The 2010 IEEE/RSJ International Conference on
Intelligent Robots and Systems
October 18-22, 2010, Taipei, Taiwan

Multiple-Person Tracking devoted to distributed multi Smart Camera Networks

Iker Zuriarrain[†], Jose Ignacio Aizpurua[†], Frederic Lerasle[‡¶] and Nestor Arana[†]

[†] Signal Treatment and Communications Research Group, University of Mondragon, 20500 Mondragon, Spain

[‡] CNRS; LAAS; 7 avenue du Colonel Roche, 31077 Toulouse Cedex, France

[¶] Université de Toulouse; UPS, INSA, INP, ISAE; LAAS; LAAS : F-31077 Toulouse, France

E-mail: {izuriarrain, narana}@eps.mondragon.edu, jose.aizpurua@alumni.eps.mondragon.edu, lerasle@laas.fr

***Abstract*— Camera networks are an important component of modern complex systems, be it for surveillance, human/machine interaction or healthcare. Having smart cameras that can, by themselves, perform part of the data processing improves scalability both in processing and network resources. In this paper, we present the** HYBRID **algorithm for multiple person tracking intended for implementation on a smart camera platform, along with the development methodology to implement said algorithm in an FPGA-based smart camera. The** HYBRID **strategy outperforms the well-known Markov Chain Monte Carlo based particle filter (MCMC-PF) in terms of (i) parallelization capabilities as the MCMC-PF sequentially processes the particles, and (ii) tracking performances (*i.e.*, robustness and precision).**

I. Introduction

People detection and tracking in human-centered environments is one of the most important and fundamental technologies to develop distributed surveillance, autonomous mobile robotics, or intelligent transports, even cooperative distributed sensor (both static and mobile sensors) systems. Each of these applications utilizes camera networks and therefore share many of the same technical challenges. Video processing for such systems is suitable for the information richness it encompasses. Depending on the surveillance task, the sensor type, and the number of sensors in the network, the processing of the raw data can be quite a daunting task. Distributing the processing of the meta level data to each of the sensing nodes in the network can dramatically reduce the bandwidth burden, allowing for larger and more capable networks. These independently operating nodes within a sensing network which provide a certain pre-processing of video stream are often called smart cameras.

While a number of advances have been made in the area of camera networks, many of them seem to be geared towards having a central processing unit that takes the raw (or, at most, low-level processed) data from the cameras, in order to provide tracking across multiple cameras [1], [2], [3]. To the best of our knowledge, there are no intelligent cameras dedicated to multiple human tracking (as opposed to human detection) currently available off-the-shelf, although some efforts in that direction have been made in the last few years, such as FPGA-based single target trackers [4], [5] or DSP-based trackers [6].

Particle filters have also been implemented in embedded systems. For instance, Hoffman *et al.* [7] implemented a mixture of a particle filter with Active Appearance Models in order to track single targets. While the appearance model used by Hoffman is more complex than ours, it is still limited to a single target, and the implementation does not take advantage of the parallelism inherent in the particle filtering algorithm.

Also in this vein, Cho *et al.* [8] implement a grayscale particle filter in an FPGA. The main differences with our work are three: first, the algorithm is the well-known and trivial CONDENSATION for single target tracking using gray-scale features, while we use a more complex multi-target MCMC/PF algorithm and color cues; also, in our case, the FPGA module is integrated in the camera, while Cho *et al.* keep it separate, which makes the communication with the camera more complex; third, no mention is made of which methodology they followed in their work. In our case, the methodology is explicitly explained in Section III-B.

Meanwhile, visual multiple person tracker (MPT in short) has received tremendous attention in the Vision literature. Monte-Carlo methods, especially particle filters (PF), are well suited for parallelization and have proved to be a powerful formalism to track varying numbers of people. Yet, joint PF suffers from exponential complexity in the number of targets and this is due to the inefficiency of importance sampling which classically draws the particles from the system dynamics and so ”blindly” *w.r.t* the measurements. A remedy addressed in [9], [10] is to replace the traditional importance step by a Markov Chain Monte Carlo (MCMC in short) sampling step within the PF. An unweighted sample swarm is obtained by storing the samples after the initial burn-in iterations in the Markov chain. Yet, the drawback of this filtering strategy remains its non parallelization on clusters as the MCMC-based PF remains sequential like pure MCMC strategies.

Our MPT differs from the aforementioned approaches as follows. First, we propose an efficient proposal distribution based on saliency maps which combine several information sources like system dynamics, SVM-based person detection [11], and adaptive background mixture models [12]. Saliency maps combined with a rejection sampling mechanism, as far as we are aware, have never been used for particle sampling within Monte Carlo methods. The motivation is to draw the particles in the relevant areas of the high dimensional state-space and so limits the notoriously burst in terms of particles and MCMC iterations. Second, we design a novel MCMC-based PF which remains well suited for parallelization.

The paper is organized as follows. Section II depicts our MCMC-based PF, compares its performance with the state-of-the art, and highlights its parallelization capabilities. Section III describes the ongoing integration on our smart camera with a focus on a motion detector based on background mixture modeling. Finally, section IV summarizes our contributions and discusses future works.

II. Multitarget Tracking

As mentioned, the camera is intended to perform tracking of multiple interacting targets. As such, an algorithm has been developed that can a) track a varying number of interacting targets along a sequence of images and b) can be implemented in a smart camera architecture capable of parallel processing (as explained in Section III).

The aim of the algorithm is to fit the template relative to each target all along the video stream through the estimation of its status $r = \{New, Tracked, Lost, Dead\}$[1], its image coordinates (u, v) and its scale factor s. These continuous parameters are accounted for in the state vector $\mathbf{x}_k$ related to the k-th frame. With regard to the system dynamics, the unpredictable motion of humans leads to define the state vector $\mathbf{x}_k = (r_k, u_k, v_k, s_k)'$ and to assume that its entries evolve according to mutually independent random walk, *viz.* $p(\mathbf{x}_k|\mathbf{x}_{k-1}) = \mathcal{N}(\mathbf{x}_k; \mathbf{x}_{k-1}, \Sigma)$ where $\mathcal{N}(.; \mu, \Sigma)$ is a Gaussian distribution with mean μ and covariance $\Sigma = \mathrm{diag}(\sigma_u^2, \sigma_v^2, \sigma_s^2)$. Finally, the global state vector is defined by $\mathbf{X}_k = (\mathbf{x}_k^1, \ldots, \mathbf{x}_k^{N_t})'$ where N_t is the number of targets.

Each target might be in one of four states: $Dead$, in which case the target is eliminated from the state vector; New, in which the target is newly appeared in the current image; $Tracked$, in which the target has been tracked for at least one previous image, and therefore, it is already known; and $Lost$, in which the target has disappeared in the current frame. Logically, New targets turn into $Tracked$ targets at the end of the processing of the current image, and $Lost$ targets likewise turn into $Dead$ targets (and as such, are removed from the state vector).

Our filtering scheme, called HYBRID for the next and detailed in Table I, combines Markov chain iterations and principles of the CONDENSATION algorithm [13]. Multiple saliency maps $\{S_k^d\}_{d=1,\ldots,N_d}$ are processed from visual detector outputs $Det^d(z_k)$ and from the targets' dynamics $p(\mathbf{x}_k^j|\mathbf{x}_{k-1}^j)$. These maps, independent and so computed in parallel on multi-core processing units, are then merged in a unified saliency map (step #5) which highlights "relevant" areas of the state space. The underlying data driven proposal densities and particle sampling mechanisms will (re)-concentrate the particles on the right regions of interest. The calculation of the saliency maps is independent of both the number of targets and number of particles, depending only on the detector's runtime and the number of detectors.

In step #7, the continuous parameters $(\mathbf{x}_k^{i,1}, \ldots, \mathbf{x}_k^{i,N_t})$ of the $i-th$ particle are randomly drawn from an uniform distribution, and the values for these positions in the saliency map lead to a likelihood $\sum_{j=1}^{N_t} S_k(u_k^j,\ v_k^j)$ which allow to accept/reject the sample based on rejection sampling mechanism. This involves simply looking up the value in the saliency map S_k, and permits: (1) the reduction of the computation cost, (2) an efficient sampling in the high likelihood areas of the continuous state-subspace and so a drastic limitation of the particle burst, (3) an easier implementation of the proposal density where this is difficult to model analytically[2]. An example of particle swarm drawn by this mechanism is shown in Figure 1. As can be seen, the particles naturally concentrate in areas where the saliency map shows there is a high probability of target appearance.

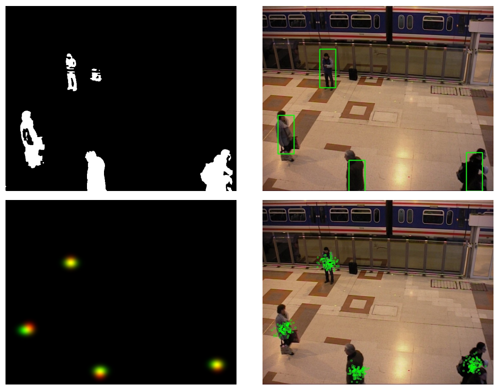
Fig. 1. Detector outputs (MGM and HOG-SVM), associated saliency map and particle sampling.

This first sampling ignores possible target jumps *i.e.* we assume the tracked targets remain the same from one frame to the next. In step #12, a second sampling step through MCMC moves leads each individual particle from a current targets' configuration to another one (based on likely jumps) which is more representative of the current image contents. Drawing new samples by moving jointly all the dimensions suffers from exponential complexity with the state-space dimensionality [14]. The popular issue is to propose target-wise marginal moves *i.e.* only moving one target at each iteration. The transition from state hypothesis $\mathbf{X}_k$ to the proposed next $\mathbf{X'}_k$ is conditioned by a proposal density $q(\mathbf{x'}_k|\mathbf{x}_k, z_k)$ for each jump in the targets' status.

Traditionally, an MCMC process requires a high number of burn-in iterations as the Markov chain must usually, given an initial state, move between trans-dimensional state-spaces. The iteration number N_{mcmc}, which role is to increase diversity in the particle swarm, is here reduced drastically as: (1) we only have to deal with targets' configuration as the continuous component subspace have been already sampled, (2) the associated target number does not change significantly from one frame to the next, (3) we consider a Markov chain moves for each individual particle and the resulting particle swarm induces diversity itself. A last observation concerns the low computation cost of each iteration since it changes the computation of the acceptance ratio $\mathcal{R}_a$ into a lookup into

[1]New and Dead status mean the target entering/leaving the field-of-view.

[2]Even if only Gaussian mixtures are considered in this work.

```
$[\{\mathbf{X}_k^i, w_k^i\}]_{i=1}^N$= HYBRID($\mathbf{X}_{k-1}, z_k$)
1: IF $k = 0$, THEN Set status $r_{k-1}^t = Dead$ for each target $t$ END IF    a
2: IF $k \geq 1$ THEN   {}
3:   Generate saliency maps $\{S_k^d(\mathbf{x}|z_k)\}_{d=1,\ldots,N_d}$ for each visual detector $Det^d(z_k)$
4:   Generate a saliency map from the targets' dynamics $S_k^{dyn}(\mathbf{x}) = \sum_{j=1}^{T} p(\mathbf{x}^j|\mathbf{x}_{k-1}^j)$
5:   Generate the unified saliency map $S_k(\mathbf{x}|z_k) = S_k^{dyn}(\mathbf{x}) + \sum_{d=1}^{D} S_k^d(\mathbf{x}|z_k)$
6:   FOR $i = 1, \ldots, N_p$, DO
7:     FOR $j = 1, \ldots, N_t$, DO
8:       IF $r_k^j == Tracked$ THEN
9:         Sample $\mathbf{x}_k^{i,j}$ from $S_k$ via rejection sampling
10:       END IF
11:     END FOR
12:     FOR $m = 0, \ldots, N_{mcmc}$, DO
13:       Randomly pick a target $j$ and propose an associated event/jump $\tau$ (events have all the same probability of happening)
14:       Apply $\tau$ to $\mathbf{X}_k^i$ to generate $\mathbf{X'}_k^i$ with update of the j-th target's status.
15:       Draw threshold $\beta$ according to a uniform distribution over $(0, 1]$
16:       Calculate the acceptance ratio $\mathcal{R}_a = \frac{p_1(z_k|\mathbf{x'}_k^i).q(\mathbf{x}_k^i|\mathbf{x'}_k^i,\ z_k).\Psi(\mathbf{x}_k^j,\mathbf{x}_k^m)}{p_1(z_k|\mathbf{x}_k^i).q(\mathbf{x}_k^i|\mathbf{x'}_k^i,\ z_k).\Psi(\mathbf{x'}_k^j,\mathbf{x'}_k^m)}$
17:       IF $\mathcal{R}_a > \beta$ THEN
18:         $\mathbf{x}_k^i = \mathbf{x'}_k{}^i$
19:       END IF
20:     END FOR
21:     Update the weight $w_k^i$ associated to $\mathbf{X}_k^i$ according to $w_k^i \propto p_2(z_k|\mathbf{X}_k^i)$, prior to a normalization step so that $\sum_i w_k^i = 1$
22:   END FOR
23:   Compute the MAP estimator $E_{p(\mathbf{X}_k|z_{1:k})}[\mathbf{X}_k] = argmax_{\mathbf{X}_k^i}[w_k^i]$ to approximate the posterior $p(\mathbf{x}_k|z_{1:k})$
24:   Remove all targets $j$ where $\{r_k^j\}_{j=1,\ldots,N_t} == Lost$, and set $\{r_k^j\} = Tracked$ for all targets where $r_t^j == New$ in $E_{p(\mathbf{X}_k|z_{1:k})}$
25: END IF
```

TABLE I

OUR FILTERING STRATEGY (HYBRID).

the saliency map S_k, instead of a more expensive operation *e.g.* the comparison of color distributions.

Step #21 corresponds to the particle weighting update. To this end, we affect a particle $\mathbf{X}_k^i$ a weight w_k^i involving its likelihood $p_2(z_k|\mathbf{X}_k^i)$. This likelihood involves three different calculations, depending on the target status. If a target is $Dead$ or $Lost$, it is calculated as a similarity measure to the background (*i.e.*, likelihood the target *is not* present); if the target is New, it is a dissimilarity measure to the background (*i.e.*, likelihood the target *is* present); finally, if the target is $Tracked$, it is calculated as both dissimilarity to the background, and similarity to an adaptive per-target appearance model (*i.e.*, likelihood the target *is* present *and* is similar in appearance to what we have seen in earlier frames).

Finally, the particle with the highest weight w_k^i is chosen as the most probable configuration of the system, and is used as input to the dynamic model for the next frame.

The results of running this algorithm on a 3400 image sequence taken from the public PETS 2006 dataset (with manually tagged ground truth, as the ground truth included with the dataset is not geared towards people tracking) are shown in Table II, along with those from a well known mixed MCMC-PF tracker[10][9]. The statistics shown are the False Positive Rate (FPR), which measures the number of targets tracked that do not really exist; the Tracking Success Rate (TSR), which measures correct tracking percentage; and the Precision Error (PE) which measures the deviation in pixels from the target position in the ground truth. Each of these statistics are averaged over four runs of the algorithm, and the standard deviation for each stat is shown in parentheses. Some snapshots of this sequence are also shown in Figure 2.

For this test, the HYBRID tracker was run with 150 particles, while the MCMC-PF tracker ran with an equivalent number of iterations (*i.e.*, since MCMC-PF only switches a target per iteration, while a HYBRID particle may modify any of its targets, MCMC-PF ran for 1500 iterations).

As we can see from the results, HYBRID maintains and even noticeably improves performance when compared to MCMCPF, with the added advantage that a number of the operations may be performed in parallel (for example, sampling of a particle can proceed in parallel with the evaluation of a different particle, given that both utilize different data: saliency map for sampling vs. image data for evaluation), permitting predicted gains of between 20% to 30% in processing speed due to that single optimization.

TABLE II

RESULTS FOR THE *PETS* 2006 SEQUENCE.

Strategy	FPR (σ)	TSR (σ)	PE (σ)
MCMCPF	0.026 ($8.3.10^{-4}$)	72.9% ($1.79.10^{-2}$)	10.17 (0.965)
HYBRID	0.023 ($2.2.10^{-3}$)	77.9% ($5.316.10^{-3}$)	7.62 (1.136)

III. CAMERA INTEGRATION

The HYBRID algorithm was intended for implementation in an FPGA-based smart camera, in order to take advantage of the parallelization posibilities offered by the architecture. In this section, we will detail the camera platform we work with, as well as the methodology we have followed for the simulation and partial implementation of the algorithm.

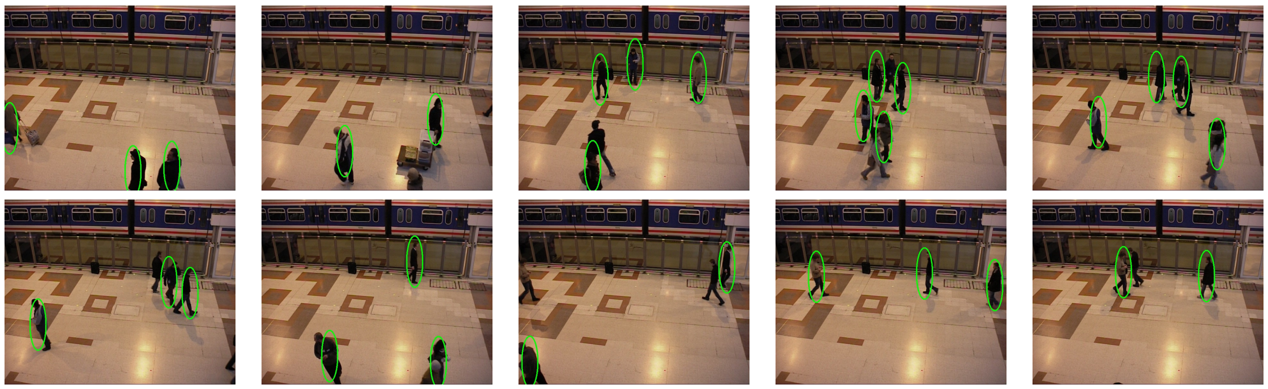

Fig. 2. Sample results from PETS dataset for the HYBRID algorithm

A. The smart camera platform

The camera we have been working with is an FPGA based smart camera (Figure 3) by Delta Technologies Sud Ouest (DTSO), a company from Toulouse, France.

Fig. 3. Our smart camera platform.

The DTSO iCam camera is currently fitted with two FPGA modules. Each of these modules is based on an Altera Cyclone-II FPGA, and includes a 18 megabit memory module (with a 18 bit word-width), as well as the communications with the neighboring modules. Currently, all programming of the FPGA modules must be done using the JTAG connector while the camera is out of its enclosure, so there is no means for online reconfiguration.

Communications between the camera and other external devices is currently done using wither an Ethernet communications module or a Wi-Fi module, since the camera is intended not as a stand-alone product, but as a node of a network of cameras. This also allows for some of the processing to be done in a more conventional processor, since the communications module includes a Freescale processor.

B. Methodology

Implementing any non-trivial algorithm in hardware is in itself a non-trivial problem that requires both a good knowledge of low level design and programming techniques, and a very intimate knowledge of the target algorithm. Even then, there are a number of design decision that might have unforeseen effects on the final implementation.

In order to lessen the effect of these factors, an accurate model of the system is often an invaluable tool, since it allows exploration of design decisions with a minimum investment of time and effort. That way, it becomes possible to test proposed changes to a hardware architecture in but a fraction of the time and cost. Also, in cases where the system is a mix of hardware and software (as many complex systems are), partitioning becomes both easier and more efficient in cases where an accurate working model of the system exists[15].

There are a number of ways these models can be developed. For example, the Hardware Resource Model promoted by the Object Management Group [16] offers a framework in which developers may describe a model of the hardware for a given system. This model is part of a broader framework named MARTE (Modeling and Analysis of Real-Time and Embedded systems) for the development of real-time systems. On the other hand, there are a number of languages such as HandelC or SystemC oriented towards programming the model in higher level languages (often derived from the C family of languages).

SystemC, along with a methodology based on Transaction Level Modeling (TLM), has been used in a number of works[15], [17]. This methodology is based on the classic Design-Implement-Redesign spiral model which is common in software development, and takes advantage of the wide range of simulation capabilities offered by SystemC: starting from a functional model (i.e., a C++ implementation of the algorithm), it is possible to iteratively add detail to different components of the system, creating more detailed models that approach the low-level RTL code necessary for actual implementation without losing confidence in the functionality of the code. Table III shows a common classification of the different detail levels involved in TLM. As the model moves downwards in that table, it acquires an increasing amount of detail, until it has reached the RTL level, where it is possible to directly synthetize it with the correct tools.

In our case, the hardware architecture obtained from iterating on the algorithm shown on Section II is divided on three main blocks: first, detection and saliency map (S_k) generation, which takes the input image, runs it through a number of detectors (in this case, a Multiple Gaussian Mixture background subtraction module and a Histogram of Oriented Gradients-based SVM classifier) and along with the data from the dynamic model generates the saliency map;

TABLE III

DETAIL LEVELS OF TRANSACTION-LEVEL MODELS[18].

Model	Communication	Functionality
System Assembly	Untimed	Untimed
Component Assembly	Untimed	Approximated
Bus Arbitration	Approximated	Approximated
Bus Functional	Cycle Accurate	Approximated
Cycle Accurate Computation	Approximated	Cycle Accurate
RTL	Cycle Accurate	Cycle Accurate

second, a particle processor, which is a self-contained unit that processes the different hypotheses that drive the particle filter, using the saliency map and the image data; and finally, the selection of the most probable particle. This architecture is shown in Figure 4.

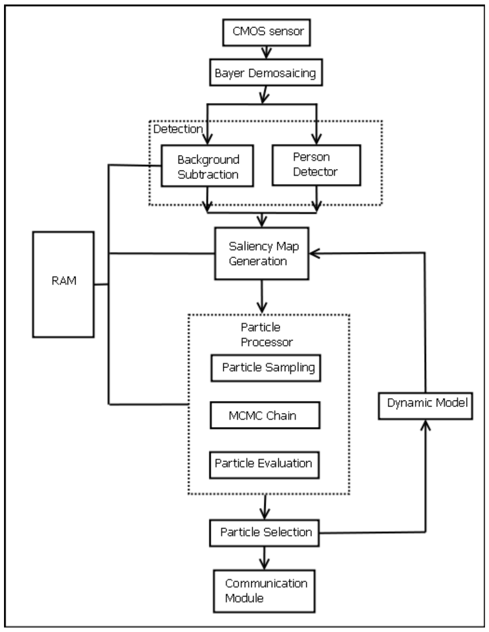

Fig. 4. Architecture of the tracking algorithm hardware after refinement with the simulation.

There are a few things worth noting in this model: it is clear that each detector can, along with the dynamic model, be run in parallel with the others up until the choke point that is the saliency map generation; a similar thing happens with the particle processor, which can, since each particle is independent to the others, be run multiple times in parallel for different particles, greatly increasing the speed of the whole system.

C. Simulation Results

According to the simulation of the previous model, the algorithm has a maximum runtime of 55 ms per image for up to 10 targets (approximation made by assuming the most pessimistic case for all processes with a variable number of iterations), which would allow it to run at slightly better than 18 fps. The breakdown of these 55 ms per image can be seen in Table IV.

TABLE IV

WORST-CASE TIMING RESULTS FOR THE TRACKING SYSTEM (MAX 10 TARGETS).

Component	Time (ms)
Detection and Saliency map	26.864
Particle sampling and evaluation	28.687
Particle selection	0.003
Total	55.554

Given that the detectors are heterogeneous, running them in parallel reduces the runtime to that of the worst-performing detector of the set, which is the HOG-SVM classifier. However, as mentioned earlier, multiple particles can be processed at the same time. Using four particle processors, for example, cuts runtime by almost 40% down to 34.039 ms, increasing performance to 29 fps, which is enough for real-time video.

D. Result validation

In order to establish the validity and accuracy of the simulation results, a component of the algorithm has been taken to the RTL-level model and implemented in VHDL. The component chosen was the MGM background subtraction module, which we believe has enough complexity regarding functions such as memory usage and parallelizability so as to be a representative sample of the algorithm, while still being simple enough that the implementation effort would not be overly demanding.

The MGM background subtraction [12] compares each pixel of the image with a multiple Gaussian model that indicates a range of colors (modelled as Gaussian distributions) that have most often appeared at that location. If the pixel is close enough to the colors predicted by the model, then it is recognized as a background pixel. The use of multiple Gaussians (as opposed to a single one) allows the model to recognize periodic variations of colouring in an area (such as a rotating fan) as forming part of the background, reducing the amount of false positives.

The pixels labeled as foreground by the model are then relabeled by a classic two-pass connected components algorithm, so that those blobs can then be used by the saliency map generation to pinpoint the areas of the image that have a high probability of containing targets to track.

The detailed hardware architecture model for this component can be seen in Figure 5. It is divided in three areas, corresponding to reception of images (left), background subtraction (middle) and connected components clustering (right). The input of the module is the image pixels from the debayering module, and the output is an image with labeled regions corresponding to each foreground blob of sufficient size in the image.

The timing results for both the SystemC simulation and the VHDL implementation are shown in Table V, showing that the result of the simulation seems to be relatively accurate, if slightly pessimistic. This is mostly due to a particular case in the SDRAM's behaviour that was not properly modelled when the SystemC model was created: the SDRAM allows

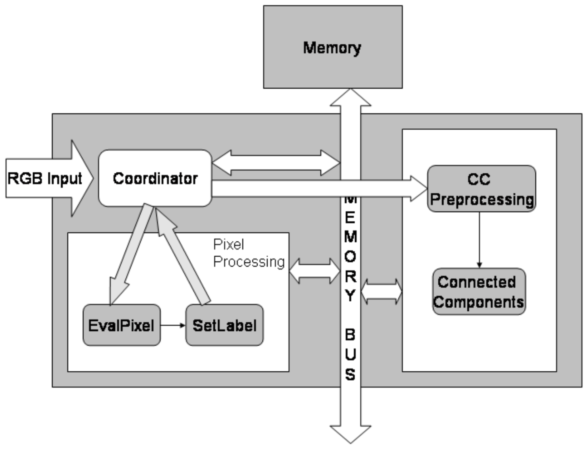

Fig. 5. Detailed HW architecture model for the Background Subtraction component.

for faster access for consecutive items in memory, while the SystemC model used an average of the access times. Since the data is structured in a way so as to take advantage of this speed-up, the VHDL implementation is slightly faster.

Assuming that the code we have chosen is indeed representative of the complexity of the problem, these results suggest that the simulation data we obtained from the SystemC model is accurate enough for practical purposes such as selecting the definitive arquitecture of the system and establishing minimum requirements.

TABLE V
TIMING RESULTS FOR THE BACKGROUND SUBTRACTION.

	Timing (ms per image)
HW model in SystemC	27.312
VHDL implementation	26.864

IV. CONCLUSION AND FUTURE WORKS

In this paper we have shown the current state of our research in two fronts: first, an algorithm for multiple target tracking intended for implementation in a FPGA-based smart camera; second, the methodology of implementation of said algorithm via SystemC and TLM. According to our predictions, the algorithm should run at 18 fps in the camera when unoptimized, and upwards from 29 fps (i.e., real time video) with some basic optimizations.

We have also validated the methodology and the predictions made by the simulation by the implementation of a selected part of the algorithm, giving the smart camera limited functionality.

Current research concerns the implementation of the full functionality of the camera by implementing the remainder of the algorithm, and the use of multiple communicating smart camera nodes in order to reliably track people across large scale human-centered environments.The major challenge is to avoid sending many full-resolution, real-time images to the video processor of the network PCs, by offloading the processing power into the cameras themselves.

REFERENCES

[1] R. Goshorn, J. Goshorn, D. Goshorn, and H. Aghajan, “Architecture for cluster-based automated surveillance network for detecting and tracking multiple persons,” in *1st Int. Conf. on Distributed Smart Cameras (ICDSC)*, 2007.

[2] C. Arth, C. Leistner, and H. Bischof, “Object reacquisition and tracking in large-scale smart camera networks,” in *Proceedings of the First ACM/IEEE International Conference on Distributed Smart Cameras, Vienna*, pp. 156–163, 2007.

[3] M. Hoffmann, M. Wittke, Y. Bernard, R. Soleymani, and J. Hahner, “DMCtrac: Distributed multi camera tracking,” in *Distributed Smart Cameras, 2008. ICDSC 2008. Second ACM/IEEE International Conference on*, pp. 1–10, 2008.

[4] F. Dias, F. Berry, J. Serot, and F. Marmoiton, “Hardware, Design and Implementation issues on a FPGA-based Smart Camera,” in *Distributed Smart Cameras, 2007. ICDSC’07. First ACM/IEEE International Conference on*, pp. 20–26, 2007.

[5] S. Hong, X. Liang, and P. Djuric, “Reconfigurable particle filter design using dataflow structure translation,” in *2004 IEEE Workshop on Signal Processing Systems Design and Implementation*, pp. 325–30, 2004.

[6] M. Litzenberger, A. Belbachir, P. Schon, and C. Posch, “Embedded smart camera for high speed vision,” in *Distributed Smart Cameras, 2007. ICDSC’07. First ACM/IEEE International Conference on*, pp. 81–86, 2007.

[7] M. Hoffmann, A. Swart, K. Hunter, B. Herbst, S. Flecky, and W. Strasser, “Model-based robust and precise tracking embedded in smart cameras: the PFAAM-CAM,” in *Distributed Smart Cameras, 2008. ICDSC 2008. Second ACM/IEEE International Conference on*, pp. 1–8, 2008.

[8] J. Cho, S. Hun Jin, X. Dai Pham, and J. Jeon, “Multiple object tracking circuit using particle filters with multiple features,” in *Int. Conf. on Robotics and Automation (ICRA’07)*, (Roma, Italy), pp. 4639–4644, April 2007.

[9] F. Bardet, T. Chateau, and D. Ramasadan, “Illumination aware MCMC particle filter for long-term outdoor multi-object simultaneous tracking and classification,” in *Int. Conf. on Computer Vision (ICCV’09)*, (Kyoto, Japan), September 2009.

[10] Z. Khan, T. Balch, and F. Dellaert, “MCMC-based particle filtering for tracking a variable number of interacting targets,” *Trans. on Pattern Analysis Machine Intelligence (PAMI’05)*, vol. 27, no. 11, pp. 1805–1818, 2005.

[11] N. Dalal and B. Triggs, “Histograms of oriented gradients for human detection,” in *Int. Conf. on Computer Vision and Pattern Recognition (CVPR’05)*, (San Diego, USA), June 2005.

[12] C. Stauffer and W. Grimson, “Adaptative background mixture models for real-time tracking,” in *Int. Conf. on Computer Vision and Pattern Recognition (CVPR’99)*, vol. 2, (Fort Collins, USA), pp. 22–46, June 1999.

[13] M. Isard and A. Blake, “CONDENSATION – conditional density propagation for visual tracking,” *Int. Journal on Computer Vision*, vol. 29, no. 1, pp. 5–28, 1998.

[14] K. Smith, D. Gatica-Perez, and J. Odobez, “Using particles to track varying numbers of interacting people,” in *Int. Conf. on Computer Vision and Pattern Recognition (CVPR’05)*, (San Diego, USA), pp. 962–969, June 2005.

[15] J. Lee and S. Park, “Transaction Level Modeling for Hardware Architecture Exploration with IEEE 802.11 n Receiver Example,” in *Communication Technology, 2006. ICCT’06. International Conference on*, pp. 1–4, 2006.

[16] S. Taha, A. Radermacher, S. Gerard, and J. Dekeyser, “An open framework for detailed hardware modeling,” in *Industrial Embedded Systems, 2007. SIES’07. International Symposium on*, pp. 118–125, 2007.

[17] C. Helmstetter and V. Joloboff, “SimSoC: A SystemC TLM integrated ISS for full system simulation,” in *IEEE Asia Pacific Conference on Circuits and Systems, 2008. APCCAS 2008*, pp. 1759–1762, 2008.

[18] D. Black and J. Donovan, *SystemC: from the ground up*. Springer-Verlag New York Inc, 2005.