# Towards an Adaptive Hardware Parallel Particle Filter

David Pérez, Mónica Villaverde and Félix Moreno
*Centro de Electrónica Industrial (CEI), ETSII*
*Universidad Politécnica de Madrid (UPM)*
*Madrid, Spain*
{david.perez.daza, monica.villaverde, felix.moreno}@upm.es

## I. Introduction

A particle filter is a Montecarlo-based method suitable for predicting future states of non-linear systems with non-Gaussian noise. It is based on a set of samples of the state where each individual sample is called particle. These particles are weighted according to the real measure of the state in order to estimate the future state of the system. Particle filter is widely used in numerous applications ranging from prediction of failures [1] and prognosis [2] to object tracking [3]. Nevertheless, it is a very computationally expensive approach since it requires many resources to generate a huge number of particles, to evaluate them and to assign an appropriated weight to each one of them. However, its main bottleneck lies in the particle weights assignment since due to the sequential execution every particle has to wait for the others before the beginning of the resampling stage (Fig. 1).

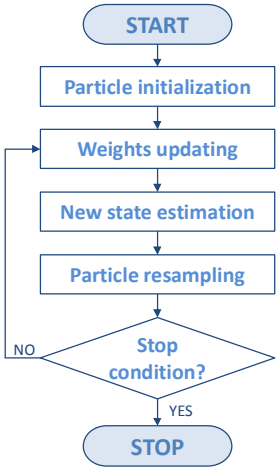


Fig. 1. Basic particle filter flowchart.

Furthermore, there are other decisive factors which have a huge influence over their implementation dependent on the kind of application. The most important ones are execution time, precision and consumption. In this work we propose a novel design which deals with all those issues making the particle filter adaptive for concrete application requirements.

The execution time is an important factor when the application demands a high processing speed to avoid malfunctions in real time. Particle filters are inherently sequential; consequently their execution time depends on the number of particles $N$, latency $L$ -which represents the time wasted to process the first particle- and clock frequency $f_{CLK}$ (1).

$$t_{execution} = (2 \cdot N + L - 1) \cdot \frac{1}{f_{CLK}} \quad (1)$$

We propose to distribute the total number of particles into $K$ different modules to reduce the execution time maintaining the same precision. All modules work in parallel splitting among them the total number of particles. Therefore, in this case, the execution time will be minimized as remarks the equation given by (2).

$$t_{execution} = \left(2 \cdot \frac{N}{K} + L - 1\right) \cdot \frac{1}{f_{CLK}} \quad (2)$$

There is a wide background concerned with how a particle filter can be parallelized through splitting the filter in different modules. In that case it is important to note that, for maintaining the Montecarlo philosophy, modules cannot be completely independent, so they must share some kind of information (i.e. particles). On the one hand, each module can evaluate a group of particles in order to generate a "big particle" which will be weighted. Therefore, each module has its own weight and the particle exchanging is carried out during the resampling stage [4]. Other alternative is shown in [5] whereas each module has a fixed number of particles and the same pairs of particles are always exchanged. On the other hand, a little more different option is presented in [6] since authors explain a specific resampling stage based on a Metropolis-Hastings algorithm.

## II. The Adaptive Hardware Parallel Particle Filter (AHP-PF)

Our proposal goes further since we propose an Adaptive Hardware Parallel Particle Filter (AHP-PF). An AHP-PF includes a hardware reconfigurable mechanism to adapt the parallelization to different application requirements during run-time. The need to implement the filter with real parallelism makes a hardware implementation better than a software one. Moreover, we have to assure that precision and consumption are suitable according to these application claims. Therefore the filter has to be able to adapt itself taking into account which is the most important factor for the application scope. This filter adaptation is achieved by adjusting its internal structure varying the number of required modules and their sizes. Thus, if we want to increase the execution speed and, at the same time maintain the accuracy of the algorithm, a module composed of N/2 particles should be substituted by two others composed of N/4 since when using more modules the execution time is reduced as demonstrated in (2). On the other hand if, at a given moment, precision is not the priority, the filter should remove some modules to simplify the algorithm execution and consequently consume less power (Fig. 2).

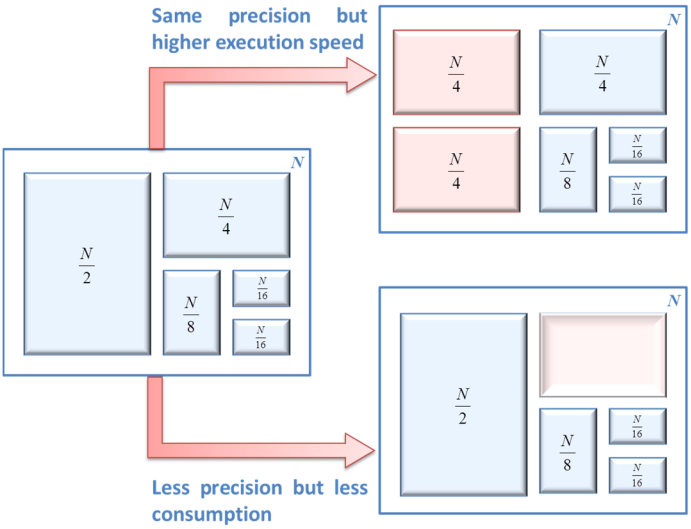


Fig. 2. Adaptive modular division of the particle filter.

A case study has to be defined in order to test the proposed method. Therefore, we have decided to focus our work on the object tracking field in order to incorporate our future system to autonomous driving application. However, this method could be applied to other fields related with signal prediction such as fault diagnosis or life cycle forecasting.

Our proposal for object tracking is to implement a HW/SW co-design mainly composed of a Raspberry Pi and a Xilinx FPGA (Fig. 3).

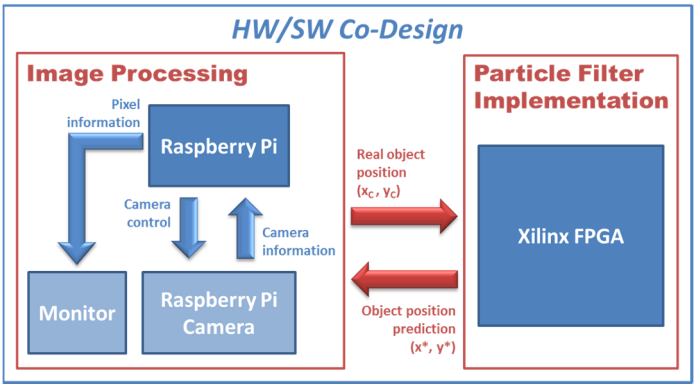


Fig. 3. System architecture.

The Raspberry Pi will be in charge of the image processing taking advance of its *Graphics Processor Unit* (GPU) included in its Broadcom BCM2835 *System-on-Chip* (SoC). A Raspicam camera is also employed to capture the images which will be analyzed -using OpenCV- in the GPU. Using the information provided by the camera the GPU has to be able to detect a specific object in real time. In this previous stage, the object will be defined by a given color and the tracking algorithm has to find that color on the received image in order to calculate the object centroid ($x_c$, $y_c$).

In contrast, the particle filter implementation will be carried out in a Virtex 5 FPGA to take advantage of its hardware resources. The Raspberry Pi *Central Processor Unit* (CPU) will send the ($x_c$, $y_c$) coordinates to the FPGA using the $I^2C$ protocol. That position will be processed in the FPGA applying the particle filter in order to estimate the object trajectory. The predicted trajectory (x*, y*) will be sent back from the FPGA to the Raspberry Pi since the object centroid and the predicted position will be displayed -overlapped with the image- on the monitor.

The filter adaptation is a complex task since it requires a thorough analysis in order to determine the additional information necessary to reconfigure the particle filter in consonance with the current situation. That reconfiguration implies the modification of the internal structure of the particle filter in order to adjust its number of modules and their sizes according to the demanded requirements. Those needs depend on the kind of the application. Perhaps, in our case it is not enough to focus only on the target object but also we aim to analyze the image in depth in order to extract additional information. For instance, if we can realize that there are many obstacles in the whole image, a more accurate prediction could be necessary; therefore we need to increase the number of particles and to add more modules. In contrast, if we are able to detect that the target is moving quickly, the real time requirements will be more critical so we have to split the particle filter into more modules to accelerate its execution.

## REFERENCES


[1] Orchard, M.E.; Vachtsevanos, G.J., "A Particle Filtering-based Framework for Real-Time Fault Diagnosis and Failure Prognosis in a Turbine Engine" *Mediterranean Conference on Control & Automation, 2007. MED '07*, pp.1,6, 27-29 June 2007.

[2] Saha, B.; Celaya, J.R.; Wysocki, P.F.; Goebel, K.F., "Towards Prognostics for Electronics Components" *Aerospace conference, 2009 IEEE*, pp.1,7, 7-14 March 2009.

[3] Yaowen Guan; Xiaoou Chen; Deshun Yang; Yuqian Wu, "Multi-person Tracking-by-Detection with Local Particle Filtering and Global Occlusion Handling," *2014 IEEE International Conference on Multimedia and Expo (ICME)*, pp.1,6, 14-18 July 2014.

[4] Bolic, Miodrag; Djuric, P.M.; Sangjin Hong, "Resampling Algorithms and Architectures for Distributed Particle Filters," *IEEE Transactions on Signal Processing*, vol.53, no.7, pp.2442,2450, July 2005.

[5] Yi Qiao Zhang; Sathyan, T.; Hedley, M.; Leong, P.H.W.; Pasha, A., "Hardware Efficient Parallel Particle Filter for Tracking in Wireless Networks" *2012 IEEE 23rd International Symposium on Personal Indoor and Mobile Radio Communications* (PIMRC), pp.1734,1739, 9-12 Sept. 2012.

[6] Lifeng Miao; Zhang, J.J.; Chakrabarti, C.; Papandreou-Suppappola, A., "A new parallel implementation for particle filters and its application to adaptive waveform design," *2010 IEEE Workshop on Signal Processing Systems (SIPS)*, pp.19,24, 6-8 Oct. 2010.