SMCP³: Sequential Monte Carlo with Probabilistic Program Proposals

Alexander K. Lew* [1], George Matheos* [1,2], Tan Zhi-Xuan[1], Matin Ghavamizadeh[1], Nishad Gothoskar[1], Stuart Russell[2], Vikash K. Mansinghka[1]

*Equal Contribution [1] MIT [2] UC Berkeley

1. Introduction

SMCP³ is a new family of SMC algorithms. It **generalizes**:

- Particle Filtering
- Resample-Move SMC
- Move-Reweight SMC
- SMC Samplers
- Annealed importance sampling

by supporting general **probabilistic programs** as proposal distributions.

SMCP³ also **automates the implementation** of SMC given probabilistic programs for the target probabilistic model and the proposal distributions.

Sequential Monte Carlo (SMC)

Given a sequence of probabilistic models $p_t(x_t, y_{1:t})$, and observations $y_{1:t}$...

$\texttt{SMC}(y_{1:t}) :=$

1. $\{(x_1^i, w_1^i)\}_{i=1}^N \leftarrow \texttt{InitializeParticles}(y_1)$
2. For $t = 2, \ldots, T$:
   a. $\{(x_{t-1}^i, w_{t-1}^i)\}_i \leftarrow \texttt{Resample}(\{(x_{t-1}^i, w_{t-1}^i)\}_i)$
   b. For $i = 1, \ldots, N$:
      i. $(x_t^i, \hat{w}_t^i) \leftarrow \texttt{ParticleUpdater}(x_{t-1}^i, y_{1:t})$
      ii. $w_t^i \leftarrow w_{t-1}^i \hat{w}_t^i$

...at each time t, SMC infers:

1. $p_t(x_t = \cdot | y_{1:t})$
2. $p_t(y_{1:t})$

...by generating a *properly weighted particle collection* $\{(x_t^i, w_t^i)\}_{i=1}^N$.

Properly weighted means: the weights correct for the mismatch between the distribution used to generate each particle x_t^i, and $p_t(x_t = \cdot | y_{1:t})$. (For formal definition, see "Theory".)

2. Our contribution: SMCP³

SMCP³ is derived by extending the "SMC Sampler" particle update to support general probabilistic program proposals—not just proposals with tractable-to-compute probability densities.

Density $k : X \to Y$

$X \xrightarrow{k} Y$

SMC Sampler (Del Moral et al. 2006b)

Components:
Density $k : X_{t-1} \to X_t$
Density $l : X_t \to X_{t-1}$

Particle update:
$x_t \sim k(x_{t-1} \to \cdot)$

Weight update:
$\hat{w} \leftarrow \frac{\tilde{p}_t(x_t)}{\tilde{p}_{t-1}(x_{t-1})} \frac{l(x_t \to x_{t-1})}{k(x_{t-1} \to x_t)}$

Probabilistic Program $K : X \rightsquigarrow Y$

Components:
Density $q_K : X \to U_K$
Function $f_K : X \times U_K \to Y$

Sample operation:
$u_K \sim q_K(x \to \cdot);\ y \leftarrow f(x, u_K)$

SMCP³ Update (this paper)

Components:
Probabilistic Program $K : X_{t-1} \rightsquigarrow X_t \times U_L$
Probabilistic Program $L : X_t \rightsquigarrow X_{t-1} \times U_K$

Particle update:
$u_K \sim q_K(x_t \to \cdot)$
$(x_t, u_L) \leftarrow f_K(x_{t-1}, u_K)$
return x_t

Weight update:
$\hat{w} \leftarrow \frac{\tilde{p}_t(x_t)}{\tilde{p}_{t-1}(x_{t-1})} \frac{q_L(x_t \to u_L)}{q_K(x_{t-1} \to u_K)} \left(\frac{d\xi_L^t}{d(\xi_K^t \circ f_K^{-1})}(x_t, u_L)\right)$
(see Theorem 1)

Unlike proposal densities, probabilistic program proposals may sample auxiliary random choices, may apply deterministic transformations, and need not admit densities over their outputs.

Role of the SMCP³ Proposals

The **forward proposal** K sees an old particle x_{t-1} (and the data $y_{1:t}$) and proposes an updated particle x_t.

The **backward proposal** L *inverts the forward proposal*. Given x_t (and $y_{1:t}$), L proposes what x_{t-1} the forward proposal may have received as input, and what random choices u_K the forward proposal may have made while updating x_{t-1} into x_t.

The **forward proposal** must also output the set of random choices u_L the backward proposal would make to invert it.

The SMCP³ Particle Update

$\texttt{SMCP3ParticleUpdater}_{K,L}(x_{t-1}^i, y_{1:t}) :=$

1. $(q_K, f_K) \leftarrow K$
2. $(q_L, f_L) \leftarrow L$
3. $u_K \sim q_K(x_{t-1}^i \to \cdot; y_{1:t})$
4. $(x_t^i, u_L) \leftarrow f_K(x_{t-1}^i, u_K, y_{1:t})$
5. $\hat{w}_t^i \leftarrow \frac{p_t(x_t^i, y_{1:t}) q_L(x_t^i \to u_L; y_{1:t})}{p_{t-1}(x_{t-1}^i, y_{1:t-1}) q_K(x_{t-1}^i \to u_K; y_{1:t})} \times |\det \mathrm{Jac}(\tilde{f}_K^{y_{1:t}})(x_{t-1}^i, u_K)|$
6. Return $(x_t^i, \hat{w}_t^i)$

3. Example

Probabilistic model

$z_0 = 0$

For $t > 0$:

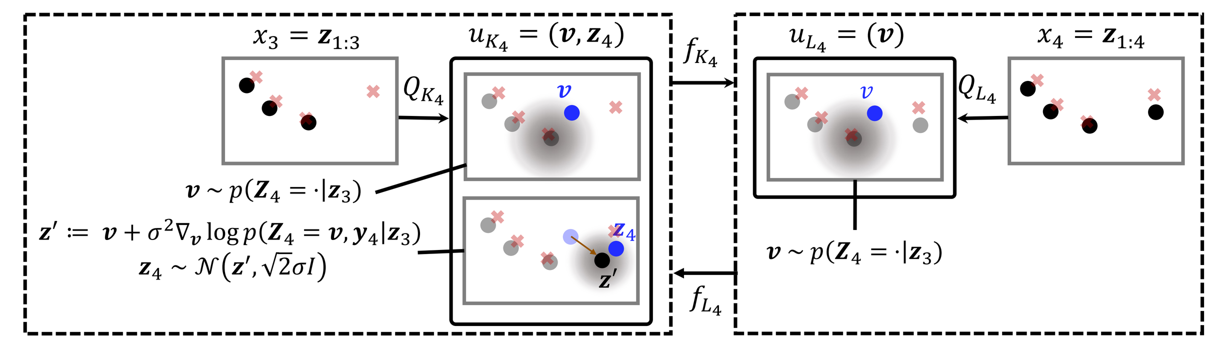
$z_t \sim \mathcal{N}(z_{t-1}, 1.0)$

$y_t \sim \mathcal{N}(z_t, 1.0)$

```
@gen function model(t)
  z = 0
  for step in 1:t
    z = {"z$(step)"} ~ normal(z, 1.0)
    y = {"y$(step)"} ~ normal(z, 1.0)
  end
end
```

In this example, the latent state x_t of inference is a *trajectory*: $x_t = z_{1:t}$

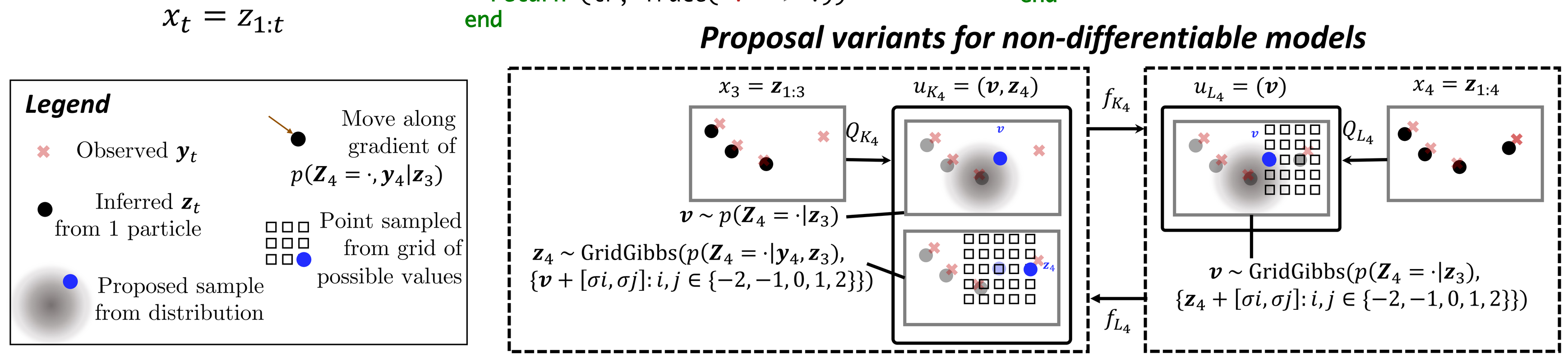
SMCP³ Proposals (for differentiable models)



```
@gen function K(tr, t, new_obs)
  # Sample v from dynamics model
  prev_z = tr["z$(t-1)"]
  v = {"v"} ~ normal(prev_z, 1.0)
  # Unadjusted Langevin Ascent to sample z
  z = {"z"} ~ ULA(v, z_prev, new_obs)
  # Update trace with newly proposed z
  tr["z$(t)"] = z
  # Return proposed trace & aux. randomness
  return (tr, Trace("v" => v))
end
```

```
@gen function L(tr, t)
  # Guess the aux. var that K sampled
  prev_z = tr["z$(t-1)"]
  v = {"v"} ~ normal(prev_z, 1.0)
  # Return previous step's trace, and trace
  # of K that would lead to this trace.
  return (Trace(["z$(i)" => tr["z$(i)"]
    for i in 1:t-1]...), Trace(
      "v" => v, "z" => tr["z$(t)"] ) )
end
```

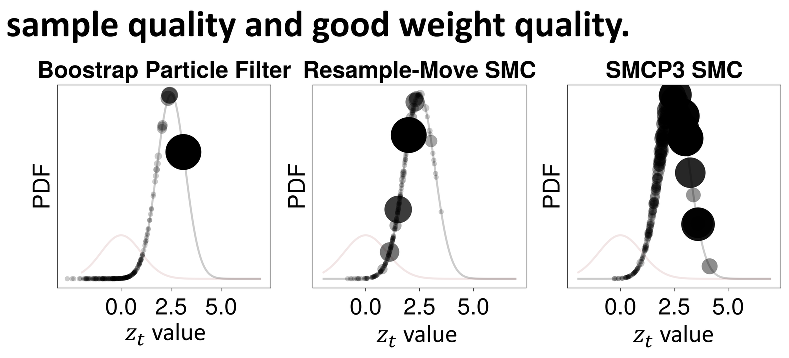
Proposal variants for non-differentiable models



SMCP3 can help to simultaneously achieve good sample quality and good weight quality.

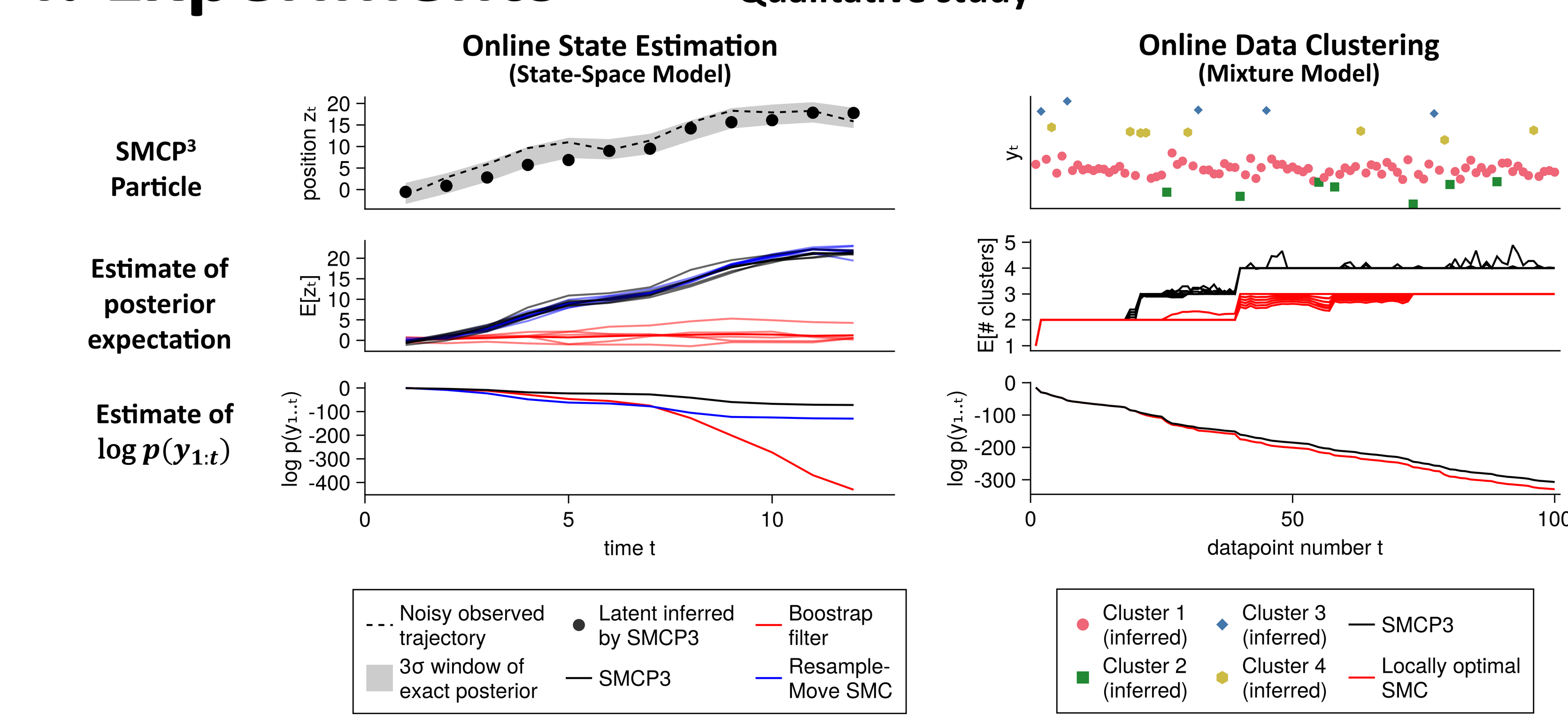
- *Sample quality* = do the particles land in high-posterior-probability regions?
- *Weight quality* = do the particle weights accurately measure the relative quality of the samples?

In more restricted families of SMC algorithms it can be hard to design algorithms with both high sample quality and high weight quality. SMCP³ has new degrees of freedom that help to achieve both.

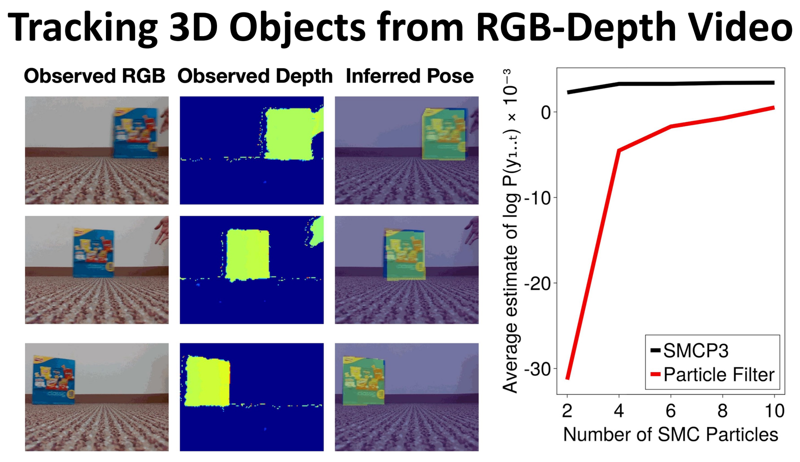


4. Experiments

Qualitative study



Tracking 3D Objects from RGB-Depth Video



Quantitative study on large datasets

	Est. of log $P(\mathbf{y}_{1:T})$
State-space model (Sec. 5.1)	
100D trajectory (synthetic)	
Bootstrap PF	-4347.67 ± 83.21
Resample-Move SMC	-2828.29 ± 22.78
SMCP³ ULA	**-2271.03 ± 10.86**
Mixture model (Sec. 5.2)	
Medicare data	
Locally Optimal SMC	-40851.15 ± 0.98
Resample-Move Split/Merge	-16898.79 ± 1117.31
SMCP³ Split/Merge	**-13882.47 ± 0.24**

We measure inference performance via SMC's average estimate of $\log p_t(y_{1:t})$. $\log p_t(y_{1:t})$ is a quantity SMC is commonly used to estimate. By Jensen's inequality, approximation error in SMC results in the average $\log p_t(y_{1:t})$ estimate being *too low*, so **higher estimates indicate more accurate inferences**. It also turns out bounded error in mean $\log p_t(y_{1:t})$ estimates implies bounded error in posterior inferences (Lew et al. 2022).

5. Theory

- **Proper weighting.** SMCP³ computes proper particle weights (Thm. 1).
  - Formally, this means: for any integrable f, $E[w_t^i f(x_t^i)] = p_t(y_{1:t}) E_{x_t \sim p_t(x_t = \cdot | y_{1:t})}[f(x_t)]$.
- **Central limit theorem.** SMCP³ converges as the number of particles $N \to \infty$ (Prop. 2).
- **The locally optimal L inverts K.** Given any forward proposal K, we precisely characterize the backward proposal which minimizes the variance of the incremental particle weights (Prop. 1).
  (This is an analogue to a similar theorem from Del Moral et al 2006a.)

6. Automation

A probabilistic programming system can automate SMCP³, given probabilistic programs for the target model and the proposals. Particles are updated by running the forward proposal, and particle weights are computed via density computations & AD.

$$\hat{w}_t = \boxed{\frac{p_t(x_t, y_{1:t}) q_L(x_t \to u_L; y_{1:t})}{p_{t-1}(x_{t-1}, y_{1:t-1}) q_K(x_{t-1} \to u_K; y_{1:t})}} \times \boxed{|\det \mathrm{Jac}(\tilde{f}_K^{y_{1:t}})(x_{t-1}, u_K)|}$$

computed via density expansions

$$q(a \to u) = \prod_{v \in u} q(v | a, u_{Pa(v;u)})$$

computed via automatic differentiation

Related work

Misc. "Recursive Monte Carlo and Variational Inference with Auxiliary Variables", Lew et al. 2022.

SMC Algorithm families generalized by SMCP³. SMC Samplers, "Sequential Monte Carlo Samplers", Del Moral et. al 2006a; "Sequential Monte Carlo for Bayesian computation", Del Moral et. Al 2006b.; Resample-Move SMC, "Following a Moving Target", Gilks and Berzuini, 2001. Move-Reweight SMC, Marques and Storvik, 2013. SMC with Transformations, Everitt et al., 2020. Annealed Importance Sampling, Neal, 1998.

Probabilistic programming languages supporting automated SMC with proposal *densities* written as probabilistic programs. Gen (Cusumano-Towner et al. 2018, 2019), Pyro (Bigham et al., 2019) Birch (Murray, 2013; Murray and Schon, 2018), Inference Combinators (supports non-density proposals with auxiliary variables; equivalent to a restricted subset of SMCP³ forcing the use of sub-optimal backward proposals), "Learning Proposals for Probabilistic Programs with Inference Combinators", Stites et al. 2021.

***MCMC* with support for proposals with auxiliary sampling and deterministic transformations ("involutive MCMC").** "A general perspective on the Metropolis-Hastings kernel", Andrieu et al. 2020. "Automating Involutive MCMC using Probabilistic and Differentiable Programming" (Cusumano-Towner et al. 2020), "Involutive MCMC: a unifying framework", Neklyudov et al. 2020.

Paper | Talk | Open Source Library