1.base_link和laser_link

- 前文三轮全向机器人底盘(4)-3D仿真模型一文中我们使用URDF创建了一个底盘模型,我们添加显示TF属性后显示  TF 这里显示出URFD文件中定义好的2个Frame，base_link和laser_link; 显示的红绿蓝分别代表x，y，z轴。
- tf view 启动新窗口输入`rosrun tf tf_echo base_link laser_link`  tf view 可以看到打印出`laser_link`到`base_link`的tf， 这里`Translation`和`Rotation`即为位置与姿态的转换，其中`Rotation`以三种不同的形式展现出来；而这些关系定义实在URDF文件中可以找到,具体在哪发布出来的，输入以下命令

```
rosrun tf view_frames
evince frames.pdf
```

可以看到是`robot_state_publisher`发布`base_link`到`laser_link`的TF

2.odom

官方有个教程http://wiki.ros.org/navigation/Tutorials/RobotSetup/Odom, https://github.com/ros-planning/navigation_tutorials 这个包这里可以看到

```
cd ~/pibot/ros_ws/src
git clone https://github.com/ros-planning/navigation_tutorials.git
cd ..
catkin_make
rosrun odometry_publisher_tutorial odometry_publisher
```

可以看到多出一个`Frame`，并且在运动。切换`Fixed Frame`为`Odom`同时添加`Odom topic` 可以看到`base_link`做圆周运动,线速度和角速度可以在代码中看到

```
double vx = 0.1;
double vy = -0.1;
double vth = 0.1;
```

> 注：这里速度并非下发给机器人速度，而是机器人反馈的速度，对于**PIBOT**解算都是在下位机完成，直接串口读取出**x y th**即可

Odom可以理解全局一张超级大地图，机器人运动就发布base_link到Odom的tf,这样就好似机器人在这张大图上运动了。

3.map

前面说到Odom是一张大的地图， 那map就可以理解为这个大的地图中的一部分。导航启动机器人不是处于地图中的正确位置时，我们会通过 设置机器人的位置姿态，**其实是根据机器人当前位置，把map"贴到"odom这张大图上**,即修改odom与map的tf关系以达到设置机器人位置姿态 同样amcl也是如此（事实上2D Pose Estimate就是amcl包中订阅的topic的处理）