ADAPTIVE SEQUENTIAL MONTE CARLO APPROACH FOR REAL-TIME APPLICATIONS

Thomas C.P. Chau[1]*, Wayne Luk*[1]*, Peter Y.K. Cheung*[2]*, Alison Eele*[3]*, Jan Maciejowski*[3]

Department of Computing[1], Department of Electrical and Electronic Engineering[2]
Imperial College London, United Kingdom
Department of Engineering, University of Cambridge, United Kingdom[3]
email: {c.chau10, w.luk, p.cheung}@imperial.ac.uk, {aje46, jmm1}@cam.ac.uk

ABSTRACT

This paper presents an adaptive Sequential Monte Carlo approach for real-time applications. Sequential Monte Carlo method is employed to estimate the states of dynamic systems using weighted particles. The proposed approach reduces the run-time computation complexity by adapting the size of the particle set. Multiple processing elements on FPGAs are dynamically allocated for improved energy efficiency without violating real-time constraints. A robot localisation application is developed based on the proposed approach. Compared to a non-adaptive implementation, the dynamic energy consumption is reduced by up to 70% without affecting the quality of solutions.

1. INTRODUCTION

Sequential Monte Carlo method (SMC) is an efficient numerical estimation technique that is applied to solve dynamic problems involving non-Gaussianity and non-linearity, where analytical solutions are not possible [1]. SMC is useful in calculating the state of a system from a set of observations which arrive sequentially in time and are corrupted by noise. This method arises in a range of applications, especially in positioning, tracking and navigation [2–4].

SMC is also known as particle filtering since the key idea is to represent probability densities by sets of particles. The quality of results relies on the number of particles being employed. However, the larger the size of the particle set, the higher the computation cost. Despite previous attempts to accelerate SMC with hardware, design approach for the real-time domain lacks attention. Real-time applications require state estimation to be ready at regular time intervals and many of the real-time systems are power-constrained. Under such scenarios, the challenges turn out to be achieving energy reduction, better resource usage and improved quality of results within the real-time bound, rather than targeting throughput.

In this paper, an adaptive algorithmic and architectural approach using reconfigurable hardware is proposed for SMC real-time applications. To the best of our knowledge, this work is the first to target energy efficient real-time SMC applications. The proposed approach can be applied to general applications in the SMC domain which share common processing steps, and where properties of a particular application are customised by reconfigurability of FPGA. The architecture makes use of multiple processing elements (PEs) to process particles. Each PE is implemented by a processor or dedicated hardware logics on FPGA. To address the challenge of energy reduction within real-time bound, an adaptive algorithm is employed to dynamically adjust the size of particle set and the number of active PEs. A PE is activated based on the number of particles distributed to it.

The novel contributions of this paper include:

1. Adaptive SMC algorithm: The computation complexity of SMC is reduced through adapting the size of particle set dynamically in a predictable manner. The quality of results is maintained.

2. Energy efficient resource allocation: PEs are allocated dynamically for reduced energy consumption without violating real-time constraints.

3. Prototype: A robot localisation application is implemented on an FPGA based on the proposed adaptive SMC approach. Compared to a non-adaptive implementation, the dynamic energy consumption is reduced by 35-70%.

2. RELATED WORK

Design methodologies for SMC in hardware have been proposed by several research groups. A detailed description of SMC is available in [1]. Sankaranarayanan et al. [5] have proposed a modified particle filtering algorithm to allow higher flexibility and scalability for hardware implementations. Though the resampling process with the SMC is removed, their approach maintains high quality of results over traditional particle filters. Saha et al. [6] have presented a parameterisable approach for FPGA implementation of particle filters. They have provided a model definition of

particle filters and have studied the trade-off of FPGA resources and execution speed. In [7], Happe et al. have developed a SMC approach for hybrid CPU/FPGA platform. Using a multi-threaded programming model, computation is switched between hardware and software during run-time to react to performance requirements. However, the above approaches target SMC for static number of particles. The proposed approach differs through adapting the particle set size during run-time.

Adaptive particle filters have been proposed to improve performance or quality of state estimation by controlling the number of particles dynamically. Likelihood-based adaptation [8] determines the number of particles such that the sum of weights exceeds a pre-specified threshold. Kullback Leibler distance (KLD) sampling has been proposed by Fox et al. in [9]. It offers better quality results than likelihood-based approach. The number of particles are determined in order to bound the error of posterior estimation. Park et al. [10] improve the KLD sampling by adjusting the variance and gradient data to generate particles near the high likelihood region. These methods introduce data dependencies in the sampling and importance steps making them difficult for parallel hardware implementation. Bolic et al. [11] proposed a particle filter that changes the number of particles dynamically based on estimation quality. Decisions about performing resampling are based on the number of particles. This work has extended their techniques for the real-time domain. Liu et al. [12] have developed a power adaptive computing system to utilise the number of processing units subject to power supply constraints, but have not addressed real-time properties.

3. ADAPTIVE SMC APPROACH

The proposed approach eliminates particles with low weights such that the particle set size decreases in successive iterations. Resampling is not performed until the number of particles drops below a threshold value. When the estimation quality is good, the number of particles decreases slowly. When there are large deviations in observation, the particle set size decreases quickly.

Fig. 1 shows the architectural flow of the proposed adaptive SMC approach. Compared to traditional SMC approaches, three stages have been added: reduction, decision and resource allocation. At design time reconfigurability is employed to customise the system for a particular application. For example, the maximum number of particles, the sampling function and weight calculation vary for different problems. During run-time, the dotted lines are control signals to activate and deactivate PEs. The functions of each block are discussed below.

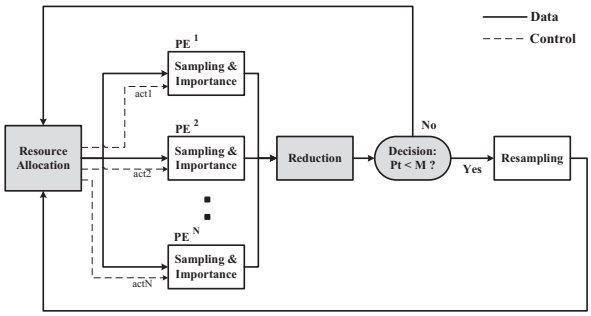

Fig. 1: Adaptive SMC architectural flow (Sampling & Importance is performed in PE^1 to PE^N, others are performed in PE^0)

Sampling and Importance Blocks: During *sampling*, a new particle set is drawn to form a prediction of current state. The likelihood of each particle is calculated in *importance* to indicate whether the current measurement matches the predicted state. Then a weight is assigned to each particle. The higher the likelihood, the higher the weight is.

Since particles processed in samling and importance are independent, the calculation can be parallelised. A set of PEs $\{PE^n|n = 1, ..., N\}$, which each can be of different implementations and speed, would be assigned to process particles simultaneously. At time t, there are ρ^n particles allocated to PE^n. Particles of each PE are stored in its shared memory for access and update by subsequent stages. As soon as a PE has finished processing its own set of particles, it is deactivated to reduce energy consumption.

Reduction Block: The objective of *reduction* is to eliminate particles with low weight. If the particle set size drops below the threshold, resampling process is triggered leading to a larger execution time. Since unpredictable behaviour is not desired for real-time applications, the minimum number of particles which remain after reduction is constrained to $\frac{1}{K}$ of the original. The value of K is user-defined in order to limit the particle reduction rate, where integers between 2 to 10 are advisable. To achieve this, the particles are sorted in descending order of weight and the first $\frac{1}{K}$ of particles are always kept.

Resampling Block: During *resampling*, particles with higher weights are replicated and the number of particles with lower weights are reduced. Resampling is performed when the number of particles is under the threshold value M. The value of M is decided empirically to balance the accuracy of state estimation and the frequency that resampling happens. After resampling, the number of particles is restored to the initial value.

Resource Allocation Block: During design time, the amount of PEs is defined to fulfill the real-time processing require-

ment for the largest particle set size. Reducing the particle set size during run-time introduces idle processing power. Therefore, the resource allocation stage distrubuted particles to PEs such that all PEs are activated at the beginning of each step and are deactivated once computation finishes.

There are P particles in total, and PE^n is assigned ρ^n particles. The energy consumption of each PE is equalised within the real-time constraint T_{RT}. Instead of distributing particles for the fastest speed, every PE gets its set of particles with both speed (S^n) and power (PW^n) taken into account. The allocation considers the total processing time $T(P)$ to avoid violating the real-time constraints. For example, a faster PE would not necessarily be allocated more particles because it could draw too much power. The way to determine the particle distribution ratio at design time is illustrated in Eq. 1.

$$\begin{aligned} &\text{Determine} \quad \{\rho^n | n = 1, ..., N\} = P \cdot \frac{\frac{S^n}{PW^n}}{\sum_{k=1}^{N} \frac{S^k}{PW^k}} \\ &\text{subject to} \quad \sum_{n=1}^{N} \rho^n = P \text{ and } T(P) \leq T_{RT} \end{aligned} \tag{1}$$

4. EVALUATION

The proposed adaptive SMC approach can be applied on platforms with multiple PEs. For demonstration purpose, a robot localisation system using SMC is implemented on an Altera DE4 development board. The board accommodates a Stratix IV EP4SGX530 FPGA clocked at 100MHz.

Localisation of mobile robots using SMC has been proposed in [2]. SMC is demonstrated to be a good choice for this application domain [13]. Given a priori learned map, the robot receives sensor values and makes a movement at regular time intervals, so the computation is subject to real-time constraints.

The PEs in the sampling and importance blocks are implemented using 7 NIOS II processors. Single precision floating-point data type is employed. It is straight forward to have a homogeneous system with identical PEs, but in practice, PEs could be implemented as hardware accelerators with various performances. Therefore, in this experiment, the processors are configured with different settings as shown in Tab. 1. PE^1 to PE^7 have local memories for program space and data memories for information of particles. A master processor (PE^0) is implemented using another NIOS II processor which is responsible for reduction, resampling and resource allocation. The processors are connected in a star topology through Avalon interfaces [14], where PE^0 can access the data memory of other processors. PE^0 can deactivate other processors by issuing *interrupts* and *wait requests*. The whole system occupies 22834 ALUTs (5%), 14481 registers (3%), 608 M9K memory blocks (48%) and 64 M144K memory blocks (100%).

Table 1: Properties of PEs

PE	S [a]	PW [b]	I-cache [c]	D-cache [d]	Grade [e]
0	16.5	36	64K	64K	f
1	16.5	36	64K	64K	f
2	16	16	16K	16K	f
3	13	12	4K	4K	f
4	10.8	11	1K	1K	f
5	10.6	11	512	512	f
6	8.3	10	512	/	s
7	1	1	/	/	e

[a] Normalised speed.
[b] Normalised power consumption.
[c] Instruction cache size in bytes.
[d] Data cache size in bytes.
[e] f - fast, s - standard, e - economy. Refer to [15] for details.

The system is initialised with 1000 particles. The size of map is 36m x 36m and the speed of robot is 0.1m/s. These parameters are comparable with those used in a similar robot localisation application [2]. The position is updated every 30s which is the real-time bound.

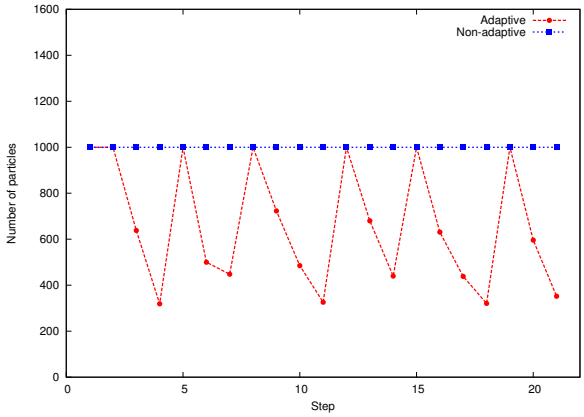

Fig. 2: Particle count and activation of PEs

The variation of the number of particles at each time step is shown in Fig. 2. The lines show the number of particles which the adaptive SMC approach is compared with the non-adaptive one. Starting from 1000 particles, the adaptive approach shows drops in the particle count in subsequent steps. When the number of particles is below 300, particles are resampled to the initial count. For the non-adaptive approach, the particle set size is fixed at 1000.

Fig. 3 show the dynamic energy consumption (bars) and execution time (lines) respectively. The energy is calculated based on the current measurement using an ADC on the FPGA board. The adaptive approach shows an average of 56% energy reduction, which is contributed by deactivating the PEs when they have finished the job before the timing margin. For the non-adaptive approach, the particle set size

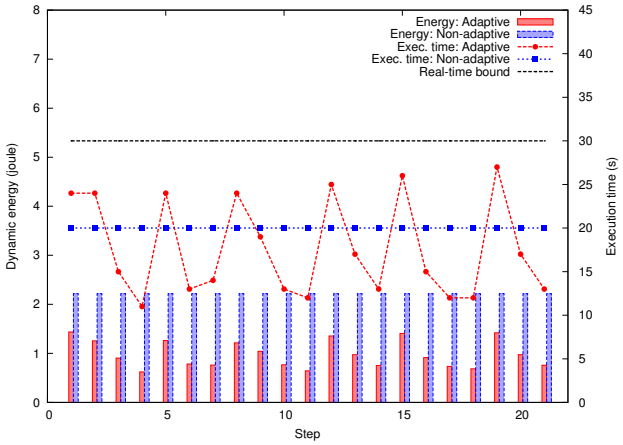

Fig. 3: Dynamic energy and execution time

is fixed and all PEs are always activated. In summary, the shorter the execution time, the lower the energy consumption. The adaptive approach has dynamic energy reduction between 35 and 70%. If the threshold for resampling is lowered, more energy saving could be achieved.

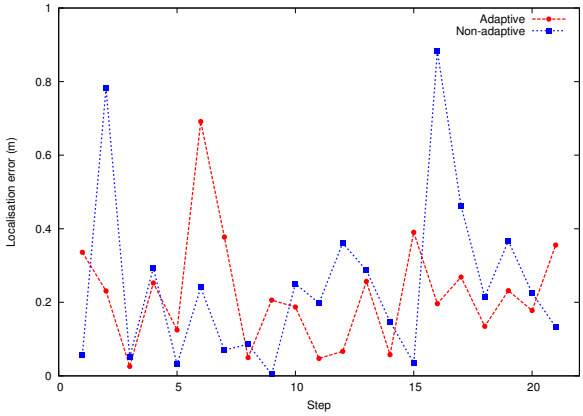

Fig. 4: Localisation error of robot

Fig. 4 shows the localisation errors compared to the actual positions. The maximum errors of all schemes are below the 1m tolerance level [2]. The average errors of the adaptive and non-adaptive approaches are 0.22m and 0.26m, respectively. Considering the randomness of different trials, the adaptive approach does not show significant deviation from the non-adaptive one.

5. CONCLUSION

This paper presents an adaptive SMC approach for real-time applications and proposes a resource allocation scheme addressing energy efficiency. Through a case study on robot localisation, the proposed approach significantly reduces dynamic energy consumption compared with the non-adaptive one, while maintaining quality of results. Ongoing and future work includes applying the adaptive approach to heterogeneous accelerators for high performance applications. Computational intensive data paths will be implemented in hardware accelerators to improve performance and power efficiency. Extension to cover run-time reconfiguration of FPGAs is also of interest, such as customising the PEs dynamically for power and performance trade-off at run-time.

Acknowledgment

This work was supported in part by the European Union Seventh Framework Programme under grant agreements number 248976, 257906, 287804, the UK EPSRC grant number EP/G066477/1, and by the Croucher Foundation and Altera.

6. REFERENCES

[1] A. Doucet *et al.*, *Sequential Monte Carlo methods in practice*. Springer Verlag, 2001.

[2] D. Fox *et al.*, "Monte Carlo localization: Efficient position estimation for mobile robots," in *Proc. Conf. on AAAI*, 1999, pp. 343–349.

[3] J. P. Villiers *et al.*, "Particle predictive control," *Journal of Statistical Planning and Inference*, vol. 141, pp. 1753–1763, 2011.

[4] A. Eele and J. Maciejowski, "Comparison of stochastic optimisation methods for control in air traffic management," in *Proc. IFAC World Congress*, 2011.

[5] A. Sankaranarayanan *et al.*, "Algorithmic and architectural design methodology for particle filters in hardware," in *Proc. Int. Conf. on Computer Design*, 2005, pp. 275–280.

[6] S. Saha *et al.*, "Parameterized design framework for hardware implementation of particle filters," in *Proc. Int. Conf. on ASSP*, 2008, pp. 1449–1452.

[7] M. Happe *et al.*, "An adaptive sequential monte carlo framework with runtime HW/SW repartitioning," in *Proc. Int. Conf. on FPT*, 2009, pp. 175–182.

[8] D. Koller *et al.*, "Using learning for approximation in stochastic processes," in *Proc. ICML*, 1998, pp. 287–295.

[9] D. Fox, "Adapting the sample size in particle filters through kld-sampling," *Int. Trans. on Robotics*, vol. 22, no. 12, pp. 985–1003, 2003.

[10] S.-H. Park *et al.*, "Novel adaptive particle filter using adjusted variance and its application," *IJCAS*, vol. 8, no. 4, pp. 801–807, 2010.

[11] M. Bolic *et al.*, "Performance and complexity analysis of adaptive particle filtering for tracking applications," in *Proc. Asilomar Conf. on SS&C*, vol. 1, 2002, pp. 853–857.

[12] Q. Liu *et al.*, "Power adaptive computing system design in energy harvesting environment," in *Proc. Int. Conf. on SAMOS*, 2011, pp. 33–40.

[13] J. Hightower and G. Borriello, "Particle filters for location estimation in ubiquitous computing: A case study," in *Ubiquitous Computing*. Springer Berlin / Heidelberg, 2004, vol. 3205, pp. 88–106.

[14] Avalon interface specification. [Online]. Available: http://www.altera.com/literature/manual/mnl_avalon_spec.pdf

[15] Nios ii processor. [Online]. Available: http://www.altera.com/devices/processor/nios2/ni2-index.html