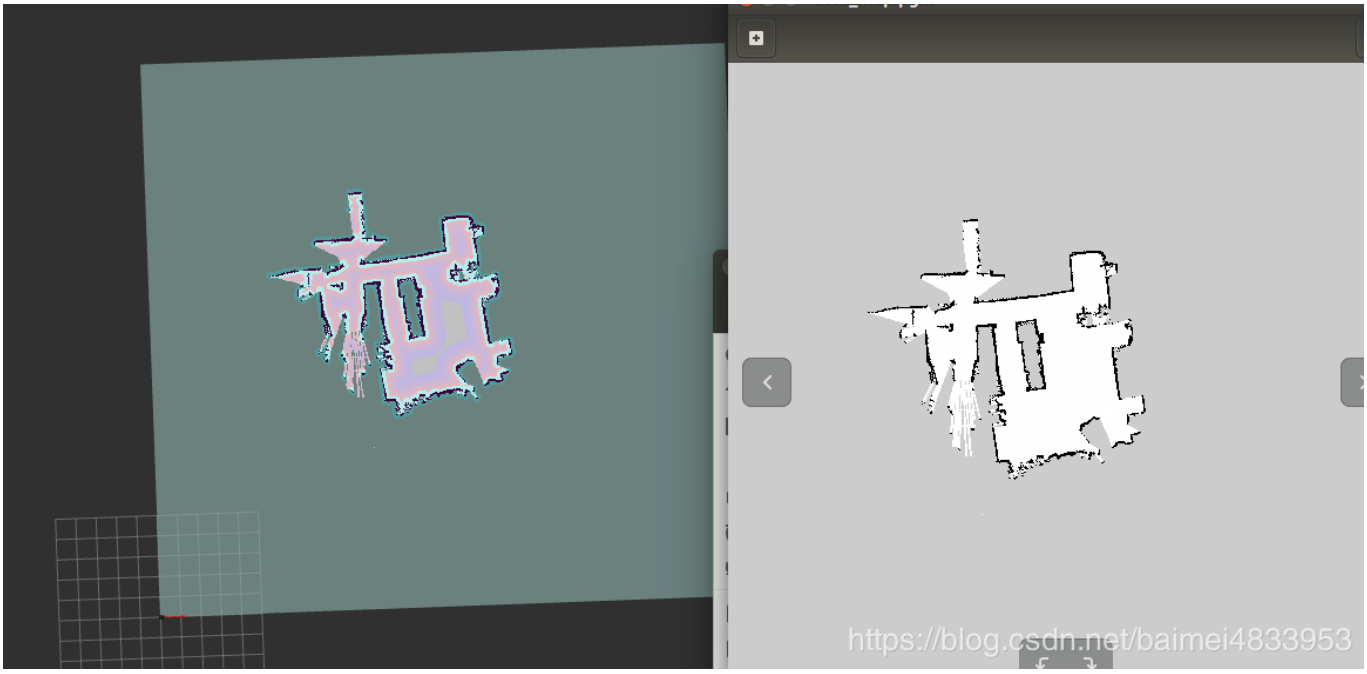
可以看到加载的是pibot_simulator/maps/test_map.yaml

```
image: test_map.pgm
resolution: 0.050000
origin: [-13.800000, -12.200000, 0.000000]
negate: 0
occupied_thresh: 0.9
free_thresh: 0.196
```

test_map.yaml 指定了使用的地图文件

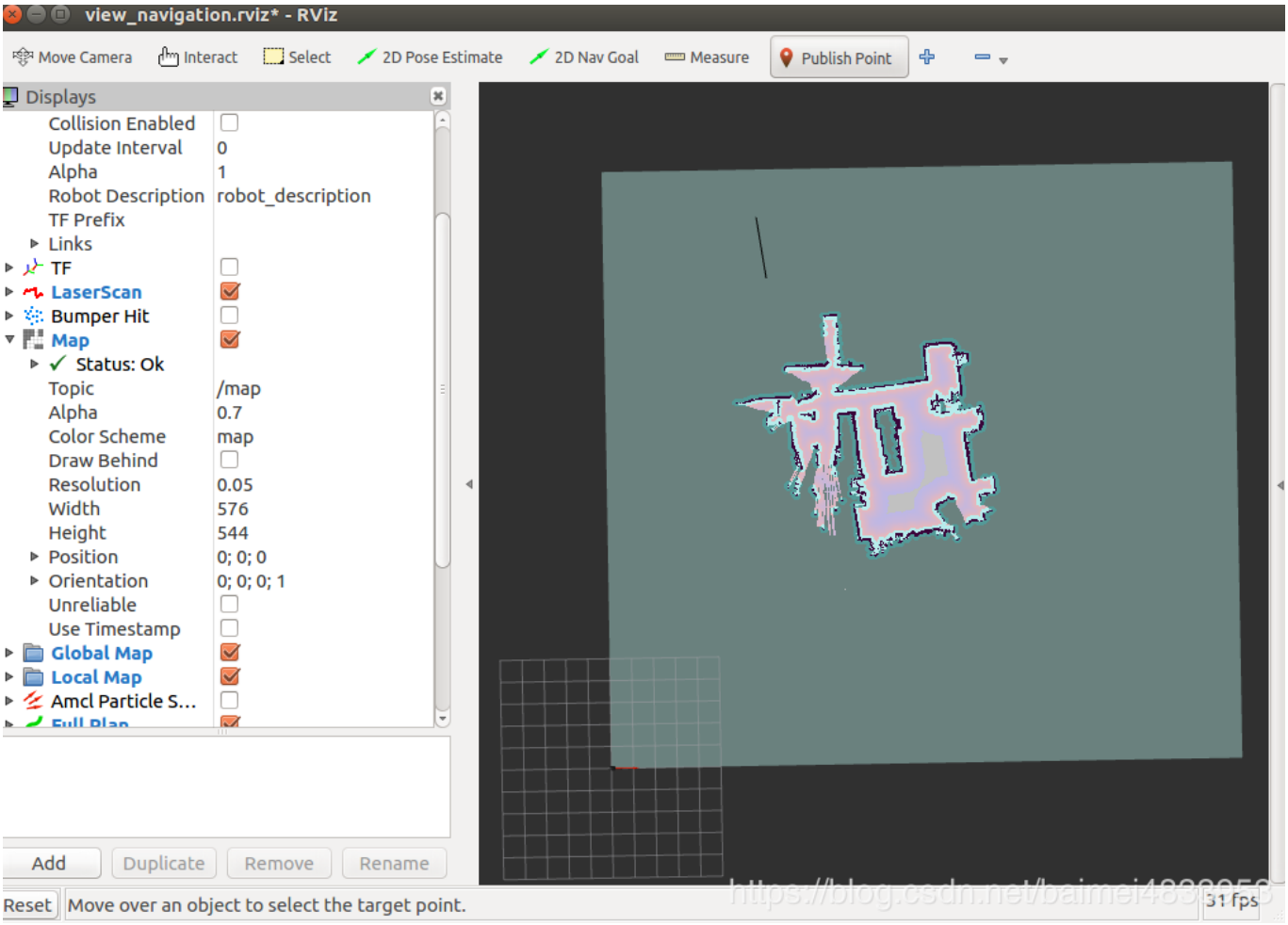
1.3坐标＆分辨率

修改test_map.yaml中的origin项 origin: [0.00000, 0.00000, 0.000000] 重新启动模拟器和Rviz



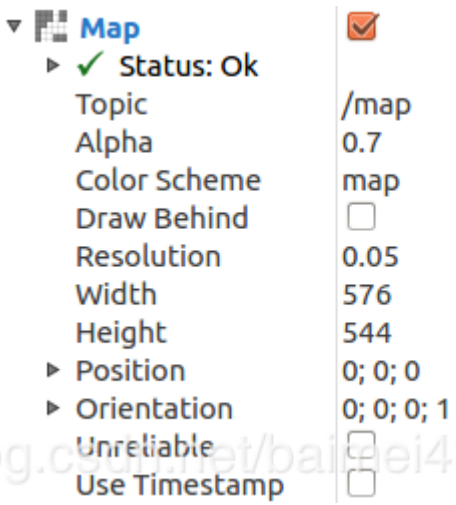
通过Rviz中地图与直接打开的地图对比可以看到机器人再地图的最左下角，也就是说最左下角是地图的原点(0,0)， 方向正左为 0 度

选中Rviz工具栏中的`Publish Point`



分别移动鼠标至地图的四个角，通过状态栏卡可看到实时的坐标显示分别为

- 左下角 (0 ,0,0)
- 右下角 (28.8 ,0,0)
- 右上角 (28.8 ,27,0)
- 左上角 (0 ,27,0)



可以看到上图中的display 中的Map下显示地图的信息

这里0.05就是地图分辨率了　　一个像素表示0.05m,实际该该地图对应大小即为(576 * 0.05, 544 * 0.05)即(28.8, 27.2),对应到上面四个角的坐标我们就可以知道x,y 坐标系如何了，实际跟我们高中几何坐标系一样，横向x轴向左增大，纵向y轴向上增大, 角度也同样为跟x的夹角

1.4初始位置

1.3开头我们设置`origin: [0.00000, 0.00000, 0.000000]`,现在我们修改回去`origin: [-13.800000, -12.200000, 0.000000]`再次重新启动模拟器，对比位置我们可以看到现在的坐标原点跑到了中间，但是坐标系仍是一样（横向x轴向左增大，纵向y轴向上增大, 角度也同样为跟x的夹角）

2.通过代码获取实时位置

通过base_link与map坐标的tf我们就可以得到机器人实时位置，之际贴代码

```
#!/usr/bin/env python
import rospy

from tf_conversions import transformations
from math import pi
import tf

class Robot:
    def __init__(self):
        self.tf_listener = tf.TransformListener()
        try:
            self.tf_listener.waitForTransform('/map', '/base_link', rospy.Time(),
rospy.Duration(1.0))
        except (tf.Exception, tf.ConnectivityException, tf.LookupException):
            return

    def get_pos(self):
        try:
            (trans, rot) = self.tf_listener.lookupTransform('/map', '/base_link',
rospy.Time(0))
        except (tf.LookupException, tf.ConnectivityException,
tf.ExtrapolationException):
            rospy.loginfo("tf Error")
            return None
        euler = transformations.euler_from_quaternion(rot)
        #print euler[2] / pi * 180

        x = trans[0]
        y = trans[1]
        th = euler[2] / pi * 180
        return (x, y, th)

if __name__ == "__main__":
    rospy.init_node('get_pos_demo',anonymous=True)
    robot = Robot()
    r = rospy.Rate(100)
```

```
    r.sleep()
    while not rospy.is_shutdown():
        print robot.get_pos()
        r.sleep()
```