Particle Filter for Unordered Observations

Longin Jan Latecki, Nagesh Adluru, and Xingwei Yang

Abstract—Usually in the Particle Filter (PF) framework it is assumed that the observations arrive sequentially, e.g., the observations are naturally ordered by their time stamps in the tracking scenario. Based on this assumption, the posterior density over the corresponding hidden states is estimated.
In this paper we relax the assumption of having ordered observations and extend the PF framework to simultaneously estimate the posterior density while exploring different orders of observations. This significantly broadens the scope of applications of PF methods. We demonstrate that the proposed framework can be applied to solve a broad spectrum of challenging problems, e.g., building jigsaw puzzles, detecting and localizing target objects in images, and inferring a globally consistent map given a set of partial maps in the task of multi robot map merging. The provided experimental results demonstrate that the proposed framework yields comparable results to state-of-the-art algorithms for object detection and localization in images.

Index Terms—Particle Filter, Object Detection and Recognition, Contour Grouping.

1 Introduction

We illustrate the key ideas of the proposed approach with an example of a jigsaw puzzle, where the goal is to estimate the poses (location and the orientation) of all puzzle pieces so that they fit according to their shape constraints and form a meaningful image. When a puzzle is created, one usually starts with an image on a paper or wooden board, e.g., see Fig. 1(a). Then the board is cut into jigsaw puzzle pieces. In our scenario, the player is only presented the jigsaw puzzle pieces in a random arrangement (e.g., see Fig. 1(b)) and is not shown the original image. Thus, the player has never seen the original image. To be more precise, the player is given m puzzle pieces described by a set of m observations $Z = \{z_1, \ldots, z_m\}$. Each observation z_t describes piece t and is given by a vector of features that represents the shape and the partial image depicted on piece t. The state of piece t, which describes its is pose, is represented by a random variable (RV) X_t. Each value x_t of X_t represents 2D coordinates and the orientation of piece t. Our goal is to determine the state vector $(x_1, \ldots, x_m)$ that solves the given puzzle, i.e., the relative poses of the puzzle pieces that yield the original image. In the statistical framework such a vector of poses maximizes the posterior distribution $p(X_1, \ldots, X_m|Z)$. In other words, $(x_1, \ldots, x_m)$ is a vector of most likely poses of the puzzle pieces given the observed features Z. The posterior distribution is a function of how well adjacent pieces fit together with respect to their shape and their images, which can be measured by similarity of their observed and derived features. Of course, the goal is to build the original image, which however in unknown to the player. Since the whole original image is not observed, Z is a set of partial observations that describe local features of each puzzle piece. Therefore, the posterior distribution, which we want to infer, is a function the set of partial observations Z, i.e., we use the observations Z to evaluate the likelihood of state vectors $(x_1, \ldots, x_m)$. Then, usually, the state vector that maximizes the posterior distribution is selected as the solution. We observe that the initial poses are irrelevant, since the player is initially given a random configuration of the puzzle pieces. We also observe that the posterior distribution $p(X_1, \ldots, X_m|Z)$ usually is very complicated and has many local maxima. This is particulary the case when the local information of the puzzle pieces is not very descriptive. Therefore, it is often important to estimate the whole posterior distribution.

It is possible to apply the classical particle filter (PF) framework as stochastic optimization to solve this problem by utilizing a fix order of observations $Z = \{z_1, \ldots, z_m\}$, e.g., by following a given numbering of the puzzle pieces. However, by doing so, we would have selected an arbitrary order, and the puzzle construction may fail because of the selected order. Would we have selected a different order, the localization task could have been successful. Moreover, the observations $Z = \{z_1, \ldots, z_m\}$ are given simultaneously at the same time. Consequently, there is no reason to favor any particular order without utilizing this fact.

The classical particle filter (PF) framework has been developed for sequential state estimation like tracking or robot localization. There, the observations arrive sequentially and are indexed by their time stamps. In the proposed framework we extend PF to handle the situations where we have unordered set of observations that are given simultaneously like in our example of jigsaw puzzle. One of our key observations is the fact that it is possible to extend the importance sampling from the proposal distribution so that the particle resampling automatically determines most informative orders of observations allowing us to simultaneously consider different orders of observation in the process estimating the posterior distribution. If our goal is maximizing the posterior distribution, we can view the order of observations that led to the global maximum as a most informative. Of course, this order does not have to

- *L. J. Latecki and X. Yang are with the Dept. of Computer and Information Sciences, Temple University, Philadelphia.*
  E-mails: latecki@temple.edu, xingwei.yang@temple.edu
- *Nagesh Adluru is with ?.*

Fig. 1. The goal is to build the original image (a) given the jigsaw puzzle pieces (b). The original image is not known, thus, it needs to be estimated given the observations shown in (b). Even if the puzzle pieces were numbered (c), following their numbering may not be helpful in solving the puzzle.

be unique, since it is possible that other orders of observations lead to the same global maximum.

2 Basic Facts about Particle Filters

In this section we summarize basic facts about Particle Filters (PFs). They will be utilized in the next section when we introduce the proposed framework.

Our goal is to compute a posterior distribution $p(X_1, \ldots, X_m \mid Z)$, where $(X_1, \ldots, X_m)$ is a vector of random variables (RVs) and $Z = \{z_1, \ldots, z_m\}$ is a set of observations. This will allow us to find value assignments $X_t = \widehat{x}_t$ for $t = 1, \ldots, m$ to RVs that maximize this posterior:

$$\widehat{x}_{1:m} = \underset{x_{1:m}}{\operatorname{argmax}}\, p(X_1 = x_1, \ldots, X_m = x_m \mid Z), \quad (1)$$

where $x_{1:m} = (x_1, \ldots, x_m) \in \mathcal{X}^m$ is a state space vector. As it is commonly the case, we will abbreviate

$$p(X_1 = x_1, \ldots, X_m = x_m \mid Z) = p(x_{1:m} \mid Z).$$

We will achieve our goal by approximating the posterior distribution with a finite number of samples in the framework of Bayesian Importance Sampling (BIS). Since it is usually difficult to draw samples from the pdf $p(x_{1:m}|Z)$, we will draw samples $x^{(i)}_{1:m} \sim q(x_{1:m}|Z)$ for $i = 1, \ldots, N$ from a proposal pdf q, form which samples are easily generated. Then approximation to the density p is given by

$$p(x_{1:m}|Z) \approx \sum_{i=1}^{N} w^{(i)} \delta(x_{1:m} - x^{(i)}_{1:m}), \quad (2)$$

where δ is the Dirac delta function and

$$w^{(i)} = \frac{p(x^{(i)}_{1:m}|Z)}{q(x^{(i)}_{1:m}|Z)} \quad (3)$$

are normalized weights (so that they sum to one).

Since it is still hard to draw samples from q due to high dimensionality of $x_{1:m}$, Sequential Importance Sampling (SIS) is usually utilized. In the classical PF approaches, samples are generated recursively following the order of dimensions in the vector of RVs $X = (X_1, \ldots, X_t)$:

$$x^{(i)}_t \sim q_t(x|x_{1:t-1}, Z) = q_t(x|x_{1:t-1}, z_{1:t}) \quad (4)$$

for $t = 1, \ldots m$, and the particles are built sequentially $x^{(i)}_{1:t} = (x^{(i)}_{1:t-1}, x^{(i)}_t)$ for $i = 1, \ldots, N$. The subscript t in q_t indicates from which dimension of the state space the samples are generated, i.e., from which RV. We use this notation to stress that q_t is a pdf from which we sample. (For $t = 1$, $x_{1:t-1} = x_{1:0}$ denotes an empty vector, thus, we sample $x^{(i)}_1 \sim q_1(x|Z)$.) Since q factorizes as

$$q(x_{1:m}|Z) = q(x_1|Z) \prod_{t=2}^{m} q(x_t|x_{1:t-1}, Z), \quad (5)$$

we obtain that $x^{(i)}_{1:m} \sim q(x_{1:m}|Z)$. In over words, by sampling recursively $x^{(i)}_t$ from each dimension t according to (4) and appending the samples to a vector $x^{(i)}_{1:m}$ we obtain a sample from $q(x_{1:m}|Z)$

Since at a given iteration we have a partial sample vector $x^{(i)}_{1:t}$ for $t < m$, we also need an evaluation procedure of this partial sample vector. For this we observe that the weights can be recursively updated according to

$$w(x^{(i)}_{1:t}) = \frac{p(z_t|x^{(i)}_{1:t}, z_{1:t-1}) p(x^{(i)}_t|x^{(i)}_{1:t-1})}{q(x^{(i)}_t|x^{(i)}_{1:t-1}, z_{1:t})}\, w(x^{(i)}_{1:t-1}). \quad (6)$$

The key observation is that when $t = m$, the weight $w(x^{(i)}_{1:m})$ of particle (i) is equal to $w^{(i)}$ (defined in (3)). The derivation of this fact can be found in [1]. Now we are able to formulate the SIS theorem in our setting:

Theorem 1. By sampling particles according to (4) and weighting them according (6), we obtain a set of weighted samples form $p(x_{1:m}|Z)$ once $t = m$. Consequently, we can approximate $p(x_{1:m}|Z)$ with any precision if the number of particles N is sufficiently large. Thus, we can write

$$p(x_{1:m}|Z) \approx \sum_{i=1}^{N} w(x^{(i)}_{1:t}) \delta(x_{1:m} - x^{(i)}_{1:m}). \quad (7)$$

The equation (6) can be simplified by making a common assumption that $q(x^{(i)}_t|x^{(i)}_{1:t-1}, z_{1:t}) = p(x^{(i)}_t|x^{(i)}_{1:t-1})$, i.e., the proposal distribution depends only on the partial state vector and does not depends on the observations. This assumption significantly simplifies the recursive weight update formula

$$w(x^{(i)}_{1:t}) = w(x^{(i)}_{1:t-1}) p(z_t|x^{(i)}_{1:t}, z_{1:t-1}), \quad (8)$$

and, consequently, the samples are generated from

$$x^{(i)}_t \sim p_t(x|x^{(i)}_{1:t-1}), \quad (9)$$

i.e., the current observations $z_{1:t-1}$ have no direct influence on the samples. Analog to (4) we added the index t in (9) to indicate the dimension of the state space from which the

samples are generated. The influence of observations on the samples is introduced when particles are resampled according to their weights, which is a common step in PF algorithms, since particle weights depend on observations.

In order to motivate the proposed approach, we derive now the recursive weight update in (6) in a different way than is usually reported in the literature, e.g., in [1]:

$$w(x_{1:t}^{(i)}) = \frac{p(x_{1:t}^{(i)}|Z)}{q(x_{1:t}^{(i)}|Z)} = \frac{p(x_t^{(i)}|x_{1:t-1}^{(i)}, Z)\ p(x_{1:t-1}^{(i)}|Z)}{q(x_t^{(i)}|x_{1:t-1}^{(i)}, Z)\ q(x_{1:t-1}^{(i)}|Z)}$$

$$= \frac{p(x_t^{(i)}|x_{1:t-1}^{(i)}, Z)}{q(x_t^{(i)}|x_{1:t-1}^{(i)}, Z)}\ w(x_{1:t-1}^{(i)}) \tag{10}$$

$$= \frac{p(Z|x_{1:t}^{(i)})}{p(Z|x_{1:t-1}^{(i)})}\ \frac{p(x_t^{(i)}|x_{1:t-1}^{(i)})}{q(x_t^{(i)}|x_{1:t-1}^{(i)}, Z)}\ w(x_{1:t-1}^{(i)}) \tag{11}$$

Eq. (11) follows from (10) by Bayes rule interchanging $x_t^{(i)}$ and Z in $p(x_t^{(i)}|x_{1:t-1}^{(i)}, Z)$.

Since both formulas (6) and (11) are equal to $w(x_{1:t}^{(i)})$, by setting (6) = (11), we obtain

$$p(z_t|x_{1:t}^{(i)}, z_{1:t-1}) = \frac{p(Z|x_{1:t}^{(i)})}{p(Z|x_{1:t-1}^{(i)})} \tag{12}$$

Equation (12) provides an intuitive justification for the reclusive weight update in (8). The weight of a particle (i) increases, if adding the new state $x_t^{(i)}$ to the state vector $x_{1:t-1}^{(i)}$ increases the probability of the observations Z, i.e., if $p(Z|x_{1:t}^{(i)}) > p(Z|x_{1:t-1}^{(i)})$.

Now we outline a **standard PF algorithm**. For every time step $t = 1, \ldots, m$ and for every particle $i = 1, \ldots, N$ execute the following three steps:

1) **Importance sampling / proposal:** Sample followers of particle (i) according to (9) (a special case of (4)) and set $x_{1:t}^{(i)} = (x_{1:t-1}^{(i)}, x_t^{(i)})$.

2) **Importance weighting / evaluation:** An importance weight is assigned to each particle $x_{1:t}^{(i)}$ according to (8) (a special case of (6)).

3) **Resampling:** Sample with replacement N new particles form the current set of particles

$$\{x_{1:t}^{(i)}|i = 1, \ldots, N\}$$

according to their weights. We obtain a set of new particles $x_{1:t}^{(i)}$ for $i = 1, \ldots, N$, all with weights of $1/N$. This procedure is called Sampling Importance Resampling (SIR) [1]. It is an important part of any PF algorithm, since resampling prevents weight degeneration of particles.

A common assumption underlying the equations (4) and (6) is that there exist two known functions f and h such that

$$x_t = f(x_{1:t-1}) + u_t \tag{13}$$

$$z_t = h(x_t) + v_t \tag{14}$$

where both u_t and v_t are mutually independent and identically distributed sequences with known probability density functions (pdfs). Often they are assumed to be Gaussians that model state prediction noise u_t and observation noise v_t. (Under the Markov assumption, (13) is simplified to $x_t = f(x_{t-1}) + u_t$, i.e., that we can determine x_t if we know the previous state x_{t-1}.)

3 Unordered Observations

In the proposed approach, the order of the observations is not predetermined, in particular, we do not follow the order of indices of the observations in Z. Our key idea is to extend the PF framework to examine all possible orders of observations and to follow the most informative orders. This way we are able to utilize the the most informative observations first. Intuitively, it makes sense, for example, if the first puzzle piece has shape very similar to many other puzzle pieces and the second puzzle pieces has a very distinctive shape, then our approach will first process the second puzzle piece, since it is more informative.

We stress that the standard SIS in Eq. 4 and particle evaluation in Eq. 6 utilize the sequential order of the RVs reflected in the order of dimensions in the state space $(x_1, \ldots, x_m)$. In many applications, this order is determined naturally by the time stamp of the observations, e.g., a single robot is collecting laser measurements at consecutive time points, in which case x_t denotes the robot pose at time t. The goal of our work is to extend SIS to applications in which there is no natural order of observations like the case of multi robot map merging.

The key idea of the proposed approach is not to utilize the fix order of the dimensions, but instead explore all possible orders of the dimensions $(x_{i_1}, \ldots, x_{i_m})$ (or equivalently RVs) so that the corresponding sequence of observations $Z = (z_{i_1}, \ldots, z_{i_m})$ is most informative. To achieve this we modify the first step of the PF algorithm so that the importance sampling is performed for every dimension not yet represented by the current particle. Thus, we sample follower states x_{i_t} of particle $(x_{i_1}^{(i)}, \ldots, x_{i_{t-1}}^{(i)})$ from each dimension that is not represented in $(x_{i_1}^{(i)}, \ldots, x_{i_{t-1}}^{(i)})$. For example, our puzzle in Fig. 1(c) has the total of six puzzle pieces and the current state vector of particle (i) is $(x_{i_1}^{(i)}, x_{i_2}^{(i)}) = (x_5^{(i)}, x_3^{(i)})$, where $i_1 = 5, i_2 = 3$, meaning that we have already placed pieces 5 and 3, Then in the next step we sample the follower states from dimensions representing the poses of the remaining puzzle pieces, which are pieces 1, 2, 4, 6 in our example.

To simplify the notation we replace the double indexing of the state variables $x_{i_s}^{(i)}$ with a bijection (onto and one-to-one function) $< \cdot >: \{1, \ldots, m\} \rightarrow \{1, \ldots, m\}$ and use the shorthand notation $< 1 : t >$ to denote $(< 1 >, < 2 >, \ldots, < t >)$. For example, if $(i_1, i_2, i_3) = (2, 3, 1)$, then

$$< 1 : 3 >= (< 1 >, < 2 >, < 3 >) = (i_1, i_2, i_3) = (2, 3, 1)$$

and we denote $x_{<1:3>}^{(i)} = x_{i_{1:3}}^{(i)} = (x_{i_1}^{(i)}, x_{i_2}^{(i)}, x_{i_3}^{(i)})$.

In order to introduce the extended sampling method, we need to revise the underlying assumption (13) to the existence of a function

$$x_s = f(x_{<1:t-1>}) + u_{s,<1:t-1>} \quad \text{for} \quad s \in \mathcal{N}(x_{<1:t-1>}), \tag{15}$$

where $\mathcal{N}(x_{<1:t-1>})$ denotes the neighborhood of the set of states $x_{<1:t-1>}$. The neighborhood is application dependent

with a simplest example being $\mathcal{N}(x_{1:t-1}) = \{t\}$, in which case we obtain (13) as a special case of 15.

In our jigsaw puzzle example, the neighborhood of the pose states of the two puzzle pieces 3 and 5 can be composed of all the remaining pieces, e.g., $\mathcal{N}(x_{<1,2>}) = \mathcal{N}(x_{3,5}) = \{1, 2, 4, 5\}$. Then function f allows to predict the poses of the puzzle pieces $s \in \mathcal{N}(x_{<1,2>})$. Consequently, (15) means that we can estimate the relative pose of the remaing puzzle pieces for a given configuration of already positioned pieces.

We stress that now the sequence of states $x_{<1:t-1>}$ visited before time t is not a sequence of consecutive numbers $(1, \ldots, t-1)$ but any subsequence $(i_1, \ldots, i_{t-1})$ formed by $t-1$ different numbers in $\{1, \ldots, m\}$. Due to noise factor $u_{s,<1:t-1>}$ in (13), each value x_s is an estimate or prediction of a true unknown value.

4 Particle Filter for Unordered Observations

We are ready now to precise formulate the proposed importance sampling. At iteration t, the assumption (15) allows us to sample

$$x_s^{(i)} \sim p_s(x|x_{<1:t-1>}^{(i)}) \quad \text{for each} \quad s \in \mathcal{N}(<1:t-1>), \tag{16}$$

where the index s at the posterior pdf p_s indicates that we sample from RV in dimension s. Thus, we generate at least one sample for each dimension $s \in \mathcal{N}(<1:t-1>)$.

For example, if the neighborhood of $<1:t-1>$ is composed of all dimensions not present in $<1:t-1>$, i.e.,

$$\mathcal{N}(<1:t-1>) = \{1, \ldots, m\} \setminus <1:t-1> \tag{17}$$

and we generate exactly one sample from each dimension (i.e., RV) $s \in \mathcal{N}(<1:t-1>)$, then at iteration t particle (i) has $m-t+1$ followers. Each follower is a sample from a different dimension of the state vector (i.e., a different RV).

Let us assume that we have total of $m = 5$ pieces in our jigsaw puzzle and after iteration 3, we determined the poses of puzzle pieces $<1:3> = (3, 1, 5)$. We then consider the two remaining pieces as followers of $<1:3> = (3, 1, 5)$ at iteration 4. The poses of pieces 2 and 4 are determined with reference to the poses of pieces $3, 1, 5$, i.e., by sampling $x_2^{(i)} \sim p_2(x|x_{<1:3>}^{(i)})$ and $x_4^{(i)} \sim p_4(x|x_{<1:3>}^{(i)})$. Of course, this process is repeated for each particle (i) for $i = 1, \ldots, N$, where N is the number of particles.

In contrast, in the standard application of rule (9), at each iteration t particle (i) has one follower. Even when sometimes each particle (i) has many followers, all followers are in the same dimension (samples form the same RV), which means that we only determine possible locations of say puzzle piece 2 and do not consider locations of puzzle piece 4 for particle (i), since a strict order of the state dimensions is followed in the classical setting.

We do not make any Markov assumption in (16), i.e., the new state $x_s^{(i)}$ is dependent on all previous states $x_{<1:t-1>}^{(i)}$ for each particle (i).

We derive now the recursive weight update formula for the unordered set of static observations Z. This derivation is analog to our derivation in (11). For every t from 2 to m, we have

$$\begin{aligned} w(x_{<1:t>}^{(i)}) &= \frac{p(x_{<1:t>}^{(i)} \mid Z)}{q(x_{<1:t>}^{(i)} \mid Z)} \\ &= \frac{p(x_{<t>}^{(i)}|x_{<1:t-1>}^{(i)}, Z)\ p(x_{<1:t-1>}^{(i)}|Z)}{q(x_{<t>}^{(i)}|x_{<1:t-1>}^{(i)}, Z)\ q(x_{<1:t-1>}^{(i)}|Z)} \\ &= \frac{p(x_{<t>}^{(i)}|x_{<1:t-1>}^{(i)}, Z)}{q(x_{<t>}^{(i)}|x_{<1:t-1>}^{(i)}, Z)}\ w(x_{<1:t-1>}^{(i)}) \\ &= \frac{p(Z|x_{<1:t>}^{(i)}) p(x_{<t>}^{(i)}|x_{<1:t-1>}^{(i)})}{p(Z|x_{<1:t-1>}^{(i)}) q(x_{<t>}^{(i)}|x_{<1:t-1>}^{(i)}, Z)}\ w(x_{<1:t-1>}^{(i)}) \end{aligned} \tag{18}$$

The only difference to (11) is that we replace the fix order of states $1:t-1$ with a dynamic order $<1:t-1>$ that may be different for each particle (i). (In particular, to obtain the last equation, we apply Bayes rule to decompose $p(x_{<t>}^{(i)}|x_{<1:t-1>}^{(i)}, Z)$ that interchanges $x_{<t>}^{(i)}$ and Z.)

As it is often the case in PF applications, we assume that $q(x_{<t>}^{(i)}|x_{<1:t-1>}^{(i)}, Z) = p(x_{<t>}^{(i)}|x_{<1:t-1>}^{(i)})$. Using this simple exploration based proposal the weight recursion in (18) becomes:

$$w(x_{<1:t>}^{(i)}) = \quad = w(x_{<1:t-1>}^{(i)})\ \frac{p(Z|x_{<1:t>}^{(i)})}{p(Z|x_{<1:t-1>}^{(i)})} \tag{19}$$

Our next derivation step differs fundamentally from the standard PF framework. By recursive substitution of weights in (19), i.e., by applying (19) to $w(x_{<1:t-1>}^{(i)}), w(x_{<1:t-2>}^{(i)}), \ldots, w(x_{<1:2>}^{(i)})$, we obtain

$$\begin{aligned} w(x_{<1:t>}^{(i)}) &= w(x_{<1:t-2>}^{(i)})\ \frac{\cancel{p(Z|x_{<1:t-1>}^{(i)})}}{p(Z|x_{<1:t-2>}^{(i)})}\ \frac{p(Z|x_{<1:t>}^{(i)})}{\cancel{p(Z|x_{<1:t-1>}^{(i)})}} \\ &= \cdots = w(x_{<1>}^{(i)})\ \frac{p(Z|x_{<1:t>}^{(i)})}{p(Z|x_{<1>}^{(i)})} \end{aligned} \tag{20}$$

Under the assumption that all particles have the same initial weight $w(x_{<1>}^{(i)})$ and the same initial observation probability $p(Z|x_{<1>}^{(i)})$ for $i = 1, \ldots, N$, we obtain

$$w(x_{<1:t>}^{(i)}) = p(Z|x_{<1:t>}^{(i)}) \tag{21}$$

Since for $t = m$ we have $w(x_{<1:m>}^{(i)}) = w(x_{1:m}^{(i)})$, we obtain that the weights computed by the recursive formulas (19) - (21) are equal to the weights in Eq. (3).

For comparison, the corresponding weight update in the standard PF framework [1] is

$$w(x_{1:t}^{(i)}) = w(x_{1:t-1}^{(i)})\ p(z_t|x_{1:t-1}^{(i)}, x_t), \tag{22}$$

where z_t denotes the new observations obtained at time t.

A distinctive feature of our derivations is the fact that we use the whole set of observations Z. In particular, this means that we do not decompose Z into partial sequences following some order of observations. Consequently, our weight update formula (21) evaluates how likely the whole set of observations Z is conditioned on the current state vector $x_{<1:t>}^{(i)}$.

We outline now the proposed **PF for unordered observations (PFUO)** algorithm. For every time step $t = 1, \ldots, m$

and for every particle $i = 1, \ldots, N$ execute the following three steps:

1) **Importance sampling / proposal:** Sample followers $x_s^{(i)}$ of particle (i) from each dimension $s \in \mathcal{N}(< 1 : t-1 >)$ according to (16), which we repeat here for completeness,

$$x_s^{(i)} \sim p_s(x|x_{<1:t-1>}^{(i)}) \tag{23}$$

and set $x_{<1:t-1>,s}^{(i)} = (x_{<1:t-1>}^{(i)}, x_s^{(i)})$. We need the second index s, since in our framework, particle $x_{<1:t-1>}^{(i)}$ usually has more than one follower and s indexes the followers of $x_{<1:t-1>}^{(i)}$. This means that the single particle $x_{<1:t-1>}^{(i)}$ is multiplied and extended to several follower particles $x_{<1:t-1>,s}^{(i)}$. If we generate one sample for each dimension $s \in \mathcal{N}(< 1 : t-1 >)$, then we the number of followers $x_{<1:t-1>,s}^{(i)}$ is equal to the number of dimensions in $\mathcal{N}(< 1 : t-1 >)$.

2) **Importance weighting/evaluation:** An individual importance weight is assigned to each particle follower $x_{<1:t-1>,s}^{(i)}$ according to Eq. 21, where we substitute $x_{<1:t-1>,s}^{(i)}$ for $x_{<1:t>}^{(i)}$, i.e.,

$$w(x_{<1:t-1>,s}^{(i)}) = p(Z|x_{<1:t-1>,s}^{(i)}) \tag{24}$$

3) **Resampling:** Sample with replacement N new particles form the current set of particles

$$\{x_{<1:t-1>,s}^{(i)} |\ i = 1, \ldots, N,\ s \in \mathcal{N}(< 1 : t-1 >^{(i)})\}. \tag{25}$$

according to the weights. We obtain a set of new particles $x_{<1:t>}^{(i)}$ for $i = 1, \ldots, N$, all with weights of $1/N$. This is the standard Sampling Importance Resampling (SIR) step [1] as in the classical PF framework, but the set of particles that is resampled is different. We recall that the index of dimensions $< 1 : t-1 >$ may be different for each particle (i), which is why we use $< 1 : t-1 >^{(i)}$ in (25).

Theorem 2. The PFUO computes an approximation of Eq. 2, i.e., for $t = m$ the particles $x_{<1:t>}^{(i)}$ with $i = 1, \ldots, N$ provide an approximation to the posterior $p(x_{1:m}|Z)$ for sufficiently large N, i.e.,

$$p(x_{1:m}|Z) \approx \sum_{i=1}^{N} w(x_{1:m}^{(i)}) \delta(x_{1:m} - x_{1:m}^{(i)}). \tag{26}$$

Before we prove this theorem, we need to make an important remark. Since $t = m$, all dimensions of considered RVs are present in each particle $x_{<1:t>}^{(i)}$. Therefore, we can simplify the notation and write $x_{<1:t>}^{(i)} = x_{1:m}^{(i)}$. We stress again that the particular order of dimensions $< 1 : t >$ is extremely important in our approach in order to derive the state vector $x_{<1:t>}^{(i)}$ of particle (i). However, once we have this state vector, we can simply present the state vector in the original order of dimensions.

Proof of Theorem 2. Our proof is based on Eq. (26) with weights defined in Eq. (3). We observe that the weights computed by the recursive formulas (19) - (21) are equal to the weights in Eq. (3), since for $t = m$ we have $w(x_{<1:m>}^{(i)}) = w(x_{1:m}^{(i)})$.

The Sampling Importance Resampling (SIR) replaces weighted particles with N particles with the weight equal to $1/N$, which provides an approximation to the same target pdf. This proves the theorem.

The fact that we can consider more than one follower of each particle and reduce the number of followers by resampling is known in the PF literature and is referred to as prior boosting [2], [3]. It is used to capture multi-modal likelihood regions. We stress that the resampling in our framework plays an additional and a very crucial role. It selects the the most informative random variables (i.e., state space dimensions) as followers of particles. Since the weights of $x_{<1:t-1>,s}^{(i)}$ are determined by the observations Z, and the resampling uses the weights to selects new particles $x_{<1:t>}^{(i)}$ by adding not yet considered dimensions, the resampling determines the order of RVs, i.e., the bijection $< t >$ for $t = 1, \ldots m$. Consequently, the order of RVs is heavily determined by observations Z, and this order may be different for each particle (i). This is in strong contrast to the classical PF, where observations Z have no influence on the order of RVs, which is fixed.

In many applications, the weight (21) of each particle is based on the evaluation of the similarity of the predicted observations $h(x_{<1:t-1>,s}^{(i)})$, i.e., the observations derived from the current state vector $x_{<1:t-1>,s}^{(i)}$, to the real observations in set Z. If we adopt the standard PF assumption (14) that underlies particle weight computation, we can only relate the predicted observations $h(x_{<1:t-1>,s}^{(i)})$ to the corresponding real observations $z_{<1:t-1>,s}$. Then we utilize a special case of (24) to compute the particle weights

$$w(x_{<1:t-1>,s}^{(i)}) = p(z_{<1:t-1>,s}|x_{<1:t-1>,s}^{(i)}). \tag{27}$$

This means that we only use the subset of observations that corresponds to the current dimensions of the state vector to evaluate particle (i). For instance, if the current state vector of particle (i) is $x_{<1>,s}^{(i)} = x_{5,3}^{(i)}$, meaning that we have positioned the puzzle piece $< 1 >= 5$ and the piece $s = 3$ as shown in our puzzle example in Fig. 1(c), then the weight of particle (i) depends on how likely the corresponding observations $z_{<1>,s} = z_{5,3} = (z_5, z_3)$ are conditioned on the state vector $x_{<1>,s}^{(i)} = (x_5^{(i)}, x_3^{(i)})$. In our jigsaw puzzle example, this means that we evaluate the likelihood of the appearance features (z_5, z_3) of puzzle pieces 5 and 3 conditioned on their poses $(x_5^{(i)}, x_3^{(i)})$. This can be computed by evaluating the fit of the puzzle pieces positioned according to the poses $(x_5^{(i)}, x_3^{(i)})$. Since the pieces 5 and 3 match well, the particle shown in Fig. 1(c) would obtain a high weight.

However, in our framework it is also possible to generalize the assumption (14) so that we always use the whole set of observations Z to compute the particle weights, i.e., we directly use formula (21). This may increase the discriminative power of the particle evaluation. For instance, if we use the appearance of all six puzzle pieces $Z = \{z_1, z_2, z_3, z_4, z_5, z_6\}$ to compute the weight of particle $x_{<1>,s}^{(i)} = (x_5^{(i)}, x_3^{(i)})$, we can additionally consider how distinctive the appearance features (z_5, z_3) are in the context of all feature vectors in Z. In this case, the value of the weight depends on two factors,

- how descriptive a given vector of observations $z_{<1:t-1>,s}$ is in the context of all observations Z and
- how good the prediction $h(x^{(i)}_{<1:t-1>,s})$ of observations $z_{<1:t-1>,s}$ is.

5 Looping Particle Filter for Unordered Observations

In the PF algorithm introduced in Section 5, the number of iterations t is equal to the dimensionality m of the state space. In our jigsaw puzzle example with six puzzle pieces, this means that $t = 6$. Consequently a large number of particles N is needed to well approximate the posterior distribution. According to Theorem 2, such a number of particles always exist, but it may be hard in practice to set it properly. Therefore, we propose in this section a modification of the importance sampling / proposal step that makes number of iterations independent from the dimensionality of the state space.

In order to obtain a second version of the proposed algorithm, called **Looping PF for unordered observations (L-PFUO)** algorithm, we only need to slightly modify the neighborhood in the importance sampling / proposal step in (16). For iteration steps $t \leq m$ we keep the user defined neighborhood structure, e.g., as defined in (17), but for $t > m$ we set the neighborhood structure to be

$$\mathcal{N}(< 1 : t-1 >) = \{1, \ldots, m\}. \quad (28)$$

We observe that for $t > m$ our L-PFUO becomes a variant of Gibbs sampler, since we loop over the state space and replace values in one dimension by sampling from the conditional proposal distribution (with values of the other dimensions being fixed). However, our PF steps (2) and (3) are very different from the existing Gibbs sampler approaches. While there exist Gibbs sampler approaches that weight samples [], none of them performs resampling as in our step (3).

In L-PFUO, the length of each state vector is equal to t if $t < m$ and is equal to m for every iteration $t \geq m$. Thus, L-PFUO can be viewed as a mixture of PF and Gibbs sampler approaches. For $t < m$ we follow the PF framework but with unordered observations and incrementally build the state vector until we reach its full length m. This makes L-PFUO significantly more efficient than Gibbs sampler, which must be initialized with the full length state vectors and therefore requires a lengthy burn in phase (i.e., the first N samples are discarded, where N can be very large). Once $t = m$, L-PFUO switches to looping over the state space. L-PFUO loops following the order determined by the resampling according to the particle weights, which allows us to generate more samples from the most informative dimensions of the state space. In contrast, most Gibbs samplers loop in a predefined deterministic order or in a completely random order [].

As we stated at the beginning of this section, L-PFUO algorithm allows us to estimate the posterior distribution with a significantly smaller number of particles than PFUO. This means that we trade the smaller number of particles N for the larger number or iterations t. While PFUO has a clearly defined stop condition $t = m$, we need to specify the stop condition for L-PFUO algorithm. We simply iterate L-PFUO until the posterior distribution stabilizes, i.e., the change in the posterior distribution is below a specified threshold. This simple stop condition indicates the convergence of L-PFUO algorithm to a stable solution. In contrast, there does not seem to exist any condition on the number of particles N that would indicate reaching a stable solution of PFUO.

6 Multi Robot Map Merging

In this section, we describe an illustrative application of the proposed framework to the problem of merging partial maps acquired by multiple robots. We use map merging as an illustrative instance of the jigsaw puzzle problem where the individual puzzle pieces are the local maps which have to be put together into a full global map. The goal becomes to estimate the poses of the individual maps such that all the individual maps align into a consistent global map.

6.1 System Description

We have a team of m robots that built m local maps, $Z = \{z_1, \ldots, z_m\}$. Each map is represented as a 2D occupancy grid with three types of cells representing free space, obstacles, and unknown space. Some maps have partial overlap, i.e., they have common parts, while some do not have any overlap. Let $X = \{x_1, \ldots, x_m\}$ be the poses of the local maps in a global reference frame that need to be estimated. Without getting into full details we explain the two most important components needed in importance weighting and proposal distribution viz. $p(Z|x^{(i)}_{<1:t-1>,s})$ and $p_s(x|x^{(i)}_{<1:t-1>})$. More details on the system can be found in [4]. An example result is shown in Fig. 2.

6.2 Importance weights

The importance weights depend on how well the partial maps fit together. We denote with $M^{(i)}_{<1:t-1>,s}$ a partially assembled map of particle (i) obtained by putting the local maps $z_{<1:t-1>,s}$ at poses $x^{(i)}_{<1:t-1>,s}$ in the global coordinate system. We define

$$p(Z|x^{(i)}_{<1:t-1>,s}) = \Psi(M_{<1:t-1>,s}) \cdot \gamma(M_{<1:t-1>,s}) \quad (29)$$

where Ψ is computed following the description in [5]. The basic idea is to treat the images as arrays of pixels (occupancy grid) and reward matching pixels based on their values and relative positions. The arrays are scanned for each class of pixels (obstacles, free space, and unknown). For each pixel, the distance to its closest similarly valued pixel in the other image is computed and its score is taken to be proportional to a Gaussian transform of this distance. Fortunately the distances can be computed in linear time in the number of pixels in the images using "distance-maps" [5].

γ is designed in such a way that its magnitude is made proportional to the merge inconsistency. The inconsistency of a merge is defined as the mismatch in the perception of the robot between two differently calculated positions. The mismatch in robot perception is based on the number of disagreeing pixels in the two images. Thus its computation involves similar steps

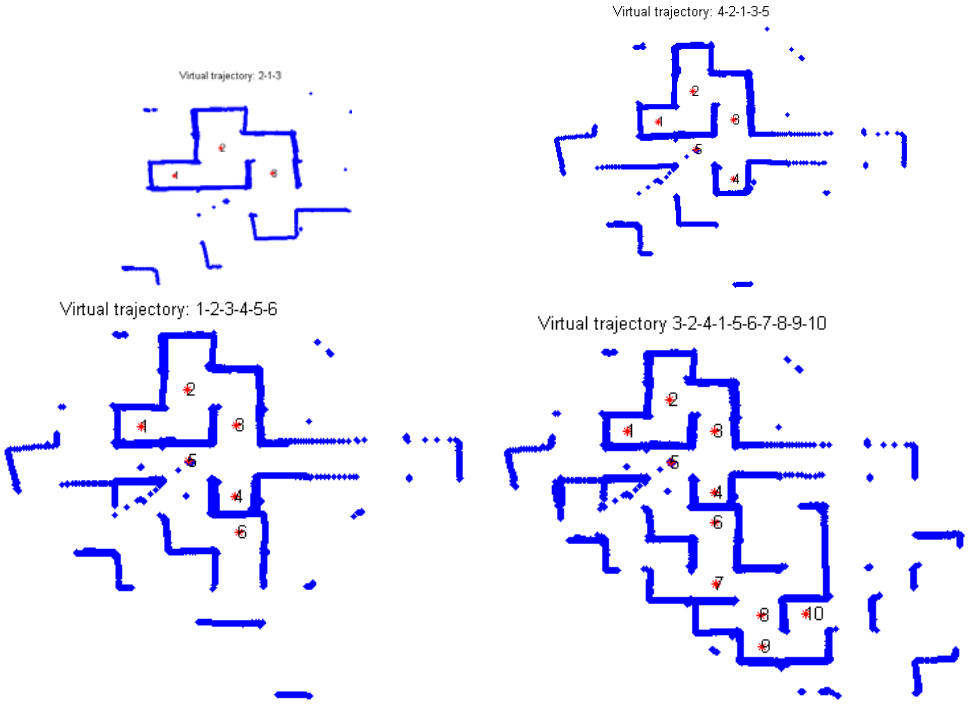


Fig. 2. Merged maps of $3, 5, 6$ and 10 robots. The map indices are placed at the local origins of the individual maps. The sequence of the maps merged is shown in the respective titles.

as those for Ψ. The larger the number of matching pixels ($c_1 = c_2$), the larger Ψ is. In contrast to this, the smaller the number of mismatched pixels ($c_1 \neq c_2$), the larger γ is.

¡Why do we need γ?¿

6.3 Proposal distribution

The discrete probability distribution function (pdf), $p_s(x|x^{(i)}_{<1:t-1>})$, is simulated using "structure registration" among local maps. The initial distribution is uniform and can pick any of the local maps, which becomes the reference coordinate (global) frame in which the puzzle is completed. The poses of rest of the puzzles are determined in reference to this frame. Pose of each frame can be imagined at the center of each of the puzzle.

Similar structures among local maps are extracted and registered using closed form solutions like in [6]–[8]. The correspondences are obtained by shape matching between similar structures. There could be several similar structures between local maps z_t and x_u. As a consequence, our proposal distribution is usually multi-modal with peaks around the poses predicted using structure registration. We note that each possible pose update is actually a transformation of the local map, z_s, into the global frame of $M_{<1:t-1>,s}$.

6.4 Experimental Results

We applied our technique to the data collected by National Institute of Standards and Technology (NIST) in a "maze" type of environment[1]. We are provided only with the local maps without any initial estimate of the relative poses. For feature extraction we fit lines to the data points and determine corners. For typical indoor local maps, corners are sufficiently distinct feature descriptors to represent the environment structure. On average there were about 10 corners in a local map and there were about 5 matching corner pairs between two local maps. Fig. 2 shows maps of best particles at iterations $3, 5, 6$ and 10, which corresponds to global maps merged from local maps of $3, 5, 6$ and 10 robots. As can be seen we could successfully merge maps of 10 robots. As virtual trajectory we list the state dimensions, which are the indices of the robots, used to build the global maps at corresponding iterations. We observe that at different iterations the order is different, which nicely illustrates the fact that the proposed framework explores different orders of particles.

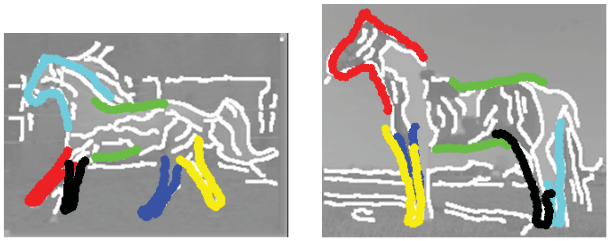
1. We would like to thank Raj Madhavan (NIST) for providing the dataset.

7 Object Detection and Recognition by Groping Contour Fragments

7.1 System Description

In this section we present an application of the proposed framework to object detection and recognition in images. Two detection examples are illustrated in Fig 3. The white curves show edge fragments obtained by edge detection. The contour segments that form detected horses in these images are marked with colors, where the colors indicate the order of detecting and localizing these segments. We observe that the two orders are different, which nicely illustrates the advantages of the proposed framework, e.g., the best particle for the image shown in (b) begins with the state dimension representing the position of the horse head, which makes sense intuitively, since

the head contour can be clearly identified in (b). The situation is different in (a), where the horse head is positioned in the cluttered background. The first dimension of the best particle in (a) represents the the state of a front leg.



(a) (b)

Fig. 3. Examples of two different inferred orders of detected contour parts. Colors represent the order, which is 1=red, 2=cyan, 3=blue, 4=green, 5=yellow, and 6=black.

Our detection is model driven, where a model is defined by the contour of an example object of a given class that can be hand drawn or extracted from an example image, e.g., see the apple logo in Fig. 4. Since edge fragments in real images may be broken and may have missing parts, we decompose our model contour into possibly overlapping model contour segments $S = \{s_1, \ldots, s_k\}$, which are grouped into part bundles. An example bundle decomposition is shown in Fig. 4. The main constraint for the bundle design is to ensure that a rough shape sketch obtained by selecting one part form each bundle still resembles the model contour. A cognitive motivation behind our bundle decomposition scheme is that an object can be recognized even if some parts of it are missing, as can be observed in Fig. 5. There are several reasons why parts of objects can be missing in real images: occlusion, missing or weak edge information, and failures of edge detection algorithms. The selection of parts and their grouping into bundles was designed manually. We have one part bundle model per shape class.

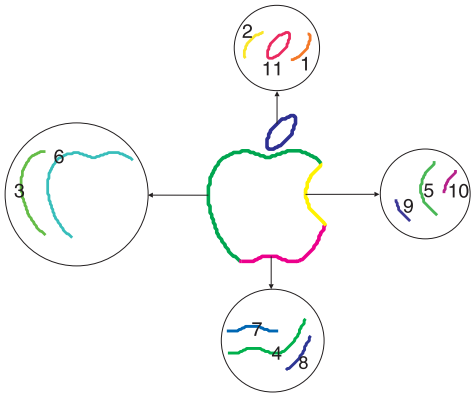


Fig. 4. The contour model of the apple and the corresponding part bundles. The contour is shown in the center. The 11 contour fragments are decomposed into four part-bundles.

Formally, $\mathcal{B} = \{B_k\}_{k=1}^{m}$, where $B_k \subset S$ and $m \leq k$, is a part bundle decomposition of S if and only if $\bigcup \mathcal{B} = S$ and $B_i \bigcap B_j = \emptyset$ for $i, j = 1, ..., m$ and $i \neq j$. For example, the part bundle decomposition of the apple logo in Fig. 4 is composed of 11 contour segments grouped into four bundles i.e.,$k = 11$ and $m = 4$. The four bundles are represented by four RVs. In general, if a model is composed of m part bundles, our goal is to estimate the states of m RVs $X_1, \ldots, X_m$. The values of each RV X_t range over a 4D parameter space $x_t = (s_t, h_t, v_t, \alpha_t)$, where s_t is an index of a selected segment from the part bundle B_t, (h_t, v_t) describe the horizontal and vertical coordinates of the centroid of this segment on the image, and α_t describes its orientation. (We assume that the scale is known, but it is also possible to estimate it in our framework).

Our shape model $\mathcal{M}$ of a given object class is composed of the part bundle $\mathcal{B}$ and a set of contours of positive instances of objects in this class. The instances represent shape constraints of the object class. We measure the similarity of deformed parts on the image (due to rotation α_t) to the set of the instance contours in order to ensure that the deformed model fragments still resemble the shape of object in this class.

Fig. 5. Parts of the objects are missing both due to missing edges and due to broken edge links.

The two phases of the proposed algorithm L-PFUO have a natural interpretation in this framework. Given a particle $x_{1:t-1}^{(i)}$, at steps $t = 2, \ldots, m$ we construct a rough shape sketch of the model shape in the edge image. Then for $t > m$ we loop over the RVs and refine the placement of the model segments on the image with possibly selecting different segments form the same bundle. Intuitively this means that the particles first trace a rough shape before adjusting the shape details. Of course, we can achieve the same effect with our first algorithm PFUO if we simply have a larger number of particles.

7.2 Importance Weights

The weight of each particle is computed based on the shape similarity of the contour constructed from the selected model segments and the edges in the image. We compute it as a Gaussian of the Oriented chamfer matching (OCD) [9]. However, we also need to consider the deformation of the model introduced by the rotation of the model segments on the image. We measure this deformation as similarity to known positive model instances. To be more precise, in this application we have a static set of observations $Z = (I, \mathcal{M})$, where I is a given edge image and $\mathcal{M}$ is a model of a given shape class, since both I and $\mathcal{M}$ are known and fixed. Since and I and $\mathcal{M}$ can be viewed as independent conditioned on $x_{<1:t-1>,s}^{(i)}$ we obtain:

$$p(Z|x_{<1:t-1>,s}^{(i)}) = p(I|x_{<1:t-1>,s}^{(i)}) p(\mathcal{M}|x_{<1:t-1>,s}^{(i)}). \quad (30)$$

The first factor defines the probability that the model parts positioned on the image according to the parameters of particle

$x^{(i)}_{<1:t-1>,s}$ led to the edges around them in the edge map I and we define it as

$$p(I|x^{(i)}_{<1:t-1>,a}) = \exp(-\beta \cdot OCD_I(x^{(i)}_{<1:t-1>,s})), \quad (31)$$

where OCD_I returns the Oriented Chamfer distance. Consequently, OCD_I measures how well the constructed partial model matches to the edge map I.

The second factor in (30) allows us to ensure that the rotated model parts on the image still resemble objects in the given shape class. It can be expressed as similarity of the model parts on the image (deformed by rotations α_t) to the set of the instance contours, which is also measured with OCD.

7.3 Proposal Distribution

In order to execute the derived PF algorithm, it remains to define the proposal distribution. The initial proposal distribution $p_s(x)$ is simply the probability of finding a model part from part bundle B_s at each location in the image I. It is obtained as a Gaussian of the OCM_I computed in the sliding window fashion at every location in I. In our application, the conditional proposal distribution $p_s(x|x^{(i)}_{<1:t-1>})$ is the probability of finding a model part from part bundle B_s around the locations of the previously positioned parts according to the parameters of particle $x^{(i)}_{<1:t-1>}$, i.e., the location of a model part from part bundle B_s is constrained by the already positioned model parts. It is also computed as a Gaussian of the OCM_I. While the initial proposal distribution is computed at every image location, the conditional proposal distribution is only computed at regions of interest determined by the previously placed model parts.

7.4 Experimental Results

This section is based on our experimental evaluation reported in [10]. Specific implementation details of our shape similarity and model construction can be found in [10].

We have tested our algorithm on three widely used data sets: the extended Weizmann Horses [11], [12], the ETHZ shapes [13] and the TU Darmstadt Database [14]. During the testing for Weizmann Horses, only 12 automatically selected horse silhouettes with one hand decomposed horse are used to learn the shape model. All the other images are used for testing. The edge maps for this dataset are obtained by Canny edge detector. We also test our method on the class of giraffe in ETHZ shape dataset [13]. The reason why we only select the category giraffes from ETHZ is that our model learning method can only transfer between objects with similar structure and giraffe is the only object in ETHZ having similar structure to horse. Only one hand decomposed horse and 6 automatically selected giraffe silhouettes are used to learn the giraffe model. Further, we work on the cow dataset the TU Darmstadt Database [14], since cows have similar structure with the above two classes. It contains 111 images. Only one hand decomposed horse and 6 automatically selected cow silhouettes are used to learn the cow model. The edge maps for this dataset are obtained by Canny edge detector.

To adapt to large scale variance, we generate multiple models by resizing the original ones to 5 to 8 scales, and choose as the final result from the best score in all the scales. We not only report our results on the commonly used bounding box intersection, but also the accuracy of our boundary localization.

We first evaluate the ability of the proposed approach to localize objects in cluttered images using bounding-box intersection, which is widely used in traditional object detection task. We adopt the strict standards of PASCAL Challenge criterion: a detection is counted as correct only if the intersection-over-union ratio with the ground-truth bounding-box is greater than 50%.

Fig. 6 reports precision-recall (P/R) curve and detection rate vs false positive per image (DR/FPPI) curve for the class Giraffes in ETHZ dataset [13]. In P/R, we compare to Lu et al. [15], Zhu et al. [16], Ommer and Malik [17] and Ferrari et al. [13], whose results are quoted from [15]. In DR/FPPI, as Ferrari et al. [13], [18], Ommer and Malik [17] and Lu et al. [15] provide their results, we compare to them. As Ravishankar et al. [19] do not give their curves, we do not compare to them in Fig. 6. According to the curves, we are better than Lu et al. [15], Ommer and Malik [17], Ferrari et al. [13], [18] and perform equally well as Zhu et al. [16]. The performance of the proposed method illustrates its ability to cope with substantial nonrigid deformations, which are present in the class Giraffes. This is demonstrated by our example results in Fig. 7(a).

Table 1 compares our detection rate to [12], [20] on Weizman Hores and TU Darmstadt Cows. The detection rate on horses is estimated from the DR/FPPI curve in [12]. The DR/FPPI curve for cows is not available in [12]. The method in [20] is also matching based, while [12] is a classification method. Some examples of our horse and cow detection results are shown in Fig. 7(b). The detection precision/recall area under curve (AUC) is a standard performance measure on the Weizmann Horses dataset. The AUC for our approach is 79.84%, which is comparable to the result 80.32% in Xiang et al. [21]. We compare to them as they also use the explicit shape model and matching based method for object detection. The AUC of classification based methods [12], [22] is 84.98% and 96%, respectively. We observe that classification based methods are bounding box classifiers and utilize significantly more information than matching based methods as ours. This explains why our detection rate and AUC is lower than [12], [22].

TABLE 1
Detection rate.

	Our method	Zhu et al. [20]	Shotton et al. [12]
Horses	93.97%	86.0%	95.20%
Cows	90.38%	88.6%	N/A

The proposed approach can not only succeed in extensive cluttered images, but also handles the problem of large range of scales and intra-class variability. This is demonstrated by several examples in Fig. 7. The images in the bottom right of Fig. 7(a) with red rectangles are the ones we fail to detect. The images of horses in Fig. 7(b) with red rectangles are false positives in the negative images provided by Shotton et. al. [12] to complement the Weizmann horse dataset. They

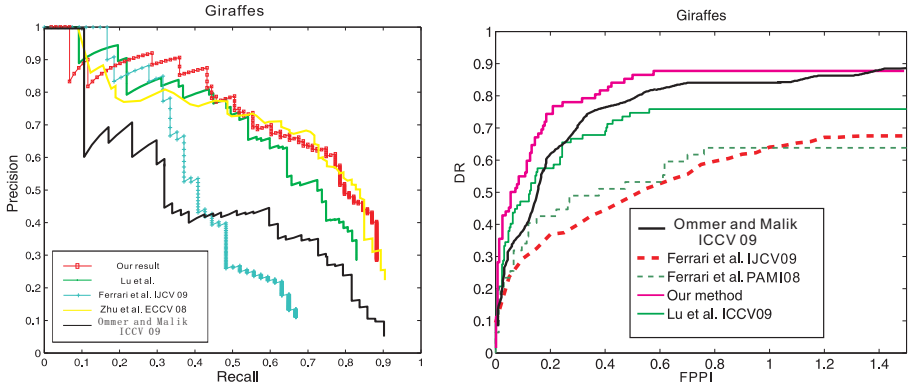
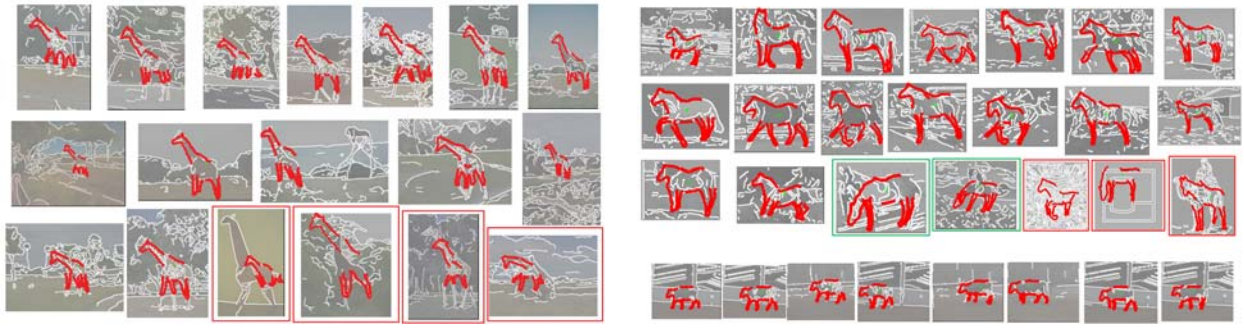


Fig. 6. Precision-recall curve and detection rate (DR) vs false positive per image (FPPI) curve for the class Giraffes in ETHZ dataset.



(a) (b)

Fig. 7. Examples of detection results for Giraffes, horses and cows.

show that the false positives in the negative set are caused by really very cluttered edges or by the structure of edges happening to match to the model very well. Interestingly, the rightmost false positive of horses is due to a camel, whose shape is very similar to that of a horse.

The statistical inference framework presented in this paper offers one important advantage compared to texture based and classification methods like [22]–[24]. It can localize object boundaries, rather than just bounding-boxes.

In order to quantify how accurately the output shapes match to true boundaries, we use the coverage and precision measures defined in [13]. Coverage is the percentage of points from ground-truth boundaries closer than a threshold t to the output shapes of the proposed approach. Reversely, precision is the percentage of points from output shapes closer than t to any point of ground-truth boundaries. As in [13] t is set to 4% of the diagonal of the ground-truth bounding box. The measures are complementary. Coverage captures how much of the object boundary has been recovered by the algorithm, whereas precision reports how much of the algorithm's output lies on the object boundaries. These measurements seem to be more suitable for evaluating shape based approaches than bounding-box evaluation, which cannot quantify the accuracy of contour localization. It is possible to have bounding-box intersection larger than 0.5 without having correctly identified the ground-truth object boundaries. Two examples of horse detection are shown in Fig. 7(b) with green rectangles.

The first two columns of Table 2 show coverage and precision averaged over all images of the class giraffes in ETHZ dataset in comparison to the results in [13]. We measure the coverage and precision for the correct detections at 0.4 FPPI, following [13]. The coverage of the proposed approach is over 11% better than [13], which shows that our approach can efficiently recover the true boundary of objects. The precision is a little lower than [13]. More importantly, the detection rate at our 0.4 FPPI is 86.75%. However, even for 20% bounding box intersection, the detection rate at 0.4 FPPI in [13] is only around 60% , which is significantly lower than our result.

For horses and cows, the coverage and precision are obtained over all correct detections. The third column of Table. 2 shows the coverage and precision of the proposed method on the Weizmann Horse dataset. As the edges are significantly worse than the ones provided for the giraffes, both measures are worse than the results on giraffes. The coverage and precision results for cow are shown in the fourth column of Table. 2. Due to less intra-shape variance, the precision is 92.02%, which is much higher than giraffes and horses. However, the coverage is only 73.86%. The main reason for the difference between these two values is that our model has a gap, since we removed the contour part representing the horse tail from the horse contour used for part decomposition. Thus, even if the model and object match perfectly, the coverage score cannot be perfect (see examples in Fig. 7).

TABLE 2
Accuracy of the boundary localization.

	Ours on giraffes	Results in [13] on giraffes	Ours on horses	Ours on cows
Coverage	79.4%	68.5%	77.5%	73.86%
Precision	74.6%	77.3%	61.7%	92.02%

8 Related Work

Particle filters (PF) are also known as sequential Monte Carlo methods (SMC) for model estimation based on simulation. There is large number of articles published on PF and we refer to two excellent books [25], [26] for an overview. To our best knowledge, the proposed PF framework with unordered observations is novel and has not been considered before by other authors. PF can be viewed as a powerful inference framework that is utilized in many applications. One of the leading examples is the progress in robot localization and mapping based on PF [1]. Particle filter (PF) has been used for object detection previously [27], [28]. They mainly utilize PF in the classical tracking/filtering scenario with a pre-defined order of states and observations. In contrast, our method explores different orders of states and observations on the fly and let particles follow the most informative orders, which is theoretically quite different from the traditional PF framework.

The proposed PF framework is also related to Gibbs sampling. The algorithm for Gibbs sampling was first introduced in the seminal paper by Geman and Geman [29]. Since then there have been many sampling algorithms like Gibbs sampler, e.g., Hot Coupling [30], Tree sampling, Swendsen-Wang sampling etc. But most of them assume restrictive conditional independencies. Recently Hamze et. al. proposed a very generic importance sampling method called Large Flip Importance Sampling (LFIS) to sample from the posterior [31]. The main motivation for their approach comes from N-Fold Way [32] and Tabu search [33], where they use heuristics to improve the sampling of the exponential state space using memory and heuristics to design good moves in the state space. Since the moves are no-longer MCMC in the traditional sense they introduce importance weights to the distinct states visited by N copies of the sampler. In this paper we combine the strengths of both PF and Gibbs sampler based approaches. In particular, our approach can be viewed as improved Gibbs sampler that employs the PF weighting scheme as well as the PF resampling scheme.

A preliminarily version of the proposed PF inference framework has also been applied to object detection and recognition in [15], where the shape similarity was based on direct matching of model contour parts to image contour fragments. The presented PF inference framework has been first applied to object detection and recognition in [10]. This article focused on theoretical presentation of the proposed PF inference framework and provides derivation of all key formulas as well as proves their theoretical properties.

Acknowledgments

The work has been supported in part by the NSF Grants IIS-0812118, BCS-0924164, the AFOSR Grant FA9550-09-1-0207, and the DOE Award 71498-001-09.

References

[1] S. Thrun, W. Burgard, and D. Fox, *Probabilistic Robotics*. The MIT Press Cambridge, 2005.
[2] N. Gordon, D. Salmond, and A. Smith, "Novel approach to nonlinear/non-gaussian bayesian state estimation," in *Radar and Signal Processing, IEE Proceedings of*, vol. 140, April 1993, pp. 107–113.
[3] J. Carpenter, P. Clifford, and P. Fearnhead, "Building robust simulation-based filters for evolving data sets," Dept. of Statistics, University of Oxford, Tech. Rep., 1999.
[4] N. Adluru, L. J. Latecki, M. Sobel, and R. Lakaemper, "Merging maps of multiple robots," in *ICPR*, 2008.
[5] A. Birk and S. Carpin, "Merging occupancy grid maps from multiple robots," in *Proceedings of the IEEE*, vol. 94, July 2006, pp. 1384 – 1397.
[6] B. K. P. Horn, H. M. Hilden, and S. Negahdaripourt, "Closed-form solution of absolute orientation using orthonormal matrices," *Journal of the Optical Society of America*, vol. 5, no. 7, pp. 1127–1135, 1988.
[7] B. K. P. Horn, "Closed-form solution of absolute orientation using unit quaternions," *Journal of the Optical Society of America*, vol. 4, no. 4, pp. 629–642, 1987.
[8] K. S. Arun, T. S. Huang, and S. D. Blostein, "Least-squares fitting of two 3-d point sets," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 9, no. 5, pp. 698–700, 1987.
[9] J. S. abd A. Blake and R. Cipolla, "Multi-scale categorical object recognition using contour fragments," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 30, no. 7, pp. 1270–1281, 2008.
[10] X. Yand and L. J. Latecki, "Weakly supervised shape based object detection with particle filter," in *ECCV*, 2010.
[11] E. Borenstein, E. Sharon, and S. Ullman, "Combining top-down and bottom-up segmentation," in *POVC*, 2004.
[12] J. Shotton, A. Blake, and R. Cipolla, "Multi-scale categorical object recognition using contour fragments," *IEEE Trans. on PAMI*, vol. 30, no. 7, pp. 1270–1281, 2008.
[13] V. Ferrari, F. Jurie, and C. Schmid, "From images to shape models for object detection," *IJCV*, vol. 87, no. 3, 2010.
[14] B. Leibe, A. Leonardis, and B. Schiele, "Combined object categorization and segmentation with an implicit shape model," in *Proceedings of the Workshop on Statistical Learning in Computer Vision*, Prague, Czech Republic, May 2004.
[15] C. Lu, L. J. Latecki, N. Adluru, X. Yang, and H. Ling, "Shape guided contour grouping with particle filters," in *ICCV*, 2009.
[16] Q. Zhu, L. Wang, Y. Wu, and J. Shi, "Contour context selection for object detection: a set-to-set contour matching approach," in *ECCV*, 2008.
[17] B. Ommer and J. Malik, "Multi-scale object detection by clustering lines," in *ICCV*, 2009.
[18] V. Ferrari, L. Fevrier, F. Jurie, and C. Schmid, "From images to shape models for object detection," *IEEE Trans. PAMI*, vol. 30, no. 1, pp. 36–51, 2008.
[19] S. Ravishankar, A. Jain, and A. Mittal, "Multi-stage contour based detection of deformable objects," in *ECCV*, 2008.
[20] L. Zhu, Y. Chen, and A. Yuille, "Learning a hierarchical deformable template for rapid deformable object parsing," *IEEE Trans. PAMI*, vol. 99, no. 1, 2009.
[21] X. Bai, X. Wang, L. J. Latecki, and Z. Tu, "Active skeleton for non-rigid object detection," in *ICCV*, 2009.
[22] J. Gall and V. Lempitsky, "Class-specific hough forests for object detection," in *CVPR*, 2009.
[23] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *CVPR*, 2005.
[24] C. Desai, D. Ramanan, and C. Fowlkes, "Discriminative models for multi-class object layout," in *ICCV*, 2009.
[25] A. Doucet, N. D. Freitas, and N. Gordon, *Sequential Monte Carlo Methods in Practice*. Springer-Verlag, 2001.
[26] J. Liu, *Monte Carlo strategies in Scientific Computing*. Springer-Verlag, 2001.
[27] S. Ioffe and D. Forsyth, "Probablistic methods for finding people," *IJCV*, 2001.
[28] ——, "Finding people by sampling," in *ICCV*, 1999.
[29] S. Geman and D. Geman, "Stochastic relaxation, gibbs distributions, and the bayesian restoration of images," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 6, no. 6, p. 721741, 1984.

[30] F. Hamze and N. de Freitas, "Hot coupling: A particle approach to inference and normalization on pairwise undirected graphs of arbitrary topology," in *NIPS*, 2005, pp. 1–8.
[31] ——, "Large-flip importance sampling," in *UAI*, 2007.
[32] A. B. Bortz, M. H. Kalos, and J. L. Lebowitz, "A new algorithm for monte carlo simulation of ising spin systems," *Journal of Computational Physics*, vol. 17, pp. 10–18, 1975.
[33] G. F., "Tabu search - part i," *ORSA Journal on Computing*, vol. 1, no. 3, pp. 190–206, 1989.