EKF-SLAM

Michael Gao

实验目标

- 利用ICP_Odom的信息作为prediction的输入 / 利用MarkerArray作为update的输入
- 实现EKF-SLAM

实验步骤

0. 状态空间定义

EKF-SLAM主要更新两个全局变量：`VectorXd status`， `MatrixXd covariance` 分别代表状态均值和状态方差：

$$\underbrace{\begin{pmatrix} \mathrm{X}_R \\ \mathrm{m}_1 \\ \vdots \\ \mathrm{m}_n \end{pmatrix}}_{\mu} \underbrace{\begin{pmatrix} \sum_{\mathrm{x}_R\mathrm{x}_R} & \sum_{\mathrm{x}_R\mathrm{m}_1} & \cdots & \sum_{\mathrm{x}_R\mathrm{m}_n} \\ \sum_{\mathrm{m}_1\mathrm{x}_R} & \sum_{\mathrm{m}_1\mathrm{m}_1} & \cdots & \sum_{\mathrm{m}_1\mathrm{m}_n} \\ \vdots & \vdots & \ddots & \vdots \\ \sum_{\mathrm{m}_n\mathrm{x}_R} & \sum_{\mathrm{m}_n\mathrm{m}_1} & \cdots & \sum_{\mathrm{m}_n\mathrm{m}_n} \end{pmatrix}}_{\Sigma}$$

`status`中储存机器人的x, y, θ和LanMark的全局坐标m_x, m_y.

由于地图中特征点数量不大，故此处固定状态空间大小为100个特征.

`status`初始化特征部分为全零列，`covariance`为值为inf的对角矩阵.

1. Prediction

由于`icp_odom`中直接提供了ICP里程计的全局坐标，故直接将`icp_odom`的增量作为prediction的控制输入：

$$\begin{bmatrix} \boldsymbol{x}_t \\ \boldsymbol{l}_t \end{bmatrix} = \begin{bmatrix} \boldsymbol{f}(\boldsymbol{x}_{t-1}, \boldsymbol{u}_t) \\ \boldsymbol{l}_{t-1} \end{bmatrix}$$

$$\begin{bmatrix} x' \\ y' \\ \theta' \end{bmatrix} = \begin{bmatrix} x + \delta x \\ y + \delta y \\ \theta + \delta\theta \end{bmatrix}$$

运动Jacobi：

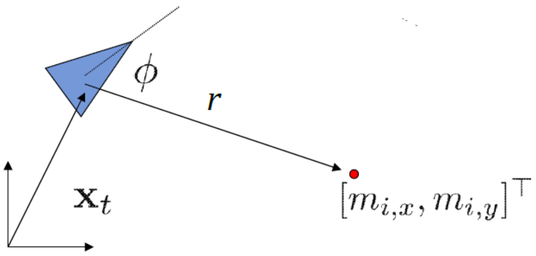
$$G_t = \begin{pmatrix} G_t^{\mathrm{x}} & 0 \\ 0 & I \end{pmatrix}$$

此处G_t^x取单位矩阵.

2. Update

2.0. 特征选择

选择$r\&\Phi$LandMark作为其特征：



$$\hat{\mathbf{z}}_t^j = \begin{pmatrix} \sqrt{(m_{jx}-x)^2+(m_{jy}-y)^2} \\ atan2\left(m_{jy}-y, m_{jx}-x\right)-\theta \end{pmatrix}$$

2.1. 与已知特征匹配

首先将接收到的LandMark转换到全局坐标系：

$$\begin{pmatrix} \bar{\mu}_{j,x} \\ \bar{\mu}_{j,y} \end{pmatrix} = \begin{pmatrix} \bar{\mu}_{t,x} \\ \bar{\mu}_{t,y} \end{pmatrix} + \begin{pmatrix} r_t^i \cos\left(\phi_t^i + \bar{\mu}_{t,\theta}\right) \\ r_t^i \sin\left(\phi_t^i + \bar{\mu}_{t,\theta}\right) \end{pmatrix}$$

```
lm_pos = tranToGlobal(lm_pos);
```

然后将接收特征与`status`中所有特征进行匹配：

```
int lm_id = findNearestMap(lm_pos);
```

如果未匹配到已知特征，则认为是新特征，此时将对状态空间进行更新：

```
// IF New feature
if(lm_id < 0){
lm_id = updateFeatureMap(lm_pos);
}
```

`status`直接stack入新特征的全局坐标，同时保证`covariance`有如下形式：

$$\begin{bmatrix} \Sigma_{t-1} & \mathbf{0} \\ \mathbf{0} & +\infty \end{bmatrix}$$

2.2 计算预估观测 Expected Observation

根据当前的预估状态计算预估观测：

$$\delta = \begin{pmatrix} \delta_x \\ \delta_y \end{pmatrix} = \begin{pmatrix} \bar{\mu}_{j,x} - \bar{\mu}_{t,x} \\ \bar{\mu}_{j,y} - \bar{\mu}_{t,y} \end{pmatrix}$$

$$q = \delta^\top \delta$$

$$
\begin{aligned}
\hat{\mathbf{z}}_t^i &= \begin{pmatrix} \sqrt{q} \\ \operatorname{atan} 2\left(\delta_y, \delta_x\right) - \bar{\mu}_{t,\theta} \end{pmatrix} \\
&= h\left(\overline{\boldsymbol{\mu}}_t\right)
\end{aligned}
$$

2.3. 计算观测 Jacobi

$$
\begin{aligned}
\operatorname{low} H_t^i &= \frac{\partial h\left(\vec{\boldsymbol{\mu}}_t\right)}{\partial \overline{\boldsymbol{\mu}}_t} \\
&= \begin{pmatrix} \frac{\partial \sqrt{q}}{\partial x} & \frac{\partial \sqrt{q}}{\partial y} & \ldots \\ \frac{\partial \operatorname{atan} 2(\ldots)}{\partial x} & \frac{\partial \operatorname{atan} 2(\ldots)}{\partial y} & \ldots \end{pmatrix}
\end{aligned}
$$

求导得Jacobi的具体表达式：

$$
\begin{aligned}
\operatorname{low} H_t^i &= \frac{\partial h\left(\overline{\boldsymbol{\mu}}_t\right)}{\partial \overline{\boldsymbol{\mu}}_t} \\
&= \frac{1}{q}\begin{pmatrix} -\sqrt{q}\delta_x & -\sqrt{q}\delta_y & 0 & \sqrt{q}\delta_x & \sqrt{q}\delta_y \\ \delta_y & -\delta_x & -q & -\delta_y & \delta_x \end{pmatrix}
\end{aligned}
$$

由于h对LandMark的求导是针对特定一个的，还需要将$_{low} H$ map 到高维：

$$
H_t^i = \Big[\underbrace{\frac{\partial \mathbf{h}}{\partial \mathbf{x}_t}}_{\substack{\text{w.r.t.} \\ \text{robot}}} \quad \underbrace{\ldots\ 0\ \ldots}_{\substack{\text{w.r.t.} \\ \text{other} \\ \text{landmarks}}} \quad \underbrace{\frac{\partial \mathbf{h}}{\partial \mathbf{m}_i}}_{\substack{\text{w.r.t.} \\ \boldsymbol{m}_i}} \quad \underbrace{\ldots\ 0\ \ldots}_{\substack{\text{w.r.t.} \\ \text{other} \\ \text{landmarks}}} \Big]
$$

$$
H_t^i = \operatorname{low} H_t^i F_{\mathrm{x,j}}
$$

映射矩阵：

$$
F_{\mathrm{x,j}} = \begin{pmatrix}
1 & 0 & 0 & 0 \cdots 0 & 0 & 0 & 0 \cdots 0 \\
0 & 1 & 0 & 0 \cdots 0 & 0 & 0 & 0 \cdots 0 \\
0 & 0 & 1 & 0 \cdots 0 & 0 & 0 & 0 \cdots 0 \\
0 & 0 & 0 & 0 \cdots 0 & 1 & 0 & 0 \cdots 0 \\
0 & 0 & 0 & \underbrace{0 \cdots 0}_{2j-2} & 0 & 1 & \underbrace{0 \cdots 0}_{2n-2j}
\end{pmatrix}
$$

2.4. EKF Update

根据如下公式完成更新：

$$
\begin{aligned}
K_t^i &= \bar{\Sigma}_t H_t^{i\top}\left(H_t^i \bar{\Sigma}_t H_t^{i\top} + Q_t\right)^{-1} \\
\overline{\boldsymbol{\mu}}_t &= \overline{\boldsymbol{\mu}}_t + K_t^i\left(\mathbf{z}_t^i - \hat{\mathbf{z}}_t^i\right) \\
\bar{\Sigma}_t &= \left(I - K_t^i H_t^i\right) \bar{\Sigma}_t
\end{aligned}
$$

此处要特别注意角度的归一化，尤其是在$\left(\mathbf{z}_t^i - \hat{\mathbf{z}}_t^i\right)$这一步：

```
Vector2d z_diff;
z_diff =  z_real - z_pre;
z_diff(1) = angleNorm(z_diff(1));
```

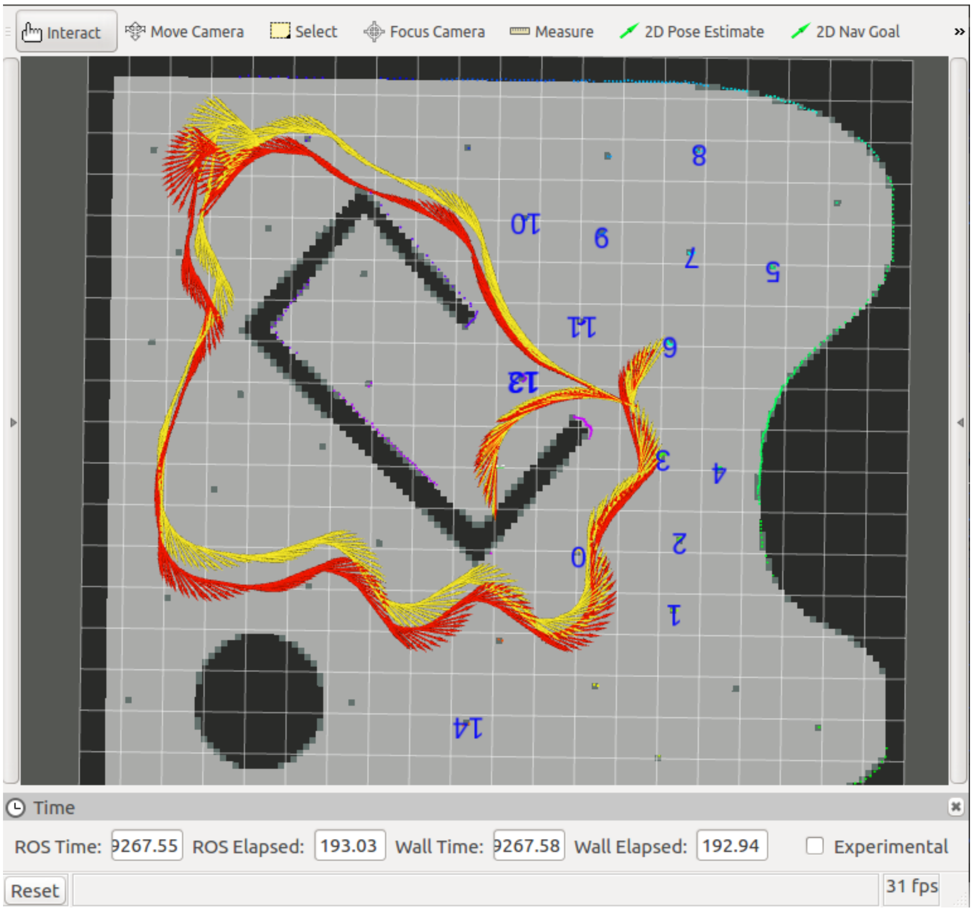
且在角度归一化时，要归一化到$(-\pi, \pi]$，而不是$(0, 2\pi]$.

实验结果

运行方法：

```
roslaunch course_agv_slam_task icp_all.launch
roslaunch course_agv_slam_task ekf.launch
```

完整结果展示见`ekfslam.mp4`



黄色的箭头是EFK坐标，红色的箭头是ICP导航坐标.

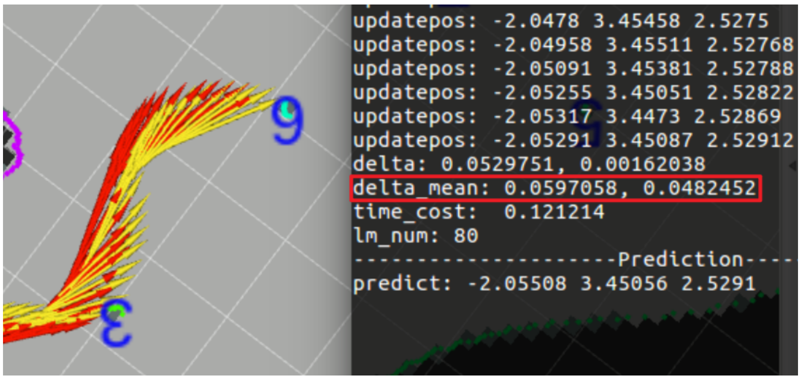
从实验结果可以看出，EKF-SLAM结果显著好于ICP导航，结果与真值十分接近，效果很好.

tf_listener 真值对比:

```
tf::StampedTransform transform;

try{
    listener.lookupTransform("world_base", "robot_base", ros::Time(0), transform);
}
catch (tf::TransformException &ex) {
    ROS_ERROR("%s",ex.what());
    cout<<"Don't Get"<<endl;
    ros::Duration(1.0).sleep();
}
```

```
double rel_x = transform.getOrigin().x();
double rel_y = transform.getOrigin().y();

double dx = fabs(rel_x - status(0));
double dy = fabs(rel_y - status(1));

dx_mean = (dx_mean * update_cnt + dx)/(update_cnt+1);
dy_mean = (dy_mean * update_cnt + dy)/(update_cnt+1);
```

实际运行：



EKF-SLAM与真值在全程平均误差：

$$|\bar{dx}| = 0.0597$$
$$|\bar{dy}| = 0.0482$$

实验误差很小，精度高.