oneway3124 / particle-filter-localization-simple-implementation

No description, website, or topics provided.

Edit

Manage topics

2 commits | 1 branch | 0 packages | 0 releases | 1 contributor | GPL-3.0

Branch: master ▾ | New pull request | Create new file | Upload files | Find file | Clone or download ▾

oneway3124 code from zju simple implementation — Latest commit 74390df 2 hours ago

LICENSE	code from zju simple implementation	2 hours ago
README.md	code from zju simple implementation	2 hours ago
particle_filter_localization.m	code from zju simple implementation	2 hours ago

README.md

粒子滤波算法

1 粒子滤波算法过程解析

粒子滤波定位算法的思想也是贝叶斯规则：

$$\frac{\text{Likelihood} * \text{Prior}}{\text{Marginal}}$$

以下的解析是结合Matlab代码所作的说明

1. 初始化
   i. 初始化一堆粒子，在MatLab中的代码显示NP=50，也就是总共有50个**采样粒子。**Estimated State [x y yaw]，可见每个状态有3个数据。px=repmat(xEst,1,NP);，
   ii. 计算初始权重pw=zeros(1,NP)+**1/NP**; 可见初始权重是均匀的
   iii. 初始化路标，landMarks=[10 0; 10 10; 0 15; -5 20]; 可见一共有**4个路标**
2. 预测：根据motion model与物体的控制信息u预测下个时刻粒子群中粒子的位置
   i. doControl()函数，输入参数time，得到**控制指令u**
   ii. doMotion()函数，输入初始状态x和控制指令u，得到**下一时刻**的x状态
   iii. doObservation()函数，输入参数**xGnd（没有噪声**的里程计位置估计），**xOdom（有噪声**的里程计位置估计），u（控制指令），landMarks, MAX_RANGE，输出参数是z,xGnd,xOdom,u
3. 更新：根据物体的**观测值**z与**地图值**zl计算出每个粒子的权重ww。更新粒子权重的依据是粒子的观测值与地图标志物**相似度的高低**，越高的话该粒子的权重越大
   i. 对每个粒子循环操作
   ii. doMotion()函数，对每个采样粒子输入初始状态x和控制指令u，得到**下一时刻**的x状态，并加入干扰
   iii. 计算权重，用各个路标距离的**高斯概率相乘**得到总概率
4. 重采样：根据粒子的权重w重新采样粒子

2 重要代码（Tasks)

2.1 观测模型

```
% do Observation model
function [z, xGnd, xOdom, u] = doObservation(xGnd, xOdom, u, landMarks, MAX_RANGE)
    global Qsigma;
    global Rsigma;
```

```
    % Gnd Truth and Odometry
    xGnd=doMotion(xGnd, u);% Ground Truth 理想状态
    u=u+sqrt(Qsigma)*randn(2,1); % add noise randomly
    xOdom=doMotion(xOdom, u); % odometry only

    %Simulate Observation
    z=[];
    for iz=1:length(landMarks(:,1))
        dx = xGnd(1)-landMarks(iz,1);
        dy = xGnd(2)-landMarks(iz,2);
        d=sqrt(dx^2+dy^2);
        if d<MAX_RANGE
            z=[z;[d+sqrt(Rsigma)*randn(1,1) landMarks(iz,:)]];   % add observation noise randomly
        end
    end
end
```

2.2 运动模型

```
% do Motion Model
function [ x ] = doMotion( x, u)
    global dt;
    Delta = [ [dt*cos(x(3)),0];
              [dt*sin(x(3)),0];
              [0,dt]];

    x = x+Delta*u;
end
```

2.3 高斯函数

```
% Gauss function
function g = Gaussian(x,u,sigma)
    g=exp(-((u-x)^2)/(sigma^2)/2.0)/sqrt(2.0*pi*(sigma^2));
end
```

2.4 粒子归一化

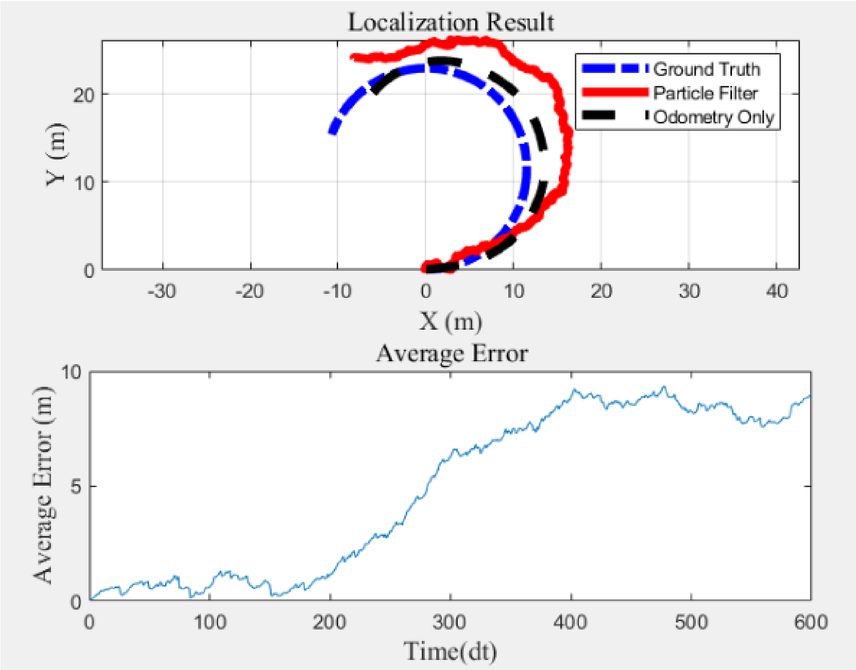
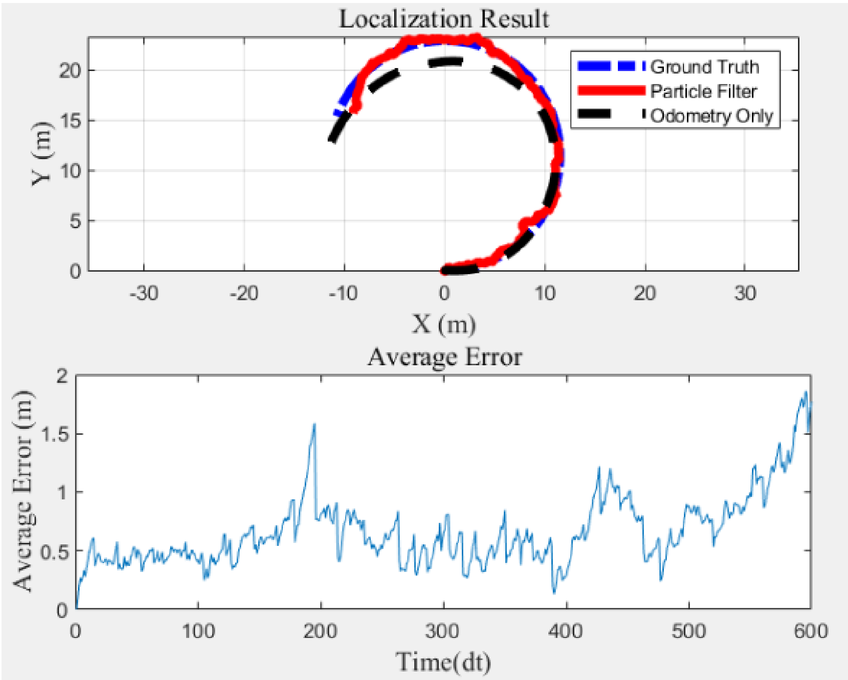
```
% Normalization
function pw=Normalization(pw,NP)
    pw=pw/sum(pw);

end
```

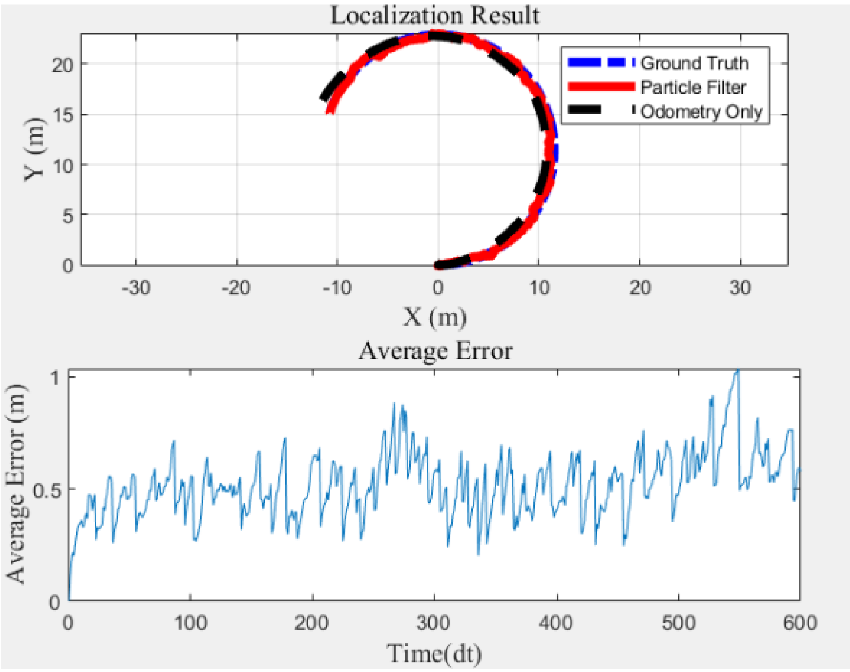
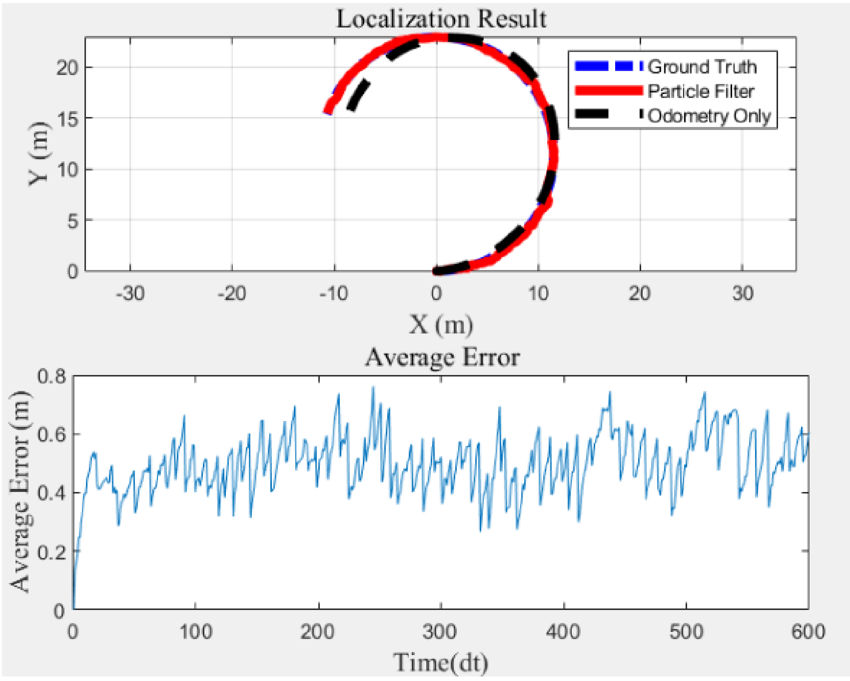
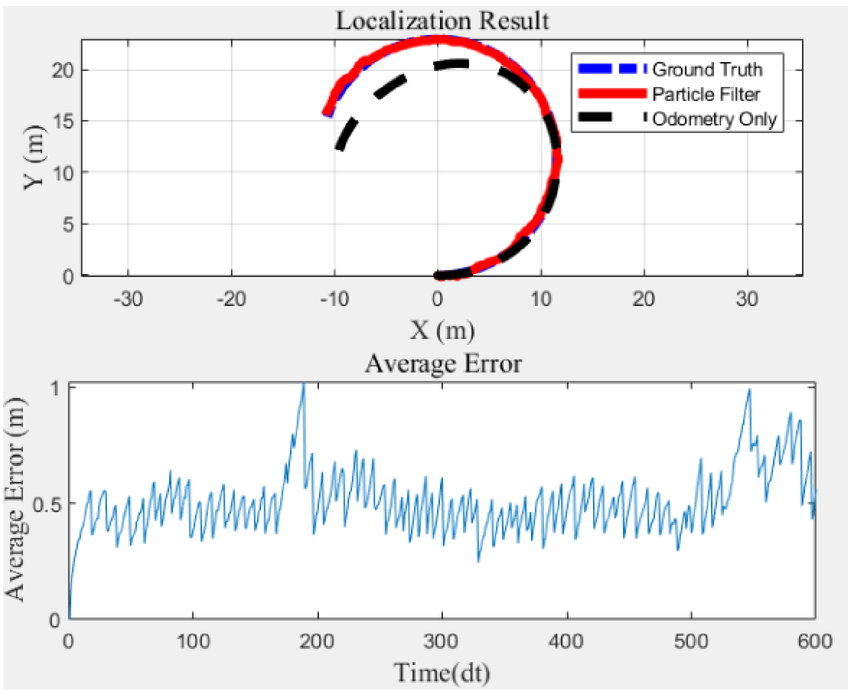
2.5 重采样

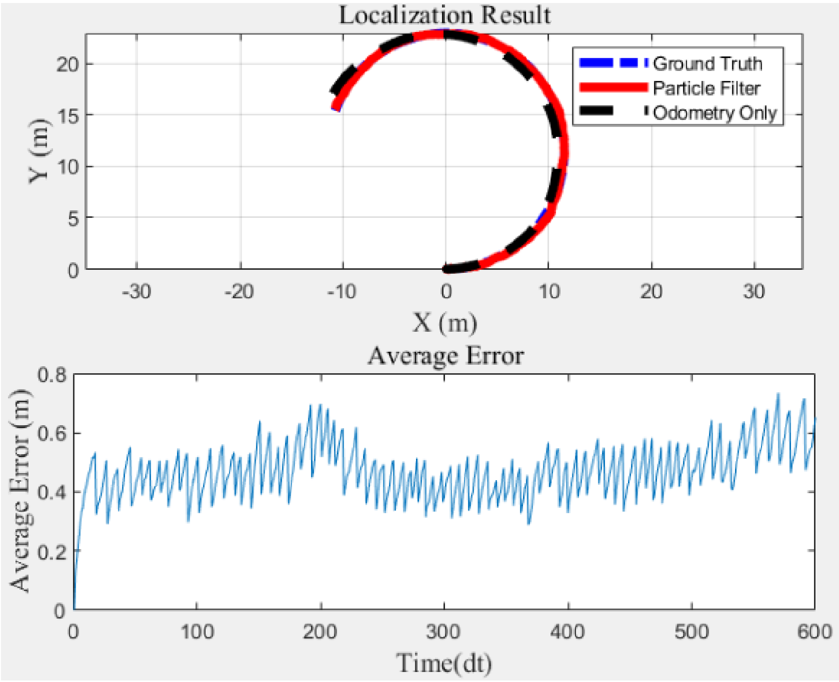
```
function [px,pw]=ResamplingStep(px,pw,NTh,NP)
    ww=pw(1);
    for iw=2:NP
        ww=[ww,ww(end)+pw(iw)];
    end
    pw1=[]
    pp=[];
    for i=1:NP
        r=rand();
        for j=1:NP
            if ww(j)>r
                pp=[pp,px(:,j)];
                pw1=[pw1,pw(:,j)]
                break
            end
        end
    end
    px=pp;
    pw=pw1;
end
```

3 参数对比实验

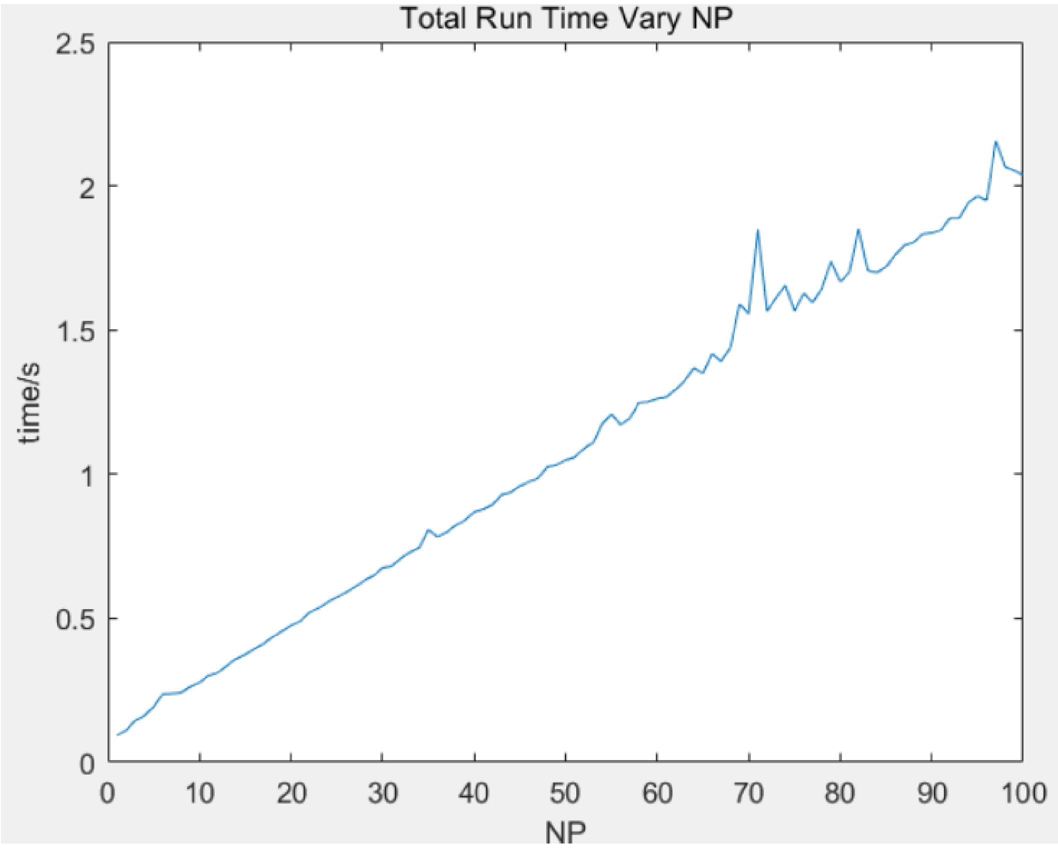
3.1 NP数效果实验

NP	效果	运行时间
3		0.220
5	 !	0.286

NP	效果	运行时间
10		0.387
25		0.640
50		1.131

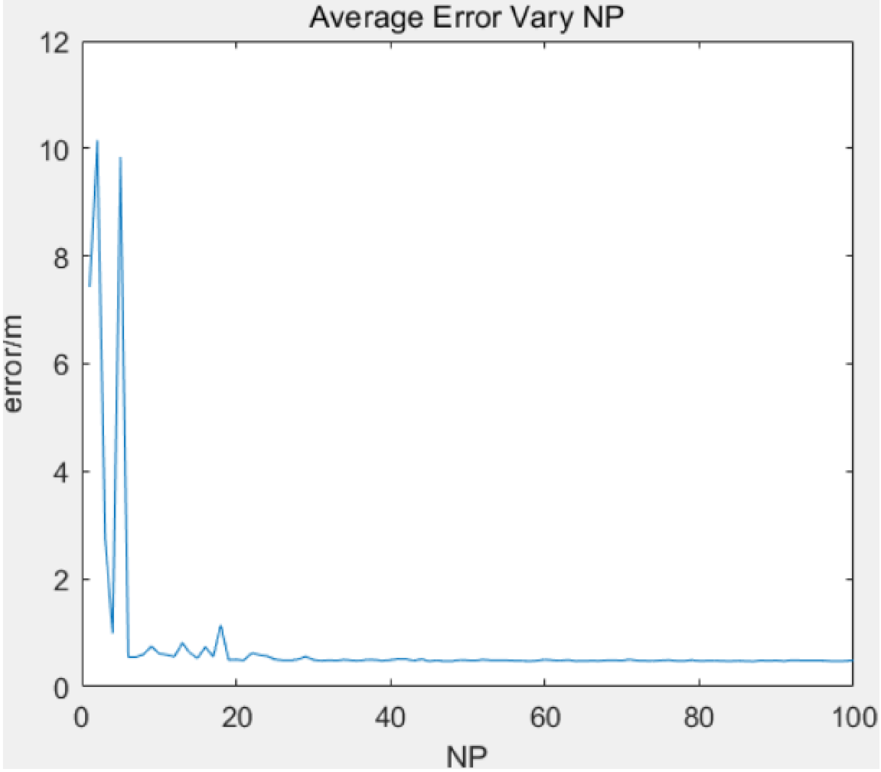
NP	效果	运行时间
100		2.159

3.2 NP数对时间的影响



可见运行时间与NP数成正比关系

3.3 NP数对误差的影响



可见误差趋向于0.5，这主要时由高斯噪声造成的

4 结论与展望

粒子滤波算法是基于概率的定位算法，主要有以下优点：

- 理解简单，一句话就是越相似，存活概率越大
- 计算量不大（计算量与粒子数线性相关）

但也存在以下问题：

- 严重依赖于对初始状态的估计，选择不当可能发散
- 需要有固定的路标

参考文献

> https://en.wikipedia.org/wiki/Particle_filter
>
> ["Probabilistic Robotics"][Sebastian Thrun]