Box Particle Filtering for SLAM with Bounded Errors

Peng Wang, Philippe Xu, Philippe Bonnifait, Jianwen Jiang

Box Particle Filtering for SLAM with Bounded Errors

Peng Wang, Philippe Xu, Philippe Bonnifait, and Jianwen Jiang

Abstract— **This paper proposes a set-membership based method for Simultaneous Localization and Mapping (SLAM). A Box Particle Filter (BPF) is exploited and improved to estimate robot states and feature positions, with interval Constraint Propagation (CP) to reduce box sizes and decrease uncertainties in estimates. Buffers are also used to get q-satisfied results when empty estimations arise, on the one hand. On the other hand, through buffer contraction, historical estimations can be improved. Illustrations of the proposed method are given over simulations and experiments, with comparisons with a Particle Filter (PF) based method. The results show that the proposed method can reach the same SLAM accuracy as PF based method with much fewer particles. Moreover, this approach is more robust to high level uncertainties.**

I. INTRODUCTION

In this paper, BPF with CP and buffers is applied to do SLAM with feature maps. According to [1], SLAM involves a front end and a back end. The front end mainly deals with sensor perceiving and signal processing, while the back end mixes geometry, graph theory, optimization, and probabilistic theory to construct a model of the environment and concurrently maintain consistent robot state estimations. Mainstream SLAM solutions are either solely probabilistic estimation based techniques (EKF SLAM [2], FastSLAM [3]), or a combination with other theories (graph SLAM [4], semantic SLAM [5]). Each solution can perform well under certain hypotheses where the a priori noise probabilistic distribution is known.

Set-membership based SLAM is another solution. Intervals or boxes are used as basic operands which drastically reduces the adverse effects of non-linearity and the prior distribution is no longer a requirement. The first set-membership based SLAM algorithm was proposed in [6] and extended in [7] with a matching step. As measurements are not directly represented as boxes, an approximation step is used to eliminate the effects of non-convex regions, which leads to extra computing burden. But consistent and guaranteed results were shown in both papers. Porta [8] formalized the SLAM problem as a typical kinematic problem. Therefore, the constraints over robot poses and landmarks could be well modeled. Experimental results showed that all valid solutions could be maintained, even though there was still an overestimation issue. Jaulin first [9] introduced CP into set-membership based SLAM applied to a submarine robot. An envelope of the robot trajectory and an interval feature map were built. But as Jaulin mentioned, with strong outliers, CP may lead to empty sets, which could potentially cause the loss of robot states. More set-membership based SLAM methods can be found in [10].

Though with the advantages of generality, simplicity and reliability, set-membership based SLAM methods still suffer from drawbacks like overestimation, computational cost, and empty sets occurring in the process. To deal with these problems, BPF that has been mainly used for positioning comes as a potential solution.

BPF is a set-membership variant of the traditional PF. It was first introduced in [11] to handle interval data by using interval analysis and constraint satisfaction techniques. The application of BPF in vehicle global localization showed that 10 particles reached almost the same localization accuracy as the traditional PF with 3000 particles. Since then, many researchers applied BPF in different areas. In [12], the Bernoulli BPF was introduced and applied in tracking a single target. By carefully designing, Bernoulli BPF can track the target accurately and is computationally more efficient compared to Bernoulli PF. A crowd tracking BPF was proposed in [13]. They generalized the conventional way of computing the likelihood function when the state vector consists of kinematic states and extent parameters. Experimental results showed that the tracking accuracy and computational time of BPF were quite competitive. The authors in [14] applied BPF to a vehicle positioning problem with GPS signals and a 3D map. Together with SIVIA and q-relaxed technique, good positioning results were obtained even with GPS signal losses. In [15], the authors introduced a regularization step after resampling and correction steps to improve the robustness of the traditional BPF. Its application in terrain navigation showed that with 200 particles (compared to 5000 particles in PF), the method could significantly improve the localization accuracy.

In this paper, we show how BPF can be applied to solve SLAM problems. This approach can report the same accuracy as classical PF, while using fewer particles and being much more robust to uncertainty. In addition, we introduce a way to extend BPF with buffers, which can reduce box sizes furthermore.

The paper is organized as follows. In section II, the problem to be solved is formulated based on intervals. The BPF SLAM algorithm is introduced in section III. Section

This work was supported by NSFC (61703387), Anhui Provincial Natural Science Foundation (1708085QF159) and the Fundamental Research Funds for the Central Universities (BJ2100100039)

P. Wang is with the Department of Automatic Control and Systems Engineering, University of Sheffield, S10 2TN Sheffield, UK Peng.Wang@Sheffield.ac.uk

Ph. Xu and Ph. Bonnifait are with Sorbonne Universités, Université de Technologie de Compiégne, CNRS, Heudiasyc, 60 203 Compiégne, France {philippe.xu, philippe.bonnifait}@hds.utc.fr

J. Jiang is with the Department of Automation, University of Science and Technology of China, 230027 Hefei, China jjwen@mail.ustc.edu.cn

IV provides extension of BPF with buffer contraction. In Section V, two sets of experimental results are provided for comparison between the proposed algorithm and PF based method. Section VI concludes the paper.

II. PROBLEM DESCRIPTION

A. Dynamic Model

Classical SLAM problem can be formalized as

$$\begin{cases} \mathbf{x}_{k+1} = \mathbf{f}(\mathbf{x}_k, \mathbf{u}_k, \mathbf{m}_k), \\ \mathbf{y}_k^j = \mathbf{g}(\mathbf{x}_k, \mathbf{l}_k^j, \mathbf{n}_k), \end{cases} \tag{1}$$

where $\mathbf{f}$ and $\mathbf{g}$ are possibly nonlinear dynamic and measurement models, $\mathbf{x}_k$ the robot state vector, $\mathbf{u}_k$ the input vector, $\mathbf{y}_k^j$ the observation vector, $\mathbf{l}_k^j$ the j-th observed feature stored in a feature map. $\mathbf{m}_k$ and $\mathbf{n}_k$ are respectively the system and observation noises.

In classical SLAM methods, all variables involved in $\mathbf{f}$ and $\mathbf{g}$ are real numbers. But either in BPF or CP, the fundamental operands are intervals or boxes. Thus, (1) is rewritten as

$$\begin{cases} [\mathbf{x}_{k+1}] = [\mathbf{f}]([\mathbf{x}_k], \mathbf{u}_k, [\mathbf{m}_k]), \\ [\mathbf{y}_k^j] = [\mathbf{g}]([\mathbf{x}_k], [\mathbf{l}_k^j], [\mathbf{n}_k]), \end{cases} \tag{2}$$

in which, $[\mathbf{f}]$ and $[\mathbf{g}]$ are inclusion functions, $[\mathbf{x}_{k+1}]$ and $[\mathbf{x}_k]$ are box state vectors, $[\mathbf{y}_k^j]$ is the box observation, $[\mathbf{l}_k^j]$ is the j-th box feature position, $[\mathbf{m}_k]$ and $[\mathbf{n}_k]$ are box noises. Sizes of the boxes at the initial stage can be arbitrarily large in order to include all feasible values. Note that as $\mathbf{u}_k$ is normally known, we do not apply interval on it. But it can be replaced by an interval if needed.

Based on (2), the objective is to exploit BPF to estimate robot states and feature positions. The size of boxes is reduced by using CP while all feasible values are maintained.

B. PF Based SLAM

PF is one of the probabilistic estimation based SLAM method, and FastSLAM [3] is among the most popular featured map based algorithms. The key idea of FastSLAM lies in using a set of particles $\mathcal{X} = \{\langle x^{[i]}, w^{[i]} \rangle\}_{i=1,\cdots,N}$ to ultimately estimate the posterior $p(x_{0:k}|u_{1:k}, z_{1:k})$, where $x = (x_{0:k}, \mathbf{l}_{1:M})^T$ with $\mathbf{l}_{1:M} = [\mathbf{l}_k^1, \cdots, \mathbf{l}_k^M]$ the state vector, and $u_{1:k}$ and $z_{1:k}$ are the corresponding inputs and measurements.

The estimation of the joint distribution of $x_{0:k}$ and $\mathbf{l}_{1:M}$ are complex and computationally consuming. Murphy introduced factorization of the SLAM posterior in [16], which enabled computing posterior distributions of $x_{0:k}$ and $\mathbf{l}_{1:M}$ as follows:

$$\begin{aligned} p(x_{0:k}, \mathbf{l}_{1:M}|u_{1:k}, z_{1:k}) = \\ p(x_{0:k}|u_{1:k}, z_{1:k}) p(\mathbf{l}_{1:M}|u_{1:k}, x_{0:k}), \end{aligned} \tag{3}$$

which now acts as a basic paradigm for PF based SLAM. We also make the same independent assumption in BPF as in FastSLAM.

III. BPF BASED SLAM

A. Basic Interval Arithmetic

1) Interval and Inclusion Function: The key contribution of interval arithmetic lies in that intervals become the basic operands rather than single numbers. An interval is normally defined as $[x] = [\underline{x}, \overline{x}] = \{x \in \mathbb{R} | \underline{x} \leq x \leq \overline{x}\}$. Given intervals $[x]$, $[y]$, and an operator $\diamondsuit \in \{+, -, \cdots, /\}$, $[x] \diamondsuit [y]$ is defined as the smallest interval that includes all feasible values for $x \diamondsuit y$. For example, consider a box $[\mathbf{x}] = ([x], [y], [z])$. The center of the box can be computed by $mid([\mathbf{x}]) = (\hat{x}, \hat{y}, \hat{z})$, with $\hat{x} = (\underline{x} + \overline{x})/2$, $\hat{y} = (\underline{y} + \overline{y})/2$, and $\hat{z} = (\underline{z} + \overline{z})/2$. The volume of the box is computed by $||[\mathbf{x}]|| = ||[x]|| \cdot ||[y]|| \cdot ||[z]||$, with $||[x]|| = \overline{x} - \underline{x}$.

Intervals can be applied to elementary functions or composite functions derived from combination of elementary functions and basic operators with the help of inclusion function. Consider a function $\mathbf{f} : \mathbb{R}^n \to \mathbb{R}^m$, an inclusion function $[\mathbf{f}]$ is defined as $\mathbf{f}([x]) \subset [\mathbf{f}]([\mathbf{x}])$, $\forall [\mathbf{x}] \subset \mathbb{R}^n$.

2) Constraint Satisfaction Problem (CSP): Though box guarantees that no feasible values are removed, it brings potential overestimation problem. CSP is usually used to reduce box size. When there are constraints among components of a vector $\mathbf{x}$, CSP can be described as finding a solution set $X = \{\mathbf{x} \in [\mathbf{x}] \, | \mathbf{h}(\mathbf{x}) = 0\}$, where $[\mathbf{x}]$ is the feasible domain and $\mathbf{h}(\mathbf{x}) = 0$ is the constraint. A contractor $\mathbb{C}$ is used to reduce the size of $[\mathbf{x}]$ and get $[\mathbf{x}_c] = \mathbb{C}([\mathbf{x}])$, where $X \subset [\mathbf{x}_c] \subset [\mathbf{x}]$ [14]. The forward-backward contractor is the most popular contractor. Normally, whether the box size is reduced significantly or not depends on the constraints and innovations used to finish contraction.

3) Q-satisfied Intersection: While computing the intersection of multiple boxes, one of the risk is resulting with empty box. Q-satisfied intersection [17], which exhaustedly finds the non-empty intersection of the maximum number of boxes, is a dual operation of q-relaxed intersection that is used in sub-paving background. Given m boxes $[\mathbf{x}]_1, \cdots, [\mathbf{x}]_m$ with the constraints that at least one of them is non-empty, the q-satisfied intersection of them is defined as

$$[\mathbf{x}] = \bigcap_{j=0}^{q} [\mathbf{x}]_j, \tag{4}$$

in which, $q = \max\{1, \cdots, m\}$ *s.t.* $\bigcap_{j=0}^{q} [\mathbf{x}]_j \neq \emptyset$.

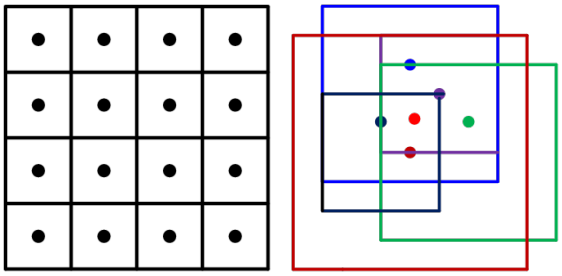
For a given set of m boxes $\{[\mathbf{x}]_1, \cdots, [\mathbf{x}]_m\}$, let q be the maximum number such that there exists at least one subset of size q of boxes for which the intersection is not empty,

$$q = \max \left\{ |A| \,\middle|\, A \subseteq \{1, \ldots, m\}, \bigcap_{j \in A} [\mathbf{x}]_j \neq \emptyset \right\}. \tag{5}$$

Let $A_1, \ldots, A_k$ be the resulting $k > 0$ subsets of size q. We define the q-satified intersection $[\mathbf{x}]$ as the box hull of the set $\{A_1, \ldots, A_k\}$, i.e., the smallest box containing this set :

$$[\mathbf{x}] = \text{boxhull}\left(\{A_1, \ldots, A_k\}\right) = \bigcap^{\{q\}} [\mathbf{x}]_{1,\ldots,m}. \tag{6}$$

The q-satisfied intersection operator is noted $\bigcap^{\{q\}}$.



(a) (b)

Fig. 1. Particle initialization: (a) Traditional BPF; (b) BPF in this paper

B. The BPF Based SLAM Algorithm

SLAM problem described by (2) can be briefly represented as iteratively updating the robot state and feature positions $\{[\mathbf{x}_k], \mathbf{L}_k = [[\mathbf{l}_1], [\mathbf{l}_2], \cdots, [\mathbf{l}_k]]\}$ based on past robot state, input and observations $\{[\mathbf{x}_{k-1}], \mathbf{u}_k, \mathbf{Z}_k = [[\mathbf{z}_1], [\mathbf{z}_2], \cdots, [\mathbf{z}_k]]\}$. But as the propagation continues, box sizes can either be very large or result in empty sets which causes a failure of the SLAM.

By combining BPF and CP together, large boxes can be effectively reduced and the diversities in box sizes provide a redundancy mechanism to avoid empty sets, resulting in an accurate and guaranteed SLAM result.

1) Initialization: In traditional BPF, the algorithm is initialized by performing a sub-paving of a box, whose size can be determined by an initial covariance matrix. Each box represents a particle. The intersection of the initial particle boxes is empty. Such initialization can easily lead to particle vanishment, i.e., only small boxes around true states are informative. In this paper, we first generate N Gaussian points with an expectation equal to the initial state expectation. Centered on these points, N boxes are generated. The size of each box should be large enough to cover initial states, which is reasonable and achievable. The weight of each particle is initialized as $w_0^i = 1/N$, $i \in \{1, 2, \cdots, N\}$. Initialization schemes of the traditional BPF and the one proposed in this paper is shown in Fig. 1.

2) Prediction: For particle i , suppose that the robot state at time k is $[\mathbf{x}_k^i]$. Using the dynamical model, we can get a prediction of the robot state at $k+1$ as

$$[\mathbf{x}_{k+1}^{i,0}] = [\mathbf{f}]([\mathbf{x}_k^i], \mathbf{u}_k), \tag{7}$$

in which, the superscript 0 is used to distinguish the predicted state to these obtained from contraction, which are denoted by superscript j.

3) Measurement Correction: Suppose that $m = m_n + m_e$ features are observed. After feature association, there are m_n new features and m_e features that are already registered in the current map. For the existing features $\{[\mathbf{l}_{k+1}^{i,j}]\}_{j=1}^{m_e}$, the corresponding measurements are $\{[\mathbf{z}_{k+1}^{i,j}]\}_{j=1}^{m_e}$. With $[\mathbf{x}_{k+1}^{i,0}]$ and the measurement model in (2), m_e feature predictions are generated as given by (8):

$$\{[\mathbf{y}_{k+1}^{i,j}]\}_{j=1}^{m_e} = [\mathbf{g}]([\mathbf{x}_{k+1}^{i,0}], \{[\mathbf{l}_{k+1}^{i,j}]\}_{j=1}^{m_e}). \tag{8}$$

Intersections of $\{[\mathbf{z}_{k+1}^{i,j}]\}_{j=1}^{m_e}$ and $\{[\mathbf{y}_{k+1}^{i,j}]\}_{j=1}^{m_e}$ are innovations, which are defined as

$$\{[\mathbf{r}_{k+1}^{i,j}]\}_{j=1}^{m_e} = \{[\mathbf{z}_{k+1}^{i,j}] \cap [\mathbf{y}_{k+1}^{i,j}]\}_{j=1}^{m_e}. \tag{9}$$

For each set $\{[\mathbf{x}_{k+1}^{i,0}], [\mathbf{r}_{k+1}^{i,j}]\}$, there exists a CSP given as

$$\{\mathbf{x}_{k+1}^{i,j} \in [\mathbf{x}_{k+1}^{i,0}], \mathbf{l}_{k+1}^{i,j} \in [\mathbf{l}_{k+1}^{i,j}] | \mathbf{g}(\mathbf{x}_{k+1}^{i,j}, \mathbf{l}_{k+1}^{i,j}) - \mathbf{r}_{k+1}^{i,j} = 0\}. \tag{10}$$

We solve this problem with a forward-backward contractor and get m_e states after contraction. Together with $[\mathbf{x}_{k+1}^{i,0}]$, we get $m_e + 1$ box estimates of $\mathbf{x}_{k+1}^i$. Intuitively, a reasonable estimation should be

$$[\mathbf{x}_{k+1}^i] = \bigcap_{j=0}^{m_e} [\mathbf{x}_{k+1}^{i,j}]. \tag{11}$$

But because the influence of noise, $[\mathbf{x}_{k+1}^i]$ could be empty. We used q-satisfied approach to get a non-empty box.

$$[\mathbf{x}_{k+1}^i] = \bigcap_{j=0}^{q} [\mathbf{x}_{k+1}^{i,j}], \tag{12}$$

where $q = \max\{1, \cdots, m_e + 1\}$ *s.t.* $\bigcap_{j=0}^{q} [\mathbf{x}_{k+1}^{i,j}] \neq \emptyset$.

4) Likelihood and Weight Updates: In traditional BPF, likelihood is defined as the ratio of the state box sizes after and before contraction. In a SLAM scenario, we also have to consider the size changes of the features boxes. Therefore, the likelihood A^i is computed by:

$$A^i = A_r^i \prod_{j=1}^{m_e} A_f^{i,j}. \tag{13}$$

in which, $A_r^i = |[\breve{\mathbf{x}}_{k+1}^i]|/|[\mathbf{x}_{k+1}^i]|$ is the likelihood component decided by robot state, $A_f^{i,j} = |[\breve{\mathbf{l}}_{k+1}^{i,j}]|/|[\mathbf{l}_{k+1}^{i,j}]|$ is the likelihood component decided by feature j, and $[\breve{\mathbf{x}}_{k+1}^i]$ and $[\breve{\mathbf{l}}_{k+1}^{i,j}]$ are the robot state and feature position boxes after contraction.

Particle weight is then updated by $w_{k+1}^i = w_k^i A^i$.

5) State Estimation: The weight normalization and resampling steps are the same as traditional BPF. By nature, interval analysis based methods do not provide point estimates. In order to compare to PF, we use (14) to compute a point estimate,

$$\hat{\mathbf{x}}_k = mid\Big(\bigcup_{i=1}^{N} [\mathbf{x}_k^i]\Big). \tag{14}$$

We use (14) because the resulted point contains more information about innovations than the weighted centre in traditional BPF. The reason is that in a weighted centre way, boxes with small size changes before and after contraction get higher weights, therefore their centres contribute more in the final point estimate. But small size changes implicate less benefits from innovation. As innovations are the most important (even the only) way to shrink box sizes, it is better to properly take advantages of innovations. (14) tends to "uniformaly" integrating information from each box, which intuitively contains more information from innovations.

The above steps constitute the main body of the BPF SLAM algorithm, which is shown in algorithm 1.

Algorithm 1 BPF SLAM Algorithm

1: Initialization
2: Set $k = 0$ and generate N boxes $\{[\mathbf{x}_k^i]\}_{i=1}^N$ with the same weights equal to $1/N$.
3: **for** $i = 1, \cdots, N$ **do**
4: $\left[\mathbf{x}_{k+1}^{i,0}\right] = [\mathbf{f}]\left([\mathbf{x}_k^i], \mathbf{u}_k^i\right)$ //prediction
5: $\{[\mathbf{z}_{k+1}^{i,j}]\}_{j=1}^m$ //Measurement, $m = m_e + m_n$
6: **if** $m_e > 0$ **then**
7: $[\mathbf{x}_{k+1}^i] \leftarrow$ q-satisfied estimation //Eq.(8) - (12)
8: $A^i = A_r^i \prod_{j=1}^{m_e} A_f^{i,j}$ //likelihood
9: $w_{k+1}^i = w_k^i A^i$ //weight update
10: $w_{k+1}^i \leftarrow w_{k+1}^i \Big/ \sum_{j=1}^N w_{k+1}^j$
11: $\hat{\mathbf{x}}_k = mid(\bigcup_{i=1}^N [\mathbf{x}_k^i])$ //state estimation
12: $N_{eff} = 1 \Big/ \sum_{i=1}^N \left(w_k^i\right)^2$, if $N_{eff} < N_{th}$, resample.
13: **if** $m_n > 0$ **then**
14: register m_n new features $\{[\mathbf{l}_{k+1}^{i,j}]\}_{j=1}^{m_n}$.
15: (OPTIONAL BUFFER)
16: $k = k + 1$, goto step2 until $k = k^{end}$.

IV. EXTENSION WITH CONTRACTION ON BUFFERS

The main advantage of BPF SLAM is to have guaranteed results. However, the resulting boxes may remain too large. We propose to buffer a certain amount of historical data and do CP in the buffer to reduce the sizes of the boxes.

Robot state box $[\mathbf{x}_k^i]$, feature position boxes $\{[\mathbf{z}_k^j]\}_{j=1}^{m_e^k}$ and input $\mathbf{u}_k$ need to be buffered. Therefore, each particle i maintains three buffers to store historical data.

$B_r^i = \{[\mathbf{x}_{k-s_c+1}^i], [\mathbf{x}_{k-s_c+2}^i], \cdots, [\mathbf{x}_k^i]\}$ is the robot state buffer at k, where s_c is the buffer size.

$B_f^i = \{[\mathbf{Z}_{k-s_c+1}^i], [\mathbf{Z}_{k-s_c+2}^i], \cdots, [\mathbf{Z}_k^i]\}$ is a feature position buffer, in which $\mathbf{Z}_t^i = [\mathbf{z}_t^{i,j}]_{j=1}^{m_e^t}$ are observations of existing features at t, and $s_f = \sum_{t=k-s_c+1}^{k} m_e^t$ is the total number of feature observations. It should be noted that the size of B_f^i in this paper is considered as s_c rather than s_f.

$B_u = \{\mathbf{u}_{k-s_c+1}, \mathbf{u}_{k-s_c+2}, \cdots, \mathbf{u}_k\}$ is the input buffer at k with the size of s_c, which is the same for all particles.

As SLAM is a dynamic process, the amount of data in a buffer increases step by step until the buffer reaches its maximum size s_c. When the buffer is full, the oldest s_o data in each buffer are discarded. Theoretically, s_o can be any value in $\{1, 2, \cdots, s_c\}$. The final value should be decided according to real application requirements.

Buffer size is a very important parameter to be determined. We found out that the size of a buffer is not proportional to the amount of box size reduction. That is because as the buffer size becoming larger, the risk of inconsistent contraction results increases. So the buffer size should not be too large, which benefits both consistent contraction and computation speed.

With the buffers, two CSPs can be built up. The state to state CSP equation is given as (15),

$$\{\mathbf{x}_{k+1}^i \in [\mathbf{x}_{k+1}^i], \mathbf{x}_k^i \in [\mathbf{x}_k^i] | \mathbf{f}(\mathbf{x}_k^i, \mathbf{u}_k^i) - \mathbf{m}_{k+1}^i = \boldsymbol{\epsilon}_x\}, \quad (15)$$

and the state to feature CSP equation is shown as (16),

$$\{\mathbf{x}_k^i \in [\mathbf{x}_k^i], \mathbf{l}_k^{i,j} \in [\mathbf{l}_k^{i,j}] | \mathbf{g}(\mathbf{x}_k^i, \mathbf{l}_k^{i,j}) - \mathbf{z}_k^{i,j} = \boldsymbol{\epsilon}_y\}, \quad (16)$$

in which, $\boldsymbol{\epsilon}_x$ and $\boldsymbol{\epsilon}_y$ are expected errors. Contraction in the buffer can then be done by using the forward-backward contractor. It should be noted that during contraction, empty sets could rise up. It happens partly because the size of the current box is too small. Therefore, when empty sets arise, we enlarge box sizes to decrease the possibility of resulting in empty sets. The reason why we can do this is because the enlarged boxes can later be decreased during further contraction. Details are given in algorithm 2.

Now, the buffer extended version of BPF SLAM is to add algorithm 2 to the end of algorithm 1 (step 15).

Algorithm 2 On Buffer Contraction Algorithm

Input: B_r^i, B_f^i, B_u, loop times n
while $loop < n$ **do**
  for $t = k - s_c + 1 : 1 : k - 1$ **do**
    $[\tilde{\mathbf{x}}_{t+1}^i] - [\mathbf{f}]([\tilde{\mathbf{x}}_t^i], \tilde{\mathbf{u}}_t, [\mathbf{m}_t^i]) = [\boldsymbol{\epsilon}_x]$ //CtcFwd
  for $t = k - 1 : -1 : k - s_c + 1$ **do**
    $[\tilde{\mathbf{x}}_{t+1}^i] - [\mathbf{f}]([\tilde{\mathbf{x}}_t^i], \tilde{\mathbf{u}}_t, [\mathbf{m}_t^i]) = [\boldsymbol{\epsilon}_x]$ //CtcBwd
  for $t = k - s_c + 1 : 1 : k$ **do**
    $[\mathbf{g}]([\tilde{\mathbf{x}}_t^i], \{[\tilde{\mathbf{l}}_t^{i,j}]\}_{j=1}^{m_e^t}, [\mathbf{n}_t^i]) - \{[\mathbf{z}_t^{i,j}]\}_{j=1}^{m_e^t} = [\boldsymbol{\epsilon}_x]$
for $t = k - s_c + 1 : 1 : k$ **do**
  if $[\tilde{\mathbf{x}}_t^i] \neq \emptyset$ **then**
    $[\mathbf{x}_t^i] = [\tilde{\mathbf{x}}_t^i]$
  else
    $[\mathbf{x}_t^i] = \alpha[\mathbf{x}_t^i]$ //α is a constant bigger than 1
for $t = t_s : 1 : t_f$ **do** //($t_f - t_s$ is the related feature number)
  if $[\tilde{\mathbf{l}}_t^i] \neq \emptyset$ **then**
    $[\mathbf{l}_t^i] = [\tilde{\mathbf{l}}_t^i]$
  else
    $[\mathbf{l}_t^i] = \beta[\mathbf{l}_t^i]$ //β is a constant bigger than 1
return $\{[\mathbf{x}_h^i]\}_{h=k-s_c+1}^{k}$, $\{[\mathbf{l}_h^i]\}_{h=t_s}^{t_f}$

V. EXPERIMENTS AND ANALYSES

A. Dynamic Model and Measurement Model

In this paper, we consider a SLAM scenario in which a wheeled mobile robot is equipped with a dead reckoning system and a laser scanner. The dynamic model can then be written as

$$\begin{cases} x_{k+1} = x_k + T \cdot v_k \cdot \cos\left(\theta_k + T \cdot \omega_k / 2\right) + m_{xk} \\ y_{k+1} = y_k + T \cdot v_k \cdot \sin\left(\theta_k + T \cdot \omega_k / 2\right) + m_{yk} \\ \theta_{k+1} = \theta_k + T \cdot \omega_k + m_{\theta k} \end{cases} \quad (17)$$

where $[v_k, \omega_k]^T = \mathbf{u}_k$ are inputs, $[x_k, y_k, \theta_k]^T = \mathbf{x}_k$ constitute the robot state vector, $[m_{xk}, m_{yk}, m_{\theta k}]^T = \mathbf{m}_k$ are the corresponding noises, and T is the sampling period.

Suppose that the position vector of a feature is denoted as $\mathbf{l}_i = [l_x^i, l_y^i]^T$, $i \in \mathbb{N}^+$, then the measurement model can be written as

$$\begin{cases} d_k = \sqrt{\left((x_k - l_x^i)^2 + \left(y_k - l_y^i\right)^2\right)} + n_{di} \\ l_\theta^k = \text{atan2}\left(y_k - l_y^i, x_k - l_x^i\right) - \theta_k + n_{\theta i} \end{cases} \tag{18}$$

in which, d_k is the distance between the robot and feature $\mathbf{l}_i$, l_θ^k is the azimuth angle of $\mathbf{l}_i$ in the robot coordinate, $[n_{di}, n_{\theta i}] = \mathbf{n}_k$ are the corresponding measurement noises.

B. Experiments and Analyses

The proposed BPF and CP based SLAM algorithm (BPF-CP) has been tested in simulation and in indoor experiments. A classical PF SLAM has been implemented for comparison purposes. A laptop with ubuntu 16.04 LTS OS, 4GB RAM, and 2.20GHz*4 processor was used to process data.

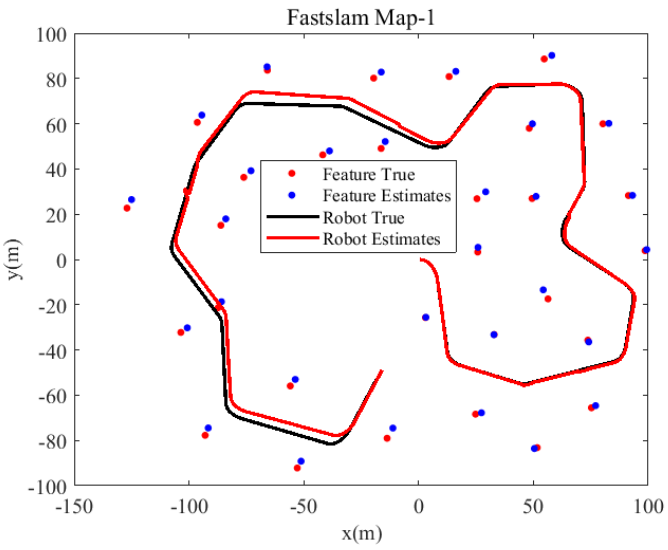
1) Simulation: A simulated feature map (Map-1) from [18] was adopted. The map was originally used to implement fastslam 2.0, which is known as one of the most popular PF based SLAM method. The results are compared with fastslam 2.0 directly using codes from [19]. We also used an interval operation toolbox pyibex [20].

In order to compare BPF-CP to PF SLAM, we set the parameters of both methods so that the resulting root mean square error (RMSE) is about the same level. Table I shows the parameters and the resulting RMSEs. We can see that with almost similar accuracy, PF needs very small noise variances, while BPF can cope with much larger uncertainty. We can also note BPF-CP needs much smaller number of particles.

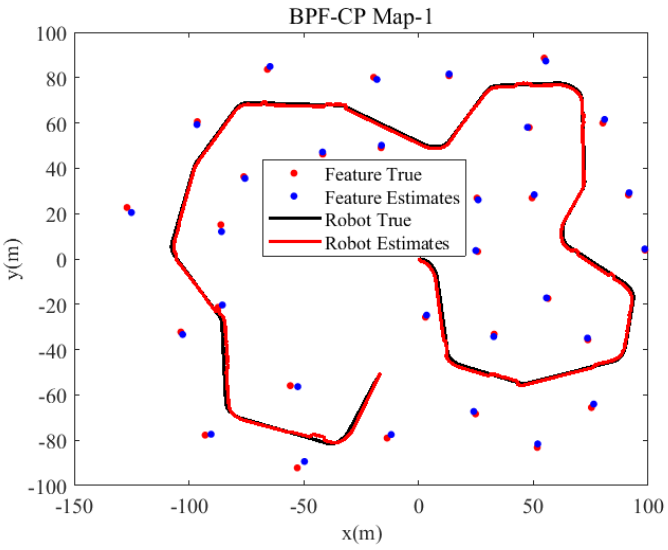
Fig. 2(a) and Fig. 2(b) show the final results of BPF-CP and fastslam 2.0, respectively. Fig. 2(c) shows the final boxes. We can conclude that most box sizes are reduced while the true values stay inside the boxes which shows the very good consistency of the method. We can also see that the size of the state boxes can increase and decrease. Both q-satisfied technique and the enlargement when empty boxes arise are responsible for increase, but boxes normally decrease when contraction is applied.

2) Real experiments: To test BPF-CP algorithm using real data, we conducted gmapping in the teaching building No.3 (about 50 m by 35 m) of USTC by using a Pioneer 3-DX mobile robot. The robot was equipped with a dead reckoning system to record linear and angular velocities, and a SICK laser scanner producing 541 points with a FOV of 270°. We then randomly sampled 1500 point features to build up Map-2. Feature locations were decided by the grid map built up by gmapping. The velocities and scans were kept the same. We processed these data by using our algorithm and PF based algorithm. Table II shows the parameters and RMSEs of the two methods. The results show that we reach the same conclusion as in the simulation. The final SLAM results are shown in Fig. 3.

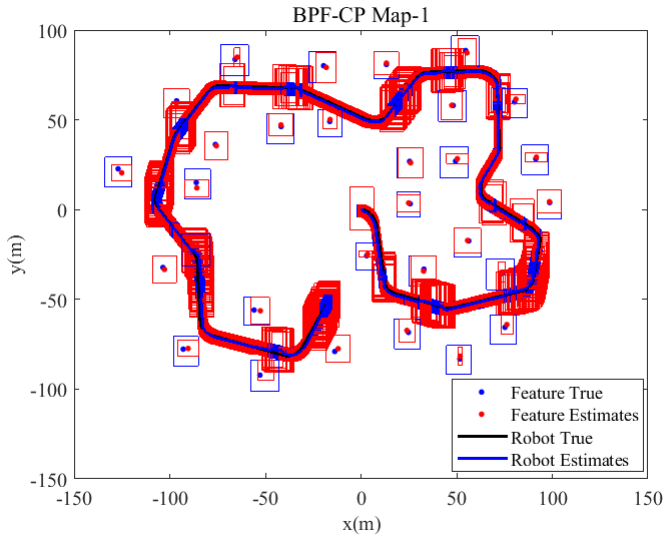
As there are many features, boxes are overlapping. We provide the box size reductions of the robot states in Fig. 4. One can notice that position and heading show significant



(a)



(b)



(c)

Fig. 2. Simulation SLAM results: (a) Results by fastslam; (b) Results by BPF-CP; (c) Final boxes with blue and red for before and after contraction

TABLE I
SIMULATION PARAMETERS OF THE TWO ALGORITHMS

BPF-CP	values	PF	values
state box	$50\cdot \mathbf{U}([x_r],[y_r],[\theta_r])^a$	control cov.	$[0.3^2 \quad 0; \; 0 \quad 3^2]$
feature box	$40\cdot \mathbf{U}([x_f],[y_f])$	meas. cov.	$[2.0^2 \quad 0; \; 0 \quad 0.12^2]$
box num.	5	particle num.	50
buffer	8	buffer	None
discarding	8	discarding	None
RMSE pose (m)	1.21	RMSE pose (m)	3.09
RMSE θ (rad)	1.15	RMSE θ (rad)	0.85
RMSE feature (m)	1.82	RMSE feature (m)	3.18

[a] U(·) means intervals bounds subject to uniform distributions.

TABLE II
PARAMETERS AND PERFORMANCE OF THE TWO ALGORITHMS

BPF-CP	values	PF	values
state box	$50\cdot \mathbf{U}([x_r],[y_r],[\theta_r])^b$	control cov.	$[0.05^2 \; 0; \; 0 \quad 0.5^2]$
feature box	$40\cdot \mathbf{U}([x_f],[y_f])$	meas. cov.	$[0.1^2 \quad 0; \; 0 \quad 1.0^2]$
box num.	5	particle num.	50
buffer	8	buffer	None
discarding	8	discarding	None
RMSE pose (m)	0.63	RMSE-pose (m) pose (m)	0.75
RMSE θ (rad)	0.02	RMSE θ (rad)	1.05
RMSE feature (m)	0.62	RMSE feature (m)	0.76

[b] U(·) means interval bounds subject to uniform distributions.

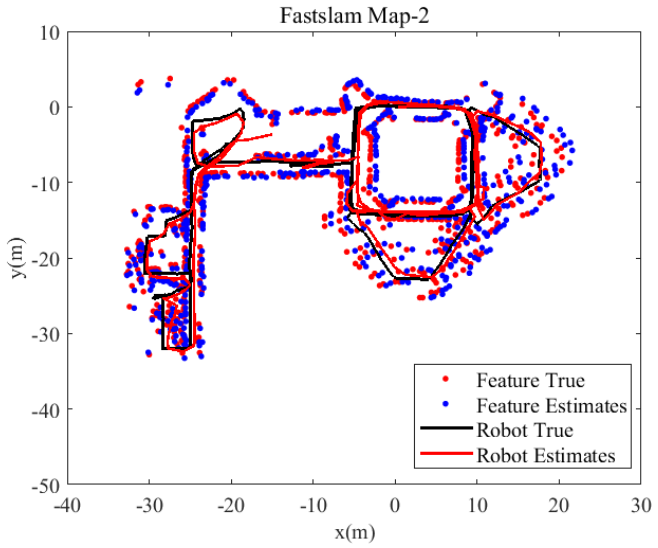
size reduction before and after the buffer contraction which highlights the interest of this stage.

We can conclude from this analysis that with less particles than a PF, BPF-CP provides good results in terms of accuracy and consistency, without any requirement on the prior distribution.

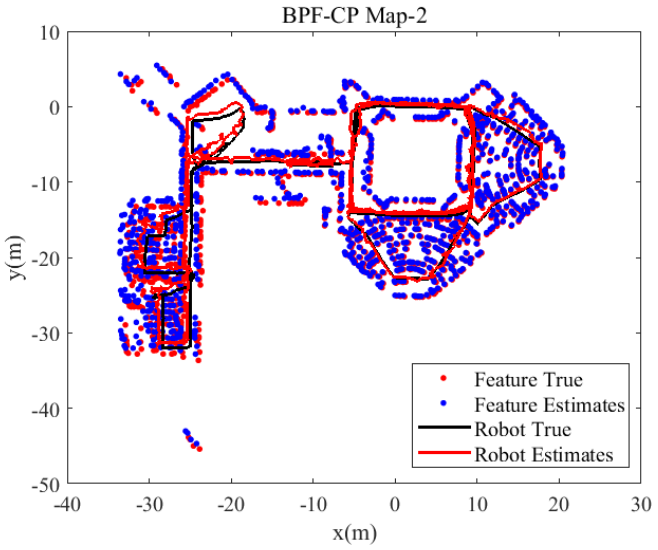
3) Computation time analysis: The q-satisfied robust method (to cope with empty boxes) and CP on the buffers (to decrease the box sizes, refer to Fig. 4) have an impact in terms of computation. Table III shows the computation time in seconds of BPF-CP, BPF and PF, respectively. We have repeated five times each simulation and experiment and then report the average time. With our sotware implementation, we can see that BPF alone is more efficient than PF with the same number of particles. BPF-CP with buffers has a computation time in the same order of magnitude as a PF with 200 particles. This indicates that the BPF-CP is not too heavy in terms of computation and the efficiency of this new approach is quite competitive.

TABLE III
COMPUTATION TIME COMPARISON

Setting	BPF-CP	BPF	PF	
			50 particles	200 particles
Simulation (s)	282.64	26.62	86.24	300.48
Experiment (s)	1947.18	839.77	1896.40	2024.94

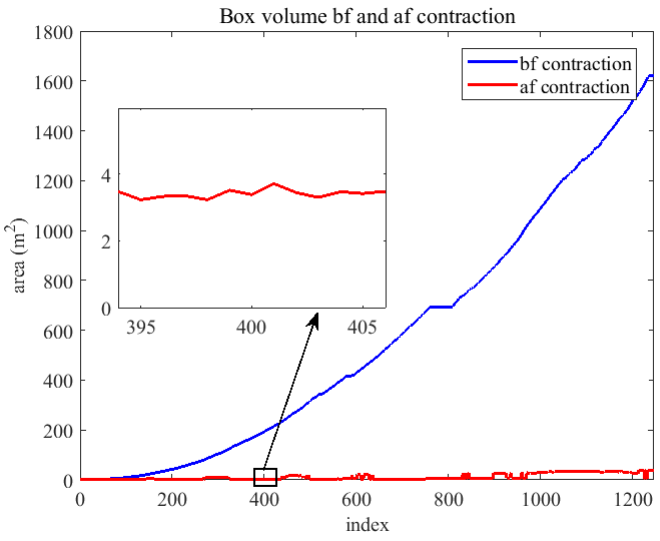


(a)

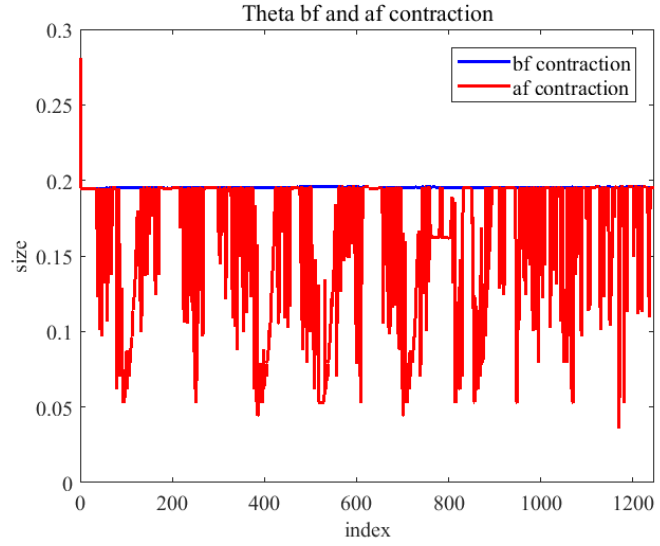


(b)

Fig. 3. Experimental SLAM results: (a) Results by fastslam; (b) Results by BPF-CP



(a)



(b)

Fig. 4. Robot state box size comparisons before and after contraction: (a) Position box sizes; (b) Theta size

VI. CONCLUSION

In this paper, we have shown how to use BPF to solve SLAM problems. In addition, we have also extend this approach with buffers, which helps a lot to reduce the uncertainty. A strong advantage of this approach is that it is robust to high level uncertainties thanks to the q-satisfied strategy that has been added in this framework. We have also demonstrated thanks to simulation and real experiments that BPF-CP can lead to the same performance as PF with a computational time that is equivalent.

A perspective is to design more efficient buffer policies and contraction strategies to improve the efficiency of the algorithm to make it suitable for online SLAM processes, with possible solutions lie in parallel computing and buffer sharing.

REFERENCES

[1] C. Cadena, L. Carlone, H. Carrill, Y. Latif, D. Scaramuzza, J. Neira, I. Reid and J.J. Leonard, Past, present, and future of simultaneous localization and mapping: Toward the robust-perception age, *IEEE Transactions on Robotics*, vol. 32, 2016, pp 1309-1332.

[2] H. Durrant-Whyte and T. Bailey, Simultaneous localization and mapping: part_I, *IEEE robotics & automation magazine*, vol. 13, 2006, pp 99-110.

[3] S. Thrun, M. Montemerlo, D. Koller, B. Wegbreit, J. Nieto, and E. Nebot, FastSLAM: An efficient solution to the simultaneous localization and mapping problem with unknown data association, *Journal of Machine Learning Research*, vol. 4, 2004, pp 380-407.

[4] Y. Latif, C. Cadena, and J. Neira, Robust loop closing over time for pose graph SLAM, *The International Journal of Robotics Research*, vol. 32, 2013, pp 1611-1626.

[5] S. L. Bowman, N. Atanasov, K. Daniilidis, and G. J. Pappas, Probabilistic data association for semantic SLAM, *in Proc. of the 2017 IEEE International Conf. on Robotics and Automation*, Singapore, 2017, pp 1722-1729.

[6] M. Di Marco, A. Garulli, S. Lacroix, and A. Vicino, Set membership localization and mapping for autonomous navigation, *International Journal of robust and nonlinear control*, vol. 11, 2001, pp 709-734.

[7] M. Di Marco, A. Garulli, A. Giannitrapani, and A. Vicino, A set theoretic approach to dynamic robot localization and mapping, *Autonomous robots*, vol. 16, 2004, pp 23-47.

[8] J. M. Porta, CuikSlam: A kinematics-based approach to SLAM, *in Proc. of the 2005 IEEE International Conf. on Robotics and Automation*, Barcelona, 2005, pp 2425-2431.

[9] L. Jaulin, A nonlinear set membership approach for the localization and map building of underwater robots, *IEEE Trans. on Robotics*, vol. 25, 2009, pp 88-98.

[10] B. Vincke, A. Lambert, and A. Elouardi, Guaranteed simultaneous localization and mapping algorithm using interval analysis, *in Proc. of 13th International Conf. on Control Automation Robotics & Vision*, Singapore, 2014, pp 1409-1414.

[11] F. Abdallah, A. Gning, and P. Bonnifait, Box particle filtering for nonlinear state estimation using interval analysis, *Automatica*, vol. 44, 2008, pp 807-815.

[12] A. Gning, B. Ristic, and L. Mihaylova, Bernoulli particle/box-particle filters for detection and tracking in the presence of triple measurement uncertainty, *IEEE Trans. on Signal Processing*, vol. 60, 2012, pp 2138-2151.

[13] A. De Freitas, L. Mihaylova, A. Gning, D. Angelova, and V. Kadirkamanathan, Autonomous crowds tracking with box particle filtering and convolution particle filtering, *Automatica*, vol. 69, 2016, pp 380-394.

[14] V. Drevelle and P. Bonnifait, Localization confidence domains via set inversion on short-term trajectory, *IEEE Trans. on Robotics*, vol. 29, 2013, pp. 1244-1256.

[15] N. Merlinge, K. Dahia, and H. Piet-Lahanier, A Box Regularized Particle Filter for terrain navigation with highly non-linear measurements, *IFAC-PapersOnLine*, vol. 49, 2016, pp. 361-366.

[16] K. Murphy, Bayesian map learning in dynamic environments, *in Proc. of 13th International Conf. on Neural Information Processing Systems*, Denver, 2000, pp 10151021.

[17] P. Wang, Q.B. Zhang, and Z.H. Chen, A grey probability measure set based mobile robot position estimation algorithm, *International Journal of Control, Automation and Systems*, vol. 13, 2015, pp 978-985.

[18] https://openslam.org/

[19] http://www-personal.acfr.usyd.edu.au/tbailey/software/slam_simulations/

[20] http://www.ensta-bretagne.fr/desrochers/pyibex/