Adapting Particle Filter Algorithms to Many-Core Architectures

Mehdi Chitchian*, Alexander S. van Amesfoort*, Andrea Simonetto†, Tamás Keviczky†, Henk J. Sips*

*Parallel and Distributed Systems Group
†Delft Center for Systems and Control

Delft University of Technology
Outline

1. Background
2. Parallel Design and Implementation
3. Application and Experiments
4. Conclusions
Outline

1. Background

2. Parallel Design and Implementation

3. Application and Experiments

4. Conclusions
Bayesian State Estimation

Estimating an unknown quantity $x$ through noisy observation $z$

Applications in:
- Robotics
- Computer Vision
- Econometrics
Capture uncertainty in a probability distribution function:

\[ p(x) \]

Fast analytical Bayesian filters only exist for (near-)linear systems with Gaussian noise

Nature is rarely linear
Particle Filter

- Monte Carlo-based estimation method
- Approximate the posterior by a set of random samples $\mathcal{X}$ with associated weights $\mathcal{W}$
- Each particle (sample) represents an instantiation of the state (hypothesis)

\[
\mathcal{X}_t = \{x_t^{[1]}, x_t^{[2]}, \ldots, x_t^{[M]}\}
\]

\[
\mathcal{W}_t = \{\omega_t^{[1]}, \omega_t^{[2]}, \ldots, \omega_t^{[M]}\}
\]
Particle Filter

Flexibility

No restrictions on system model
- Suitable for non-linear and/or non-Gaussian systems

Accuracy

Particle filters outperform other estimation methods given enough particles
Particle Filter

Stages

1. **Sampling** Random state propagation
2. **Importance Weights** Incorporate measurements
3. **Resampling** Prevent degeneration
Particle Filter

Sampling

State Space

<table>
<thead>
<tr>
<th>Time</th>
<th>t</th>
<th>t+1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Receive Noisy Measurement</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Importance Weighting

Estimated State Value

Particle Filter
**Degeneracy Problem**

- The variance of the weights increases over time
- All but a single particle have negligible weight
- Wasted computation as these particles do not contribute to the estimate

**Resampling**

- Create new particle set
- Draw with replacement from original particle set
- Particle weight determines probability of selection
Particle Filter

- Sampling
- Importance Weighting
- Resampling

State Space

Receive Noisy Measurement

Estimated State Value

Time

$t$

$t+1$
Particle Filter

Powerful and accurate, but computationally intensive

Thousands of particles needed even for moderately sized problems

Limited practical use in real-time estimation applications
Outline

1. Background
2. Parallel Design and Implementation
3. Application and Experiments
4. Conclusions
Parallel Design

**Research Goal**
Enable real-time particle filtering for complex estimation problems using modern many-core architectures

**Hardware Trends**
Trending towards increasing core counts over larger and more complex core designs
Parallel Design

**The Good**
Sampling and weight calculation stages are trivially parallel

**The Bad**
Resampling requires global communication
- Most attempts either forgo global resampling or perform this step sequentially
Distributed Particle Filter

Construct a network of smaller particle filters (sub-filters)

Sub-filters operate independently and only direct neighbours communicate

Communication limited to exchanging a few representative particles

Scale the total number of particles by increasing the number of sub-filters
Distributed Particle Filter

Filter Parameters
- Number of particles per sub-filter
- Number of sub-filters
- Exchange scheme
- Number of particles per exchange
Distributed Particle Filter

Particle Exchange Schemes

- All-to-All
- Ring
- 2D Torus
Particle Filter Framework: Esthera

**Esthera**

Generic particle filter framework for estimation problems

https://github.com/alxames/esthera

- Separation of model specific and generic filter code
- Particle filter network parameters entirely configurable
- Offline filtering accuracy and performance measurements
Particle Filter Framework: Esthera

Kernels

- Sampling/Importance Weighting (model specific code)
- Local Sorting
- Global Estimate
- Particle Exchange
- Resampling

- Pseudo-Random Number Generation (batch mode)
Particle Filter Framework: Esthera

Execution

- Single particle per thread
- Each sub-filter mapped onto a thread block
- Synchronisation between threads for sorting and resampling

Data

- Particle data and weights stored in global memory
- Weights are loaded in local memory for efficient sorting and resampling
- Particle data itself does not fit into local memory
Outline

1. Background
2. Parallel Design and Implementation
3. Application and Experiments
4. Conclusions
Robotic Arm Setup
Robotic Arm Setup

- Tracking an object using a camera mounted at the end effector
- Noise in both measurement and actuation
- Highly non-linear system model
- Variable number of joints in simulation
Many-Core Architectures

- GPGPU
- Multi-core CPU

OpenCL implementation for both GPGPU and multi-core CPU

Sequential implementation of a centralised particle filter in C
Achieved Update Rate

![Graph showing achieved update rate vs number of particles for different GPUs and CPUs.]

- Distributed (OpenCL):
  - GTX 680
  - GTX 580
  - HD 7970
  - HD 6970
  - 2x E5-2650
  - i7-2820QM

- Centralized (C):
  - 2x E5-2680
  - i7-2820QM
Filtering Accuracy

Centralized/Distributed
(particles p/filter)

- distr. (2)
- distr. (4)
- distr. (8)
- distr. (16)
- distr. (32)
- distr. (64)
- distr. (128)
- distr. (256)
- distr. (512)
- centralized
For all filter sizes, certain configurations of the distributed filter perform no worse than (or even outperform) their centralised counterpart.
Filter Parameters: Exchange Scheme

All-to-All

Ring

Particles p/filter
- 4
- 8
- 16
- 32
- 64
- 128
- 256
- 512

Number of filters

Estimation error [-]

Particles p/filter
- 4
- 8
- 16
- 32
- 64
- 128
- 256
- 512

Number of filters

Estimation error [-]
Filter Parameters: Exchange Scheme

Ring

2D Torus
Filter Parameters: Number of Exchanged Particles

$t = 0$

$t = 1$
Filter Parameters: Number of Exchanged Particles

\[ t = 1 \]

\[ t = 2 \]
Filter Parameters: Optimal Configuration

No single optimal filter configuration

Rules of thumb:
- Low particle settings: limited communication, lower connectivity
- High particle settings: increased communication, more connectivity
Conclusions

- Real-time particle filtering using current generation GPU
- At least one order of magnitude faster than current state of the art
- Scalable design based on sub-filters matching hardware trends of increasing core counts
- Particle filter accuracy does not suffer from distributed design and limited communication
- Introducing the generic particle filtering framework ‘Esthera’ for custom estimation problems
Implement a particle filter for your estimation problems with the Esthера framework:

https://github.com/alxames/esthера