Acknowledgements

This paper was supported by plan projects National Administration of Surveying, Mapping and Geoinformation of China (Grant NO.2013CH-15, special public welfare industry research 201512009) and by the National Natural Science Foundation of China (Grant No.41301429). Project also supported by the Special Scientific Research Fund of Surveying Public Welfare Profession of China (Grant No. 201512009).

References

1. Alboul, L., Chliveros, G., A system for reconstruction from point clouds in 3D: Simplification and mesh representation. Control Automation Robotics & Vision (ICARCV), 2010, 2010 11th International Conference, pp. 2301-2306.
2. Alexa, Behr J, Cohen-Or D, Point Set Surfaces. In Proc. IEEE Visualization, 2001, pp. 21-28.
3. Chen Y H, Ng CT, Wang Y Z, Date reduction in integrated reverse engineering and rapid prototyping. Computer Integrated Manufacturing, 1999, 12(2), pp. 97-103.
4. Chen Zhangwen, Da Feipeng, 3D point cloud simplification algorithm based on fuzzy entropy iteration. Acta Optica Sinica, 2013, 33(8), pp. 0815001.
5. Filip D, Magedson R, Markot T, Surface algorithms using bounds on derivatives. Computer Aided Geometric Design, 1986, 3(2),pp: 295-311.
6. Li Fengxia, Rao Yonghui, Liu Chen, Jie Fei, Point cloud simplification based on angle between normal. Journal of System Simulation, 2012, 24(9), pp. 1980-1983, 1987.
7. Ni Xiaojun, Research and implement on the algorithm of point cloud simplification and fast reconstruction. Suzhou, Soochow University, 2010, pp. 9-28.
8. Shi Baoquan, Liang Jin, Zhang Xiaoqiang, Shu Wan, Research on point cloud simplification with preserved features. Journal of XiAn Jiaotong University, 2010, 44(11), pp. 38-40.
9. Wang Renfang, Zhang Sanyuan, Ye Xiuz, Simplification of point-sampled model based on geometry images. Journal of Computer Aided Design & Computer Graphics, 2007, 19(8), pp. 1022-1027.
10. Xiao ZhaoXia, Huang Wenming, Kd-tree based nonuniform simplification of 3D point cloud. In Proc. WGEC '09, 2009, pp.339-342.
11. Zhang Liyan, Zhou Rurong, Cai Weibin, Zhou Laishui, Research on cloud data simplification. Journal of Computer Aided Design & Computer Graphics, 2001, 13(11), pp.1019-1023.
12. Zhou Bo, Chen Yinggang, Gu Zeyuan, Data point reduction on octree cube algorithm. Modern Manufacturing Engineering, 2008(3), pp. 64-66, 67.

A New Particle Filtering Method for Robot Sensors Autonomous Positioning

Chang Lin *, Piao Songhao, Leng Xiaokun, Li Guo, Wang Di

Harbin Institute of Technology, 92West Dazhi St.Harbin, 150001, China

Abstract

The positioning is to estimate the position and posture of the robot and accurate position estimation is necessary to achieve autonomous navigation. So, the study of the positioning method of the mobile robot has a very important significance. Now there are three mainly approximate techniques types: the first is the probability density function

on the approximate state space. Extended Kalman filter (EKF) as well as the EKF variants, approximate Bayesian formula with the increase of the traveling distance, the accumulated error will grow indefinitely. Therefore, dead reckoning method is only suitable for short-term and short-distance position estimation and with low precision. But all the particle filter algorithms is deficient in that weight variance with the random time increasing, sample weight gathers together into a small number of samples. To solve this problem, this paper presents the algorithms based on neural networks, and the importance of sample to adjust the particle filter (NNISA-PF). The technical data derives from laboratory and in-situ experimentation. The algorithm contains the establishment of robot motion system, the sensor observation model and the particle filter based on GRNN sample adjustment. Also the algorithms takes advantage of the simulation plat form provided by Alberto Vale to make a further analysis of the results of robot localization, and the conclusion suggests that through combining neural network theory with particle filter, so that the samples effectiveness and diversity to be kept and reduce sample dilution in the re-sampling stage. The experimental results show that the improvement measures can effectively improve the performance of the algorithm, so that it enables them to maintain a reliable positioning.
Keywords: AUTONOMOUS POSITIONING, PARTICLE FILTERING METHOD, ROBOT SENSORS

1. Introduction

The positioning is to determine the location of the robot in its working environment. For mobile robot, accurate position estimation is the necessary content to achieve autonomous navigation. The location information is the basis of navigation and control decisions. Therefore, the study of the positioning method of the mobile robot has a very important significance.

There are many positioning methods of the mobile robot. Some depend on the external environmental information perception. And some positioning methods rely entirely on the robot itself motion information. The most primitive method is to use submerged navigation belt for navigation. Later, it is appeared environment map-based positioning which based on the signal lights and the signal of natural landmark.

The most basic positioning method is dead reckoning, where the robot merely measures a signal originating within itself. The structure of dead reckoning method is simple, inexpensive, but accumulated error is large. With the increase of the traveling distance, the accumulated error will grow indefinitely. Therefore, dead reckoning method is only suitable for short-term and short-distance position estimation. For long-term positioning, we can use the sensor to observe the environment based on dead reckoning, in order to achieve precise robot positioning.

Meanwhile, the robot pose can be seen as the state of the system, and use the Bayesian filtering to estimate the position and orientation of the robot, where the most commonly method is the Kalman filter algorithm. Kalman filter algorithm is a simple and effective linear optimal recursive estimation algorithm. It has robot pose express as a Gaussian probability distribution function, using the mean and variance of the robot to deliver its pose and uncertainty. However, Kalman filter must assume that the system noise and the observation noise must be white noise of the Gaussian distribution.Whereas, mobile robot often has nonholonomic constraints. And the robot motion equation and observation equation is a highly nonlinear system, which greatly limits the application of the Kalman filter algorithm.

To solve this problem, Use all kinds of different approximate technology to estimate suboptimal pose has become a important positioning method. D Fox using the Markov localization to approximate system posterior probability density with the state space of the sample set and through robots' position probability of the piecewise linear function, global space can be searched and agile for different sports and perception model (7, 8). Considering the number of dimensions, resolution and size of the grid, Markov needs optimization. After reducing the consumption of the computational resources, it can be used for real-time system, which greatly increases the complexity of the calculation.

At the same time, through the Extended Kalman filtering (EKF) and the different varieties of EKF, Julier and Roumeliotis and others propose methods such as UKF (Unscented Kalman Filter), IEKF (Iterated Extended Kalman Filter) and PKF (Perturbation Kalman Filter), etc. to approximate nonlinear motion equation and nonlinear observation equation through the linear system function and use the most optimal linear Kalman filtering framework to deal with nonlinear state estimation problem in recurrence [9, 10]. Although EKF, UKF and some methods are the nonlinear gaussian filtering method based on analytical approximation method, using kalman recursive way and with small amount of calculation, they are all based on local linearization Gaussian model, the posterior probability distribution is also similar to the Gaussian distribution, for strong nonlinear, non-Gaussian property system, the estimation precision is not ideal.

For Markov localization algorithm, the Monte Carlo Localization is its effective improvement, belonging to a particle filter technology. This technology is not only the storage space of the algorithm is reduced, and the positioning result is more accurate than Markov. The particle filter algorithm has an obvious flaw: Weight variance with the random time increasing, sample weight gathers together into a small number of samples, as the results of making sample set unable to express the desired posterior probability density, which cause a serious degradation phenomenon of the sample. To solve this problem, this paper presents the algorithms based on neural networks, the importance of sample to adjust the particle filter (NNISA-PF).

2. Model

Before the model determined, the first step is to make sure coordinate system. In the mobile robot localization study, the main use of the three types of coordinate systems: The first category is the Cartesian coordinate system; second type is the polar coordinate system; the third class is the DIN70000 coordinate system. This paper selects the Cartesian coordinate system as the representation of rectangular coordinate system. Mobile robot autonomous positioning research mainly use in the three coordinate systems: global coordinate system $X_W O_W Y_W$, robot coordinate system $X_R O_R Y_R$ and the sensor coordinate system $X_S O_S Y_S$. Their mutual relations are shown below **Figure 1**.

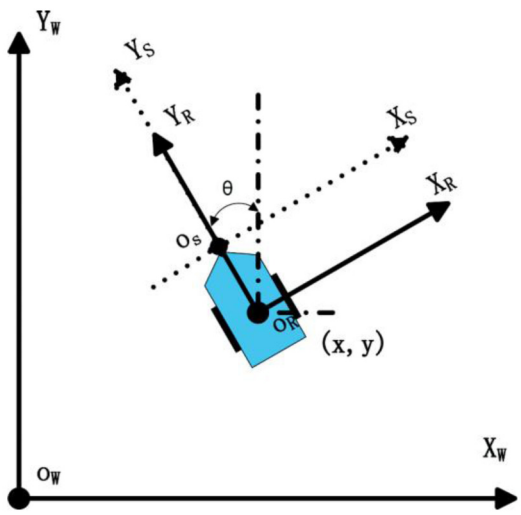

Figure 2.1. The coordinate system of the mobile robot

2.1. The Establishment of Robot Motion System

The positioning is to estimate the position and posture of the robot, which is the process of determining the position of the robot in the global coordinate system and the vehicle body direction. The position of the robot can be expressed by a point (x, y) in the global coordinate system. And the vehicle body direction expresses as angle θ with the robot body deviating from the axis of the global coordinates Y_W. θ direction is defined as follow: Y_W axis is 0 degrees, and the clockwise direction is negative, the counterclockwise direction as positive, as **Figure 2** shown below. Thus, in a movement of the two-dimensional plane environment, the position and orientation of the robot can be expressed as a three-dimensional state vector $\mathbf{X} = [x\ y\ \theta]$, where the ($x$, y) represents the position of the robot, θ represents the robot body direction.

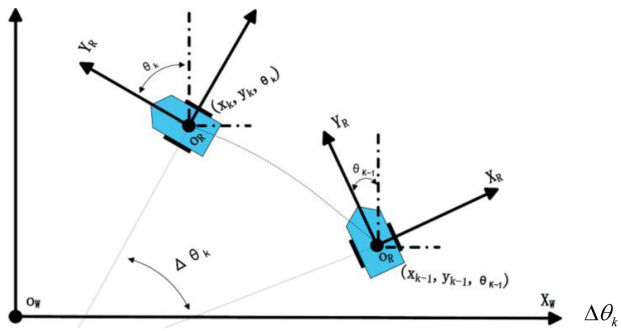

Figure 2.2. Mobile robot motion model

This paper adopts the odometer to reckon robot relative position and orientation change. It is based on the photoelectric encoder installed in two driving wheel motor to detect wheel's adduction of radian in a certain period of time. Set the wheel radius is r, the optical encoder is p line/turn and the optical encoder's output pulse number in ΔT time is N, then the wheel movement distance is $\Delta d = 2 \times \frac{N}{p} \times \pi r$. Hypothetically, Assuming that the moving distance of the left and right wheels detected by the optical encoder respectively is Δd_L and Δd_R, and the distance of two wheel is a. When the robot move to position of $\mathbf{X}(k) = (x_k, y_k, \theta_k)$ from $\mathbf{X}(k-1) = (x_{k-1}, y_{k-1}, \theta_{k-1})$, the moving distance of the robot is $\Delta D = (\Delta d_L + \Delta d_R)/2$, and turned angle of the robot is $\Delta\theta = (\Delta d_L - \Delta d_R)/a$. As **Figure 2** is shown in the graph, $\mathbf{u}(k) = (\Delta D_k, \Delta\theta_k)$ is the input of the model, so the robot motion model can be expressed as follows:

$$\mathbf{X}(k) = f(\mathbf{X}(k-1), \mathbf{u}(k)) + \omega(k) \tag{2.1}$$

The $\omega(k)$ is zero mean white Gauss noise.

In order to better approximate the actual trajectory, it is assumed that the robot is moving along an arc movement. The arc model considers not only the

displacement change of robot motion, but also the change of direction angle in movement. Therefore the actual effect of robot motion trajectory with arc approximation is better. As **Figure 2** inferred the arc model equation is:

$$\mathbf{X}(k)=f(\mathbf{X}(k-1),\mathbf{u}(k))=\begin{bmatrix} x_{k-1}+\dfrac{\Delta D_k}{\Delta\theta_k}(\cos(\theta_{k-1}+\Delta\theta_k)-\cos(\theta_{k-1})) \\ y_{k-1}+\dfrac{\Delta D_k}{\Delta\theta_k}(\sin(\theta_{k-1}+\Delta\theta_k)-\sin(\theta_{k-1})) \\ \theta_k=\theta_{k-1}+\Delta\theta_k \end{bmatrix} \quad (2.2)$$

Among them,

ΔD_k is the length traveled by the robot in time ΔT;

$\Delta\theta_k$ is the deflection angle of bodywork direction, and $\Delta\theta_k$ signs in a counterclockwise direction as the reference.

If the linear and angular velocities of the robot movement is (v,γ) at the moment before, and the time interval is ΔT. Therefore, the distance traveled by the robot is $v\times\Delta T$ and the direction of deflection is $\gamma\times\Delta T$. So that you can get robot traveled distance and the direction of deflection $(\Delta D_k,\Delta\theta_k)$ in time ΔT. When a controlled quantity containing forward speed and angular velocity puts on the robot, the motion model can be expressed as follow:

$$\mathbf{X}=\begin{bmatrix} x(k) \\ y(k) \\ \phi(k) \end{bmatrix}=\begin{bmatrix} x(k-1)+v\Delta T\cos(\phi(k-1)+\gamma(k)\Delta T) \\ y(k-1)+v\Delta T\sin(\phi(k-1)+\gamma(k)\Delta T) \\ \phi(k-1)+\gamma(k)\Delta T \end{bmatrix}+\begin{bmatrix} \omega_x \\ \omega_y \\ \omega_\phi \end{bmatrix} \quad (2.3)$$

2.2. Sensor Observation Model

The robot applies with the information of sensor observation to infer its own position. The ranging sensor observed value $\mathbf{Z}$ is an environment characterized by relative distance and the direction of sensor, which is represented in polar coordinates as follows:

$$\mathbf{Z}=[\rho\ \varphi]^T \quad (2.4)$$

Shown in **Figure 3**, the observation model describes the relationship between the sensor observables with the mobile robot position; and the observation equation is as follows:

$$\mathbf{Z}(k)=h(x(k))+v(k) \quad (2.5)$$

Wherein,

$\mathbf{Z}(k)$ is the observed quantity in the k time;

$h(\cdot)$ is a measurement function;

$v(k)$ is the observation noise, used to describe the measurement of the noise and the model error itself.

Assuming the current robot position is $\mathbf{X}(k)$, the position of an environmental target feature is $\mathbf{X}_l=(x_l,y_l)$, thus observation model of the system is as follows:

$$\mathbf{Z}(k)=\begin{bmatrix} \rho(k) \\ \varphi(k) \end{bmatrix}=\begin{bmatrix} \sqrt{(x_l-x(k))^2+(y_l-y(k))^2} \\ \tan^{-1}\left(\dfrac{y_l-y(k)}{x_l-x(k)}\right)-\theta(k) \end{bmatrix}+v(k) \quad (2.6)$$

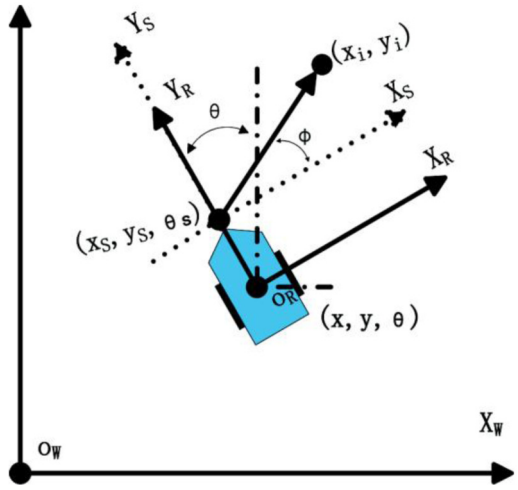

Figure 2.3. Observation model

3. Method

The particle filter is a statistical filtering method based on the Monte Carlo method and the recursive Bayesian estimation, which uses the Monte Carlo method to solve Bayesian estimation calculus in accordance with the law of large numbers. The basic idea is: Firstly according to the system state vector's experience conditional distribution in the state space to produce a set of random sample set, calls these samples for the particles. Then constantly adjusted the weight and position of the particle according to the measured, the initial experience of condition distributions are corrected through the particle information adjusted. Its essence is to use by particle and its weight of discrete random measure approximate related probability distribution and to update discrete

measure by the algorithm recursive. When the sample size is large, this Monte Carlo description approximates the true state variables posteriori probability density function.

3.1. The Particle Filter Algorithm

Priori known dynamic system state conditional probability $p(x_0)$, and using of $\left\{x_{0:k}^i, \omega_k^i\right\}_{i=1}^{N_s}$ to describe the target state x_k 's posterior probability distribution $p(x_{0:k} \mid Z_{1:k})$ in the k time, $\left\{x_{0:k}^i, i=0,\ldots,N_s\right\}$ is the corresponding weight $\left\{\omega_k^i, i=0,\ldots,N_s\right\}$'s set of particles. Right value is normalized to $\sum_i \omega_k^i$, then the target state posterior in the k time probability distribution can be discretely weighted

$$p(x_{0:k} \mid Z_{1:k}) \approx \sum_{i=1}^{N_s} \omega_k^i \delta(x_{0:k} - x_{0:k}^i) \tag{3.1}$$

Weights are selected by importance sampling method. If the particle set $\left\{x_{0:k}^i\right\}_{i=1}^{N_s}$ can be gotten by important density function $q(x_{0:k} \mid z_{1:k})$, then the right value is:

$$\omega_k^i = \propto \frac{p\left(x_{0:k}^i \mid z_{1:k}\right)}{q\left(x_{0:k}^i \mid z_{1:k}\right)} \tag{3.2}$$

If the importance density can be decomposed into:

$$q\left(x_{0:k} \mid z_{1:k}\right) = q\left(x_k \mid x_{0:k-1}, z_{1:k}\right) q\left(x_{0:k-1}, z_{1:k-1}\right) \tag{3.3}$$

By $q\left(x_k \mid x_{0:k-1}, z_{1:k}\right)$ to obtain particle $\left\{x_k^i\right\}_{i=1}^{N_s}$ and by $q\left(x_{0:k-1}, z_{1:k-1}\right)$ to get particle sets $\left\{x_{0:k-1}^i\right\}_{i=1}^{N_s}$ you can get a new set of particles $\left\{x_{0:k}^i\right\}_{i=1}^{N_s}$.

Since the posterior probability density function can be expressed as:

$$p\left(x_{0:k} \mid z_{1:k}\right) = \frac{p\left(z_k \mid x_{0:k}, z_{1:k-1}\right) p\left(x_{0:k} \mid z_{1:k-1}\right)}{p\left(z_k \mid z_{1:k-1}\right)} \tag{3.4}$$

And

$$p\left(x_{0:k} \mid z_{1:k-1}\right) = p\left(x_k \mid x_{0:k-1}, z_{1:k-1}\right) p\left(x_{0:k-1} \mid z_{1:k-1}\right).$$

Therefore, the posterior probability density can be obtained:

$$p(x_{0:k} \mid z_{1:k}) = \frac{p(z_k \mid x_k) p(x_k \mid x_{k-1})}{p(z_k \mid z_{1:k-1})} p(x_{0:k-1} \mid z_{1:k-1}) \tag{3.5}$$

Obviously

$$p(x_{0:k} \mid z_{1:k}) \propto p(z_k \mid x_k) p(x_k \mid x_{k-1}) p(x_{0:k-1} \mid z_{1:k-1}) \tag{3.6}$$

Through Substituting, you can get the importance weights update formula

$$\begin{aligned} \omega_k^i &\propto \frac{p(z_k \mid x_k^i) p(x_k^i \mid x_{k-1}^i) p(x_{0:k-1}^i \mid z_{1:k-1})}{q(x_k^i \mid x_{0:k-1}^i, z_{1:k}) q(x_{0:k-1}^i \mid z_{1:k-1})} \\ &= \omega_{k-1}^i \frac{p(z_k \mid x_k^i) p(x_k^i \mid x_{k-1}^i)}{q(x_k^i \mid x_{0:k-1}^i, z_{1:k})} \end{aligned} \tag{3.7}$$

If $q(x_k \mid x_{0:k-1}, z_{1:k}) = q(x_k \mid x_{k-1}, z_k)$, importance density function depends only on x_{k-1} and z_k. In the calculation, it is only need to store particles $\left\{x_k^i\right\}_{i=1}^{N_s}$, without concern for the particle set $\left\{x_{0:k-1}^i\right\}_{i=1}^{N_s}$ and past measurement values $z_{1:k-1}$. Weight of amended is:

$$\omega_k^i \propto \omega_{k-1}^i \frac{p(z_k \mid x_k^i) p(x_k^i \mid x_{k-1}^i)}{q(x_k^i \mid x_{k-1}^i, z_k)} \tag{3.8}$$

Standard particle filter algorithm selects prior probability density that is the most easy to implement as important density function. i.e.

$$q(x_k^i \mid x_{k-1}^i, z_k) = p(x_k^i \mid x_{k-1}^i) \tag{3.9}$$

Through Substituting, importance weights can be simplified as

$$\omega_k^i \propto \omega_{k-1}^i p(z_k \mid x_k^i) \tag{3.10}$$

The weights ω_k^i normalization. i.e.

$$\omega_k^i = \omega_k^i / \sum_{i=1}^{N_s} \omega_k^i \tag{3.11}$$

And then the posterior probability density $p(x_k \mid z_{1:k})$ can be expressed as

$$p(x_k \mid z_{1:k}) \approx \sum_{i=1}^{N_s} \omega_k^i \delta(x_k - x_k^i) \tag{3.12}$$

Wherein the weight value as the formula shown. Definitely, when $N_s \to \infty$, by the law of large numbers it can ensure that the above equation can be close to the real posterior probability $p(x_k \mid z_{1:k})$.

However, since the standard particle filtering algorithm selects a priori probability density as an important density function, this selection method is able to obtain better results in low measurement accuracy requirements occasions. Yet because there is no consideration of the current measured value, the samples obtained by sampling from the importance of the probability density and the real posterior probability density have large deviation. Therefore, the importance weights of the variance over time will random increase, so that the weight is concentrated to a small number of particles of the particles. Even

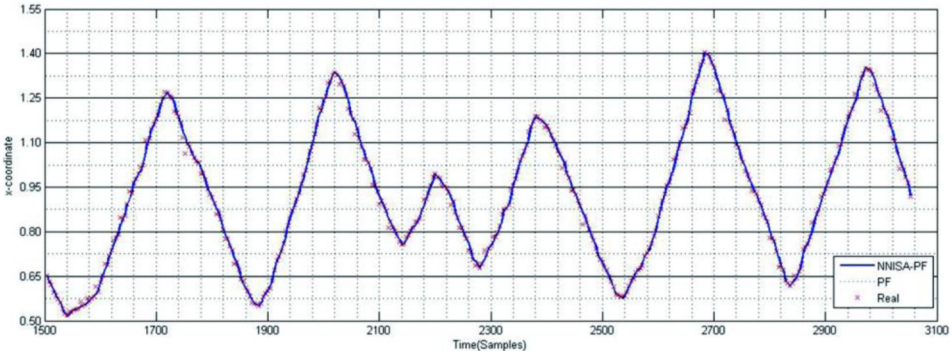

Figure 3.1. Estimation of x-direction position

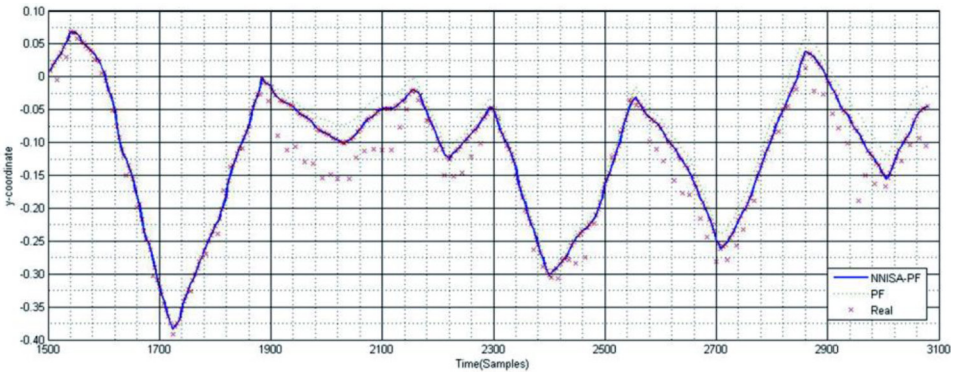

Figure 3.2: Estimation of y-direction position

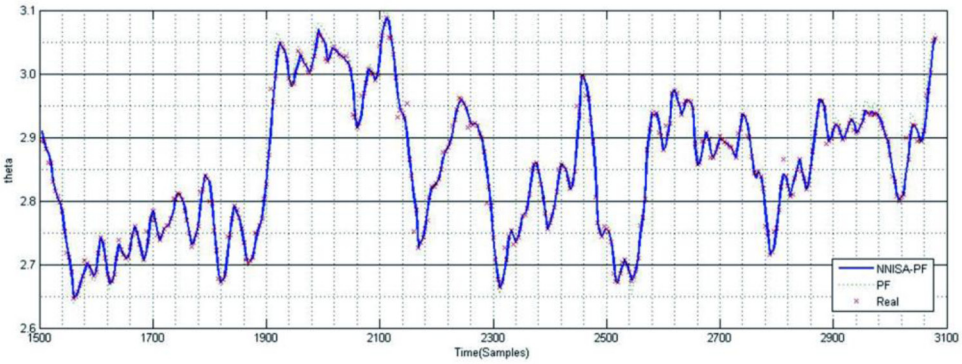

Figure 3.3. Estimation of robot azimuth

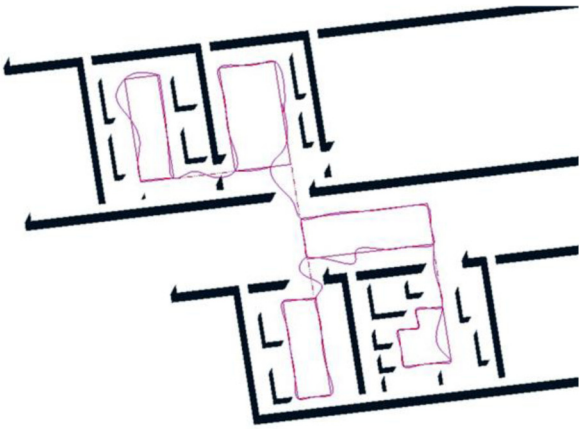

Figure 3.4. The positioning results of ordinary PF algorithm with 300 samples

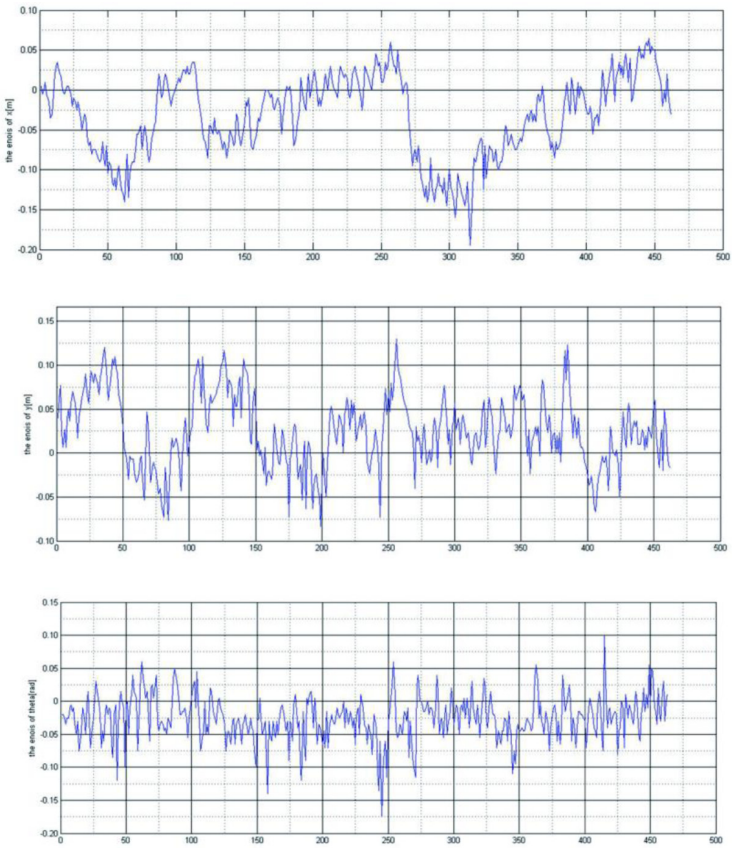

Figure 3.5. The positioning deviation of ordinary PF algorithm with 300 samples

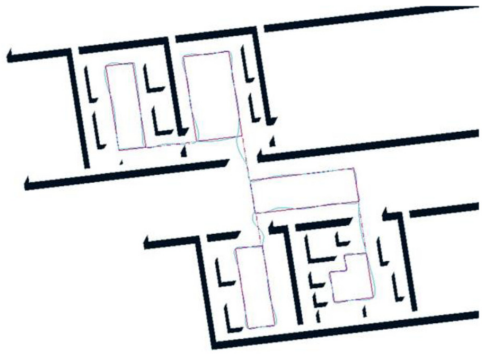

Figure 3.6: The positioning results of NNISA-PF algorithm with 100 samples

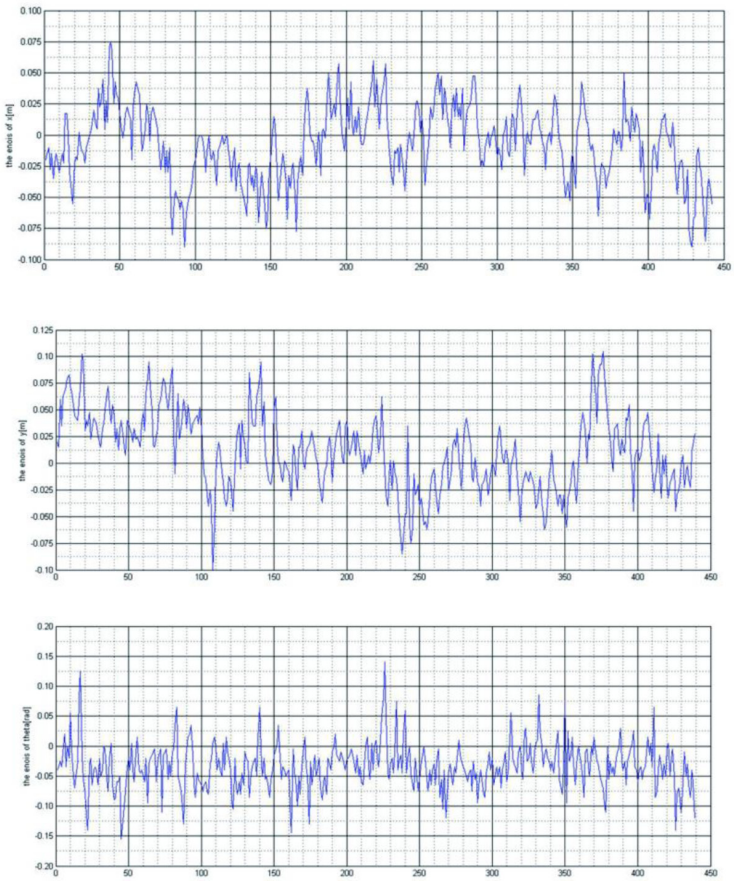

Figure 3.7. The positioning deviation of NNISA-PF algorithm with 300samples

after a few steps of the recursive, it may be in a particle with nonzero values, and values of the weights of the other particles are small, which can be negligible. So, a large number of calculations are wasted in those who are used to update the $p(x_k \mid z_{1:k})$ estimated that almost does not work on the particle. The results set of particles cannot express the actual posterior probability distribution, causing the particle degradation. To avoid degradation occurs introducing re-sampling technique, but the negative effect of the re-sampling technique is sample depletion, i.e. particles larger weights are selected more than once and many complex points included in the same sampling results. Thus that will lose the sample particle diversity.

3.2. GRNN

GRNN is a new type of neural network, a branch of the Radial Basis Function (RBF) neural network, and it's a kind of feed-forward neural network which is based on nonlinear regression. This network has strong advantages in the approximation ability, learning speed as well as forecast effect. Using GRNN, there is no need to determine the form of equation, and it uses probability density function instead of inherent equation form. Through the PARZEN nonparametric estimation, we obtain the joint probability density function between dependent and independent variables from measuring samples, then we can calculate the regression value of the dependent variable to the independent variable value.

Assume the joint probability density function $f(x,y)$ of random vector x and y, x value for x_0, the regression value of y is,

$$E(y \mid x_0) = \tilde{y}(x_0) = \frac{\int_{-\infty}^{\infty} y f(x_0, y) dy}{\int_{-\infty}^{\infty} f(x_0, y) dy} \tag{3.13}$$

Apply PARZEN nonparametric estimation; the density function can derive from the sample data $\{x_i, y_i\}_{i=1}^{m}$, which can be listed as follows:

$$f(x_0, y) = \frac{1}{n(2\pi)^{\frac{p+1}{2}} \sigma^{p+1}} \sum_{i=1}^{n} e^{-d(x_0, x_i)} e^{-d(y, y_i)} \tag{3.14}$$

$$d(x_0, x_i) = \sum_{j=1}^{p} [(x_{0j} - x_{ij}) / \sigma]^2 \tag{3.15}$$

$$d(y, y_i) = (y - y_i)^2 \tag{3.16}$$

Here,

n is for sample capacity;

p is the dimension of x;

σ is the width coefficient of Gaussian function, which is used as smooth factor here.

After finishing these three formulas, and exchange the order of integral and add, we get:

$$\tilde{y}(x_0) = \sum_{i=1}^{n} \left(e^{-d(x_0, x_i)} \int_{-\infty}^{\infty} y e^{-d(y, y_i)} dy \right) / \sum_{i=1}^{n} \left(e^{-d(x_0, x_i)} \int_{-\infty}^{\infty} e^{-d(y, y_i)} dy \right) \tag{3.17}$$

After calculation, we get:

$$\tilde{y}(x_0) = \sum_{i=1}^{n} y_i e^{-d(x_0, x_i)} / \sum_{i=1}^{n} e^{-d(x_0, x_i)} \tag{3.18}$$

Thus it can be seen that the forecast value $\tilde{y}(x_0)$ is the weighted sum of all the training sample of the dependent variable value y_i and its weights $e^{-d(x_0, x_i)}$. When the smooth factor σ is very big, even $d(x_0, x_i)$ tends to zero, $\tilde{y}(x_0)$ is similar to the average of all samples of the dependent variable. On the contrary, when the smooth factor tends to zero and $\tilde{y}(x_0)$ is very close to training sample, when the points which need to predict are included in the training sample, the predicted value we obtain from the upper formula is very close to the dependent variable of the samples. And when we get the value σ appropriate, all the dependent variable y_i of the training sample are taken into consideration. And the dependent variables corresponding are near some sample points which are closer to predicted points endowed with more weights.

3.3. The Particle Filter Based on GRNN Sample Adjustment

Set X is one of the measurement value of random variable x, X_i also is the sample value of x. definite scalar function:

$$D_i^2 = (X - X_i)^T (X - X_i) \tag{3.19}$$

Define indicator set value as follows:

$$\hat{Y}(X) = \sum_{i=1}^{n} Y_i e^{-\frac{D_i^2}{2\sigma^2}} / \sum_{i=1}^{n} e^{-\frac{D_i^2}{2\sigma^2}} \tag{3.20}$$

Here, Y_i is the sample of Y, and σ is the sample probability width for the summation of each samples X_i and Y_i. In other word, it is the smooth factor. In the particle filtering algorithm, we can use GRNN adjusted sample value according to measuring value z_k. Because in any moment, likelihood function attributes are fixed, but its mathematical expression is not clear, we can obtain the result only though some discrete measuring value. Therefore, we can make it close to the likelihood function according to the mea-

suring value training network, and then use this network to adjust any samples.

Firstly, construct input vector and target vector to network to carry on the training. Through sampling the likelihood function to get a group of samples, n adjacent samples and their likelihood functions respectively form the input vector and target vector.

After training the network, we adjust the sample of the particle filter algorithm in the form of input vector. First of all, construct a n d vector $X_k^i = \left[x_k^i, x_k^i \pm j\Delta\right]$, $j\Delta < L(j = 1,\ldots,n/2)$, parameter L define the adjusting range; Then, converse $X_k^i \to h(X_k^i) - z_k$ as the after-training the GRNN input vector; Finally, through the instructions of network output vector, the sample X_k^i is replaced by $x_k^i \pm j\Delta$ the most appropriate points. After the adjustment of a series of sample more approximate importance density.

STEP1: initialization. $k = 0$, sampling $x_0^i \sim p(x_0)$, in other words, $p(x_0)$ according to distribution sampling to get $x_0, i = 1,2\ldots,N$.

STEP2: calculating importance weights. Set $k := k+1$, sampling

$$x_k^i \sim q(x_k \mid x_{0:k-1}^i, z_{0:k}), i = 1,2\ldots,N. \tag{3.21}$$

Using GRNN to adjust samples, and define GRNN input vector is x_k^i, $i = 1,2\ldots,N$, target vector is z_k, and then began to network training.

$$\hat{Y}(X) = \sum_{i=1}^{n} Y_i e^{\frac{\left[\left(z_k - x_k^i\right)^T \left(z_k - x_k^i\right)\right]}{2\sigma^2}} / \sum_{i=1}^{n} e^{\frac{\left[\left(z_k - x_k^i\right)^T \left(z_k - x_k^i\right)\right]}{2\sigma^2}} \tag{3.22}$$

Then, construct a n d vector $X_k^i \left[x_k^i, x_k^i \pm j\Delta\right]$, $j\Delta < L(j = 1,\ldots,n/2)$, parameter L define the adjusting range. Converse $X_k^i \to h(X_k^i) - z_k$ the GRNN input vector.

$$\hat{Y}(X) = \sum_{i=1}^{n} \hat{Y}(X) e^{\frac{\left[\left(z_k - x_k^i\right)^T \left(z_k - x_k^i\right)\right]}{2\sigma^2}} / \sum_{i=1}^{n} e^{\frac{\left[\left(z_k - x_k^i\right)^T \left(z_k - x_k^i\right)\right]}{2\sigma^2}} \tag{3.23}$$

Finally, through the instructions of the network output vector X_k^i, the sample is replaced by the most appropriate points $x_k^i \pm j\Delta$.

Calculating importance weights are as follows:

$$\omega_k^i = \omega_{k-1}^i p(z_k \mid X_{k-1}^i) = \omega_{k-1}^i \frac{p(z_k \mid x_k^i) p(x_k \mid x_{k-1}^i)}{q(x_k^i \mid x_{k-1}^i, z_k)}, \quad i = 1,2\ldots,N \tag{3.24}$$

Normalized importance weights for:

$$\omega_k^i = \omega_k^i / \sum_{i=1}^{N} \omega_k^i \tag{3.25}$$

STEP3: Re-sampling. If

$$N_{\text{eff}} = 1 / \sum_{i=1}^{N} \left(\omega_k^i\right)^2 < N_{\text{th}} \tag{3.26}$$

Re-sampling, and put the original weighted sample $\left\{x_{0:k}^i, \omega_k^i\right\}_{i=1}^N$ map into equivalent weighted sample $\left\{x_{0:k}^i, N^{-1}\right\}_{i=1}^N$.

STEP4: Output.

State estimation:

$$\hat{X}_k = \sum_{i=1}^{N} \omega_k^i X_k^i \tag{3.27}$$

Variance estimation:

$$P_k = \sum_{i=1}^{N} \omega_k^i (X_k^i - \hat{X}_k)(X_k^i - \hat{X}_k)^T \tag{3.28}$$

STEP5: Judge whether we reach the end, if so, introduces the algorithm; otherwise, return to STEP2.

4. Experiment and Result

In order to verify the performance of the algorithm NNISA-PF, we apply NNISA-PF and the general PF algorithm in robot localization process. Positioning process of the two algorithms is shown in the Figures below. Compared with ordinary PF algorithm, the introduction of NNISA increase the calculated amount of the algorithm, but its application largely over comes the impact of the incompleteness of prior knowledge of the measurement noise, and improves the positioning accuracy and convergence of the algorithm. The experimental result shows the effectiveness of this metIn advantage of the simulation plat form provided by Alberto Vale, we make a further

analysis of the results of robot localization. The trajectory of robot is from point A to point B, after all the marked points. In order to verify the effectiveness of the algorithm NNISA-PF of different number of sample sand the ordinary PF were applied to the mobile robot localization process. The figure shows the positioning results of ordinary PF algorithm with 300 samples, and its positioning deviation. The next figure shows the positioning results of algorithm with 100 samples and its positioning deviation.

As we can see from the upper figures, the positioning result of NNISA-PF algorithm is better than ordinary PF algorithm. The positioning accuracy of NNISA-PF algorithm with 100 samples can be achieved even more than the positioning accuracy of ordinary PF algorithm with 300 samples.

Conclusions

In the ordinary PF algorithm, since the sampling efficiency is very low and the weight is not accurate, the use of small sample cannot be correctly expressed in the system of the posterior probability density. In addition, the sample degradation is an unavoidable problem, despite re-sampling can eliminate sample degradation. But it will lead to samples effectiveness and lack of diversity, causing sample dilution. Moreover, as the iterations increase, sample dilution is more and more serious, even seriously undermining the effectiveness and diversity of the sample.

In order to make the PF algorithm effective on mobile robot localization, this paper presents the NNISA-PF algorithm combining neural network theory with particle filter, so that the samples effectiveness and diversity to be kept and reduce sample dilution in the re-sampling stage. The experimental results show that the improvement measures can effectively improve the performance of the algorithm, so that it enables them to maintain a reliable positioning.

Conflicts of Interests

The authors declare that there is no conflict of interests regarding the publication of this article.

References

1. Castellanos J, Neira J, Strauss O. Detecting high level features for mobile robot localization. In Proceedings of IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems. 1996, p.p. 611-618.
2. Batke M. and Gurvit L. s. Mobile robot localization using landmarks. IEEE Transactions on Robotics and Automation, 1997, 13(2), p.p.251-263.
3. Leonard J Jand Durrant-Whyte H F. Mobile robot localization by tracking geometric beacons. IEEE Transactions on Robtics and Automation, 1991, 7(3):376-382
4. Elouardi Abdelhafid, Lambert Alain, Merigot Alain. Efficient implementation of EKF-SLAM on a multi-core embedded system. IECON 2012-38th Annual Conference on IEEE Industrial Electronics Society, 2012, p.p.3049-3054.
5. Suzuki T, Amano Y, Hashizume T. Development of a SIFT based monocular EKF-SLAM algorithm for a small unmanned aerial vehicle. SICE Annual Conference, 2011, p.p. 1656-1659.
6. Casarrubias-Vargas H, Petrilli-Barcelo A, Bayro-Corrochano E. EKF-SLAM and Machine Learning Techniques for Visual Robot Navigation. 2010 20th International Conference on Pattern Recognition, 2010, p.p. 396-399.
7. W Bugard, D Fox, D Hennig, T Schmidt. Estimating the absolute position of a mobile robot using positon probability grids. Proc 14th National Conference on Artificial Intelligence. Portland: AAAI Press, 1996, p.p. 896-901.
8. D Fox, W Burgard, S Thrun. Markov localization for mobile robots in dynamic environments. Journal of Artificail Intelligence Research, 1999, 11(1), p.p.391-427.
9. I.Arasaratnam, S.Haykin, R.J.Ellion. Discrete-Time Nonlinear Filtering Algorithms Using Gauss-Hermite Quadrature. Proceedings of the IEEE, 2007, 95(5), p.p.953-977.
10. Roumeliotis S I, Bekey G A. Bayesian estimation and Kalman filtering: A unified framework for mobile robot localization. In Proceedings of IEEE International Conference on Robotics and Automation. San Francisco, California, 2000, p.p.2985-2992.
11. Han Yueqi, Wu Zhuhui, Wang Yunfeng, Cheng Xiaoping, Qian Jin. The Successive Analysis Method in Monte Carlo H Filter and Numerical Experiments. 2011 International Conference on Computer Engineering and Management Sciences (ICM). 2011, p.p. 63-67.
12. S Lenser, MVeloso. Sensor resetting localization for poorly modeled mobile robots. Int Conf on Robotics and Automation. San Francisco: IEEE Press, 2000, p.p. 1225-1232.
13. A Doucet, N de Freitas, N Gordon, et al. Sequential Monte Carlo in Practice. Springer Verlag, 2001.

14. Kang Xiao, Li Kejie, Zhu Wei. A new localization method for mobile robots using Genetic Simulated Annealing Monte Carlo Localization. 2011 International Conference on Mechatronics and Automation, 2011, p.p. 1780-1785.
15. Gutmann J S, Burgard W, Fox D, Konolige K. An experimental comparison of localization methods. In Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 1998.
16. Gutmann J S, Fox D. An experimental comparison of localization methods continued. In Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 1998.

The Development of a Scale to Measure Smartphone Dependency

Hsin-I Chen

Chinese Culture University,55, Hwa-Kang Road, Yang-Ming-Shan, Taipei 11114, Taiwan

Corresponding author is Hsin-I Chen

Abstract
The purpose of this study is to develop a scale to measure users' social interaction on smartphone toward dependency. Smartphone dependency refers to users' compulsive behaviors resulting from overuse of the social interaction function pertaining to smartphone. This article constructs the smartphone dependency scale and its development, reliability, exploratory factor analyses, structure, and validity. The research instrument was pilot tested and necessary modifications were made. The reliability and validity of the instrument are determined using Exploratory Factor Analysis. Further, Confirmatory Factor Analysis using AMOS 20.0 is carried out for the scale. Confirmatory Factor Analysis model-fit indicators were found acceptable in this study. A five-dimension scale of smartphone dependency was developed. The five subscales are social connection of smartphone, compulsive behavior, salient behavior, pleasure to engage interpersonal interaction, and withdrawal.
Keywords: SMARTPHONE DEPENDENCY, SCALE DEVELOPMENT.

1. Introduction

According to ITU (International Telecommunication Union), the number of mobile phone users worldwide will reach a record high of 7 billion by the end of 2014, with the total market penetration rate being at a high of 96% [1]. Specifically in America, according to Pew Research Center's Internet & American Life Project, 91% of the population used mobile phone, with smartphone popularity rate being as high as 56% [2]. Presently, smartphone functions as a micro computer. People communicate with others by calling, text messaging and emailing. Ad-