Implementation and Optimization of Particle Filter Tracking Algorithm on Multi-DSPs System

Gongyan Li, Bin Li,
Research Center of Integrated Information System
Institute of Automation, CAS
Beijing 100190, China
gongyan.li@ia.ac.cn
lixiebin2003@yahoo.com.cn

Zhou Liu, Xiaopeng Chen
National Key Lab of Pattern Recognition
Institute of Automation, CAS
Beijing 100190, China
zliu@nlpr.ia.ac.cn

***Abstract*: Particle Filter is a filter method based on Monte Carlo and recursive Bayesian estimation, which has special advantage in dealing with the state and parameter estimation in the nonlinear and non-Gaussian system. However, high computational complexity and lack of dedicated embedded DSP and ARM hardware for real-time processing have adversely affected its application in real life. In this paper, we present an embedded hardware architecture based on Multi-DSPs (TMS320DM642) for speeding up the basic computational performance, thereby, making Particle Filtering based solutions amenable to real-time constraints. Simultaneously, on this embedded DSP system, we also do some improvement to the Particle Filter algorithm for realization. First, the number of particles is reduced by fusing Mean-shift algorithm after resampling step. Then, RSR (Residual Systematic Resampling) method is mended to reduce the time-consuming division computation and to retain the number of particles same pre-and-post resampling procedure. The performance of the proposed embedded DSP system and optimized algorithm are evaluated qualitatively on real-world video sequences with moving target.**

Keywords: Particle Filter, Mean-shift, Object tracking, TMS320DM642

I. INTRODUCTION

Particle Filter is a filter method based on Monte Carlo and recursive Bayesian Sequential Estimation, which uses a weighted set of particles to approximate the posterior distribution. Recently, there are a lot of its applications in vision objects tracking, as it can deal with the state estimation in the nonlinear and non-Gaussian system. Particle Filter shows higher performance in tracking compared with other methods, such as Kalman Filter, EKF, UKF and Mean-shift, when dealing with cluttered environments and occlusive situations [1] [2]. However, high computational complexity and lack of dedicated embedded DSP and ARM hardware for real-time processing have adversely affected its application in real resource-constrained environment [3~5]. So far, there are two possible ways to solve the problem of computational complexity: one is to design a wholly parallel hardware system which needs high performance FGPA components, such as Xinlinx's Virtex-4 SX [3], an alternative is to design an embedded system with multiply DSP processors; the other is to optimize and simplify the algorithm itself, such as, reducing the number of particles and division operations, transferring the floating-point computation to fixed-point, etc.

In this paper, we present an embedded hardware architecture based on Multi-DSPs (TMS320DM642) for speeding up the basic computation, thereby, making Particle Filtering based solutions amenable to real-time constraints in embedded DSP system. On this embedded hardware, we also do some improvement to the Particle Filter algorithm for realization: first, by Mean-shift clustering method after resampling, every particle can approach the real target position much more, and thereby, less number of particles is needed to estimate the target status. Second, RSR (Residual Systematic Resampling) method is mended to reduce the computational complexity and to make the number of particles same pre-and-post Resampling procedure. The performance of the proposed hardware system and optimized algorithm are evaluated qualitatively on real-world video sequences with moving target.

The rest paper is organized as follows. In Section 2, we will give a short introduction to Particle Filter, and then derive the state transition model and observation model in our method. In Section 3, some variations on Particle Filter are given for reducing computational complexity. Section 4 shows the implementation of our method on Multi-DSPs hardware. Evaluation of our hardware system and optimized method is presented in Section 5. Section 6 concludes the paper.

II. PARTICLE FILTER THEORY

The algorithm flow of Particle Filter for a dynamic system is usually as follows [1] [2].

1) Initialization. At time $k = 0$, sample system states $\{\tilde{x}_0^i\}_{i=1}^N \propto p(x_0)$, and the weights of all particles is set to $1/N$, where N is the number of particles.

2) Particle sampling and weight computation. Based on the observation z_k at time k and the state of the system at time $k-1$ (represented by particles $\{ x_{k-1}^i, \omega_{k-1}^i \}$, $i=1,...,N$,),

propose a set of new particles $\{ x_k^i, i=1,...,N, \}$ from the importance density function $q(x_k|\tilde{x}_{0:k-1}^i, z_{1:k})$. Then, with each proposed particle x_k^i, associate an unnormalized weight defined as:

$$w_k^i = w_{k-1}^i \frac{p(z_k|x_k^i)\, p(x_k^i|\tilde{x}_{k-1}^i)}{q(x_k^i|\tilde{x}_{0:k-1}^i, z_{1:k})} \tag{1}$$

At last, normalize weights according to:

$$\tilde{w}_k^i = w_k^i \Big/ \sum_{i=1}^{N} w_k^i \tag{2}$$

3) Compute the Expected Inference. The posterior distribution can be approximated by the following equation:

$$p(x_k|z_{1:k}) \approx \hat{p}(x_k|z_{1:k}) = \sum_{i=1}^{N} \tilde{w}_k^i \delta(x_k - x_k^i) \tag{3}$$

Therefore, an expected inference can be computed as:

$$E(g(x_{0:k})) \approx \sum_{i=1}^{N} \tilde{w}_k^i g(x_{0:k}^i) \tag{4}$$

4) Resampling. Sampling a new set of particles from the set $\{x_{0:k}^i\}_{i=1}^N$ according to weight $\tilde{w}_k^i$. The resulting particles are represented by $\{\tilde{x}_{0:k}^i\}_{i=1}^N$ and they are assigned identical weights: $\tilde{w}_k^i = w_k^i = 1/N$;

5) $k=k+1$, and go Step 2) for a next iteration.

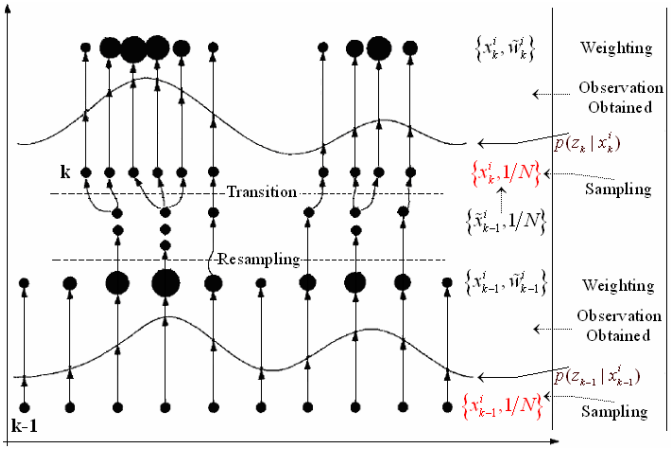


Figure 1. Flowchart of the Particle Filter.

From the algorithm flow described above, we need to make certain the state transition model $p(x_k^i|\tilde{x}_{k-1}^i)$ and the observation model $p(z_k|x_k^i)$ to describe a dynamic system. In this paper, for vision tracking, we use $X_k^i = [x_k^i, y_k^i]^T$ which represents the coordinates of moving object to denote the state vector of the dynamic system. Therefore, the particles are represented by $\{X_{0:k}^i, w_k^i\}_{i=1}^N$. We construct the state transition model as follows [6]:

$$X_k^i = X_{k-1}^i + \frac{1}{m} \sum_{n=k-m}^{k} \left| X_{n-1}^i - X_{n-2}^i \right| + \lambda v_k \tag{5}$$

Where $v_k : U[-1,1]$ and $\lambda = [\lambda_1, \lambda_2]^T$。

The construction of observation model is similar to the method in [7]. In this paper, we construct the color model of moving object at time k in the YCbCr color space. For consideration of computation, only Y component is used. We assume that, at time k, the moving object under tracking whose coordinates are y has n_h candidate points and the histogram is composed of m bins. Then, the corresponding histogram of a moving object at current frame is computed as:

$$\hat{p}_u(y) = C_h \sum_{i=1}^{n_h} k\left(\left\| \frac{y - x_i}{h} \right\|^2 \right) \delta\left[b(x_i) - u \right], \quad u = 1, 2, ...m \tag{6}$$

Where C_h is a normalization constant, h is window radius, $k(x) = \begin{cases} 1-x, x<1 \\ 0, \quad \text{else} \end{cases}$, which is a kernel function, $b(x_i)$ represents the bin index associated with the Y component at pixel location x_i, and u is the color index in the histogram. By the similar method, we can construct the reference distribution which is gathered at the initial time $k=0$ and at location X_0, denoted as $\hat{q}_u = \hat{p}_u(X_0)$. The reference color model is updated with current target model in the new video image sequence in order to improve the performance of tracking.

Then, Bhattacharyya distance is used to measure the difference between the histogram obtained at location X_k^i and the reference histogram at time k as follows:

$$d_i = \sqrt{1 - \sum_{u=1}^{m} \sqrt{\hat{p}_u(y) \hat{q}_u}} \tag{7}$$

Finally, we obtain the observation model:

$$p(Z_k | X_k^i) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{d_i^2}{2\sigma^2}\right) \tag{8}$$

In this paper, the importance density function is selected $q(X_k^i | \tilde{X}_{0:k-1}^i, Z_{1:k}) = p(X_k^i | \tilde{X}_{k-1}^i)$, allowing for computational complexity and memory capacitance. Then,

considering all the weights are equal after resampling step, the weight of the i^{th} particle can be computed as:

$$w_k^i = p(Z_k \mid X_k^i) = \frac{1}{\sqrt{2\pi}\sigma}\exp(-\frac{d_i^2}{2\sigma^2}) \quad (9)$$

III. VARIATIONS ON PARTICLE FILTER

As described in the previous paper, the main drawback of Particle Filter is its high computational complexity. For each observation received, all the particles need to be processed. Moreover, to obtain an accurate approximation of posterior distribution, the number of particles is usually large. Therefore, reducing the number of particles is crucial for implementation of Particle Filter algorithm on an embedded system based on DSP processor. In this Section, we first reduce particle number by Mean-shift method. Then, according to the architecture of TI's DSP, we fuse the normalization and resampling steps to overcome some shortcomings of RSR. Finally, some tricks for DSP programming are introduced to improve computational performance.

A. *Combining Mean shift algorithm in Particle Filter*

The Mean-shift algorithm is a non-parametric method which converges to a local maximum of the measurement function under certain assumptions on the kernel behaviors. Therefore, the application of Mean-shift algorithm in Particle Filter can refine the location of particles, and then, reduce the particle number needed in tracking. The application of Mean-shift in Particle Filter has already been presented in the literature, such as, [8] [9]. However, in the traditional methods, the computation of the correlation between candidate region and reference template for every particle before resampling is time-consuming. In this paper, the Mean-shift process is applied between step 4) and step 5) described in Section 2. We assume that the number of replication for every particle is saved as r_i during resampling process. Then, after resampling, the new location of a particle is denoted as:

$$\tilde{X} = \frac{\sum_{i=1}^{N} X_k^i r_i g\left(\left\|\frac{X-X_k^i}{h}\right\|^2\right)}{\sum_{i=1}^{N} r_i g\left(\left\|\frac{X-X_k^i}{h}\right\|^2\right)} \quad (10)$$

Where, $X_k^i=[x_k^i, y_k^i]^T$ is the location of the i^{th} particle before resampling, $g(\|x\|^2)$ is a kernel function and h is window radius. The principle of Particle Filter Embedded Mean-shift algorithm can be illustrated in figure 2.

Through Mean-shift operation, every particle is more close to the local maximum, i.e. the real position of target. Therefore, less number of particles is needed to estimate the target status. Subsequently, the computational complexity is reduced greatly. As a result, make Particle Filter algorithm implement on embedded DSP system feasible.

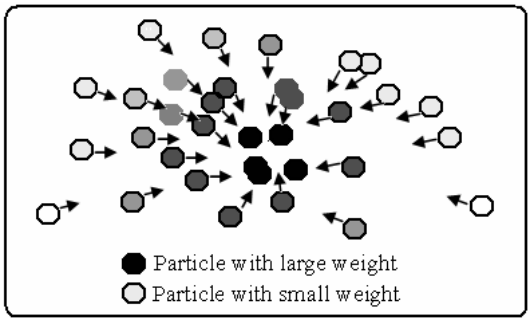


Figure 2. chematic representation of the Particle Filter embedded Mean-shift algorithm.

B. *Fusing normalization step and resampling step*

A common problem with Particle Filter is the degeneracy phenomenon. When after a few iterations, all but one particle will have negligible weight. The degeneracy phenomenon means that a large part of computation is devoted to the particles which only contribute little to the posterior distribution. To alleviate the phenomenon, some resampling methods have been proposed, such as: Stratified Resampling, Residual and Systematic Resampling [10~12]. From experiments, we find that the RSR in [12] is more suitable for implementation on embedded DSP system. However, it still has some shortcomings and we improve it in two ways:

1) The number of particles before and after RSR in [12] may change. In this paper, we can retain the particle number same after resampling by increment/decrement of particles similar with the method in [11].

2) We extend the RSR method to deal with unnormalized situation and this can neglect the division computation of weight normalization.

To extend the RSR, the original algorithm is amended as follows:

a) Initialize $W_N=0$, $u_0 \sim U[0,1]$.

b) *for* $i=1,...,N$, $W_N=W_N+w_k^i$.

c) $\text{temp1}=N/W_N$.

d) *for* $i=1,...,N$, $\text{temp2}=w_k^i*\text{temp1}-u_{i-1}$, $r_i=round(\text{temp2})+1$, $u_i=r_i-\text{temp2}$.

In this algorithm flow, the temp1 and temp2 are intermediate variables, r_i is the number of replication for the i^{th} particle.

Figure 3 shows a simulated experiment of the improved RSR algorithm. In this experiment, the number of particles is set to 64 and the weight of each particle is randomly sampled from [0, 1].

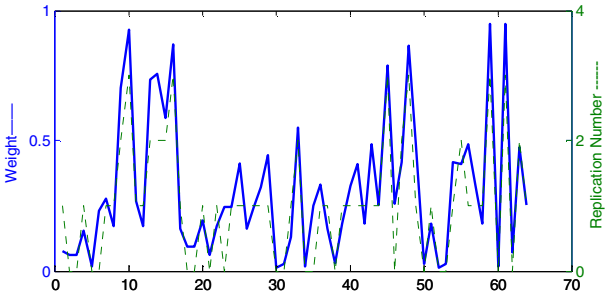


Figure 3. Simulated experiment result of the improved RSR algorithm in Matlab.

C. *Optimization of the code*

After improving the algorithm as described above, the computational complexity has been reduced a lot. In this section, we will introduce some tricks on DSP programming:

1) Design the formulas which are computed repeatedly as intermediate variables, for example, $\frac{1}{\sqrt{2\pi}\sigma}\exp(-\frac{d_i^2}{2\sigma^2})$ can be denoted as $C_1\exp(C_2 d_i^2)$, where $C_1=\frac{1}{\sqrt{2\pi}\sigma}$, $C_2=-\frac{1}{2\sigma^2}$.

2) Obtain the random numbers which fit $U[0,1]$ or $U[-1,1]$ offline, then, save these numbers in a table on DSP. Therefore, we can use look up table method to obtain random numbers

3) To enhance the efficiency, loop operations which are time consuming are rewritten in assemble language.

IV. Implementation of Particle Filter on Embedded DSP Hardware

A. *Hardware Description*

In order to implement the Particle Filter algorithm in real-time, an embedded system based on Multi-DSPs is designed with "pipelining" architecture as illustrated in figure 4. The hardware is composed of two TI DSP processors--TMS320DM642. TMS320DM642 is one of the highest-performance fixed-point DSP generation in the TMS320C6000™ DSP platform, which is based on the second-generation high-performance, advanced VelociTI™ very-long-instruction-word (VLIW) architecture (VelociTI.2™), making this an excellent choice for digital media applications. With performance of up to 5760 million instructions per second (MIPS) at a clock rate of 720 MHz, the TMS320DM642 device offers cost-effective solutions to high-performance DSP programming challenges. The peripheral set of TMS320DM642 includes: three configurable video ports; a 10/100 Mb/s Ethernet MAC (EMAC); an inter-integrated circuit (IIC) Bus module; two multichannel buffered serial ports (McBSPs); a user-configurable 16-bit or 32-bit host-port interface (HPI16/HPI32).

The digital video decoder of our DSP system is selected TVP5146, and encoder is SAA7121.The TVP5146 device is a high quality, single-chip digital video decoder, which includes four 10-bit 30-MSPS A/D converters (ADCs), and supports decoding of NTSC, PAL, and SECAM composite and S-video into component YCbCr. The SAA7121 encodes digital YCbCr video data to an NTSC or PAL CVBS or S-video signal. The circuit accepts CCIR compatible YCbCr data with 720 active pixels per line in YUV 4:2:2 multiplexed formats.

The SDRAM and Flash connected to EMIF are 64M bytes and 4M bytes respectively. Every DSP has its own 32M bytes SDRAM, but they share only one Flash for storing program data, which make the second DSP can only boot from HPI data bus instead of EMIF memory.

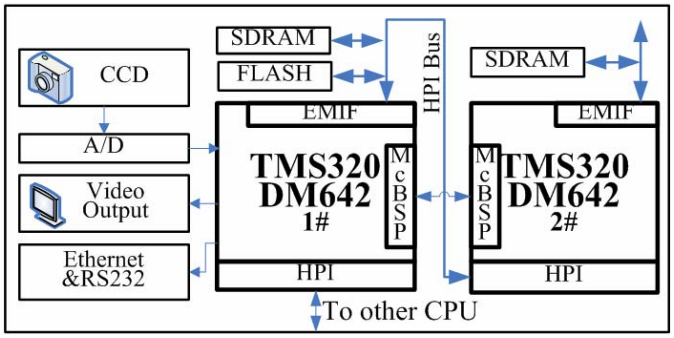


Figure 4. Architecture of the Multi-DSPs system.

B. *Task Assignment of Different DSP*

As we can see from the figure 4, the two DSPs are communicated with each other through 32 bits 100MHZ HPI data bus. The first DSP is designed to convert the input video image signal from CCD camera into digital data, and inversely, convert the processed digital image data into PAL analogy signal or send the digital image date to remote terminal for further analysis through Ethernet, meanwhile, the first DSP is also used to image preprocessing, target detection, target reference color model constructing and updating; The second DSP is particularly designed to implement the Particle Filter algorithm, include initialization, state predication, weight calculation, resampling, and Mean-shift operation, etc. The detailed task assignment of the two DSPs is described in figure 5.

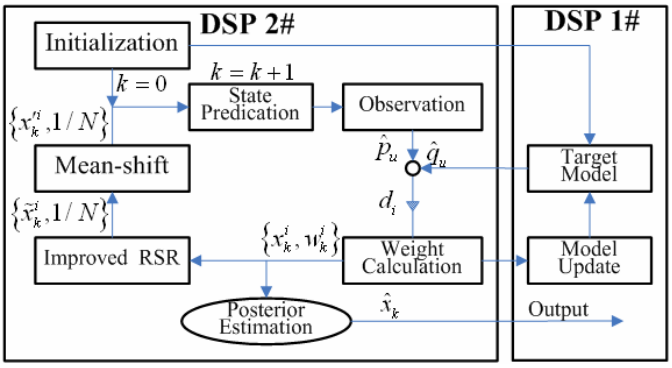
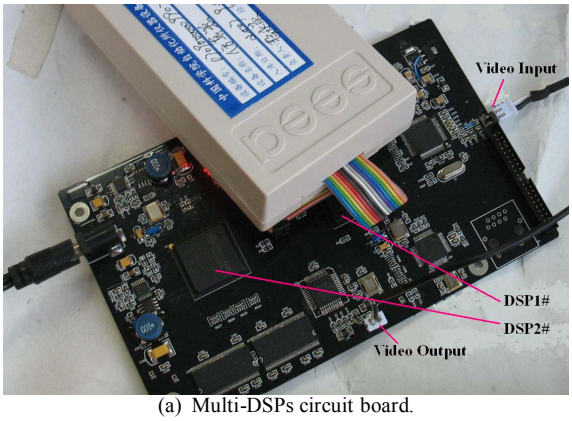


Figure 5. Assignment of the two DSPs.

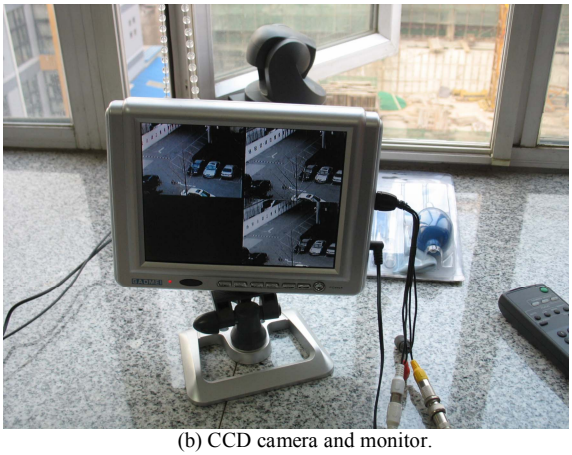
V. Experimental Results

To test the proposed method, an experimental Intelligent Vision System (IVS) is built based on the description in section 4. The composition of hardware system is showed in figure 6, which composed of Multi-DSPs circuit board, as well as CCD camera and monitor. The camera we selected is

SONY EVI-D70P, which combines a high quality 1/4 type Exview HAND^{TM} CCD color camera with the flexibility of a remote Pan/Tilt/Zoom operation, and the signal system is PAL.



(a) Multi-DSPs circuit board.



(b) CCD camera and monitor.

Fig.6. Experimental IVS.

To illustrate our method is practicable and can be well implemented on DSP platform rather than usual Personal Computer, we use a set of frames of a video sequence consisting of one person keeping away from or keeping close to the fence of one international airport. In this case, the objective is to keep track of the location of the person in any time. In the experiment, the frame size is 720×320 originally, we only sample 352 × 288 per-frame for reducing computational complexity, and σ in equation (8) or (9) is set to o.5, the m in equation (5) is set to 4 for reducing memory space, and the number of particles is set to 2^6=64. The processing speed is about 12 frames per-second (fps), which is suitable for real applications.

Frame 6 Frame 31

Frame 90 Frame 115

Frame 126 Frame 151

Fig.7. Tracking results.

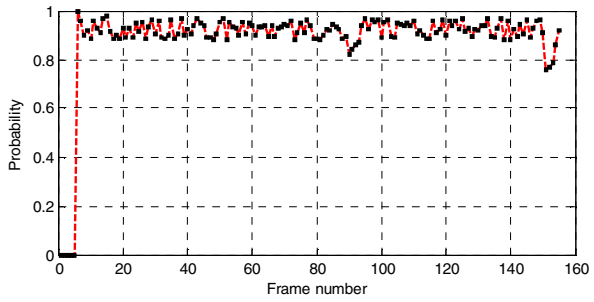


Fig.8. Probability of existence for object in the video sequence of Fig. 7.

The experimental results are indicated in Figure 7 and figure 8. We selected six frames from over 200 sequential video frames, which are displayed in figure 7. The person appears in the 6th image frame and continues to exist throughout the video sequence. Combined with object detection algorithm running on the first DSP, the Particle Filter algorithm proposed in this paper detects the person in the frame number 6: the probability of existence of the person jumps to the value of 1 between frames 5 and 6, as indicated in Figure 8. The detected object/person is indicated in each image by a red rectangle, located at the estimated object position. Frames 90 and 151 are noteworthy: here the person changes its motion direction and gait abruptly, and this is reflected in the drop of the probability of existence for the person, we can see from the figure 8, the probability of existence drops to 0.82 and 0.76 respectively.

Ultimately, three different platforms are used to run the same Particle Filter algorithm for the purpose of comparing. The detailed results are described in TABLE I, from which we can conclude that the Multi-DSPs system is an ideal hardware platform to implement Particle Filter Object-tracking algorithm for its high computational capability compared with the single DSP system as well as its high performance-cost ratio and ultra-low power consumation compared with the PC system.

TABLE I. COMPARATION OF THREE PLATFORM

Items	Different Platform		
	Single DSP System	***Multi-DSPs System***	***PC System***
Resource description	DM642 500HZ SDRAM 32M CCS2.20.18	DM642 500HZ SDRAM 64M CCS2.20.18	Pentium IV 2.4G DDR2 1G XP VC6.0
Processing speed	4~6 fps	11~14 fps	20~26 fps
Power consumation	=< 7.2w	=< 10.8w	>> 200w

VI. CONCLUSIONS AND DISCUSSION

In this paper, an embedded hardware architecture based on Multi-DSPs is proposed for speeding up the basic computation of Particle Filter, thereby, making Particle Filtering based solutions amenable to real time constraints. Simultaneously, on this embedded Multi-DSPs system, we also do some improvement to the Particle Filter algorithm for realization. First, the number of particles is reduced by fusing Mean-shift after resampling step. Then, RSR (Residual Systematic Resampling) method is mended to reduce N times division computation and to retain the same number of particles pre-and-post resampling procedure. The performance of the proposed Multi-DSPs system and optimized Particle Filter algorithm are evaluated qualitatively on real-world video sequences with moving target/person.

Although the proposed algorithm and hardware platform are effective for person tracking in our laboratorial scene, there are two limitations inherent to our method. First, as only color cue is used, our tracking algorithm in fact is simply a general heuristic method and could not be similarly effective when used in other tracking problems under clutter and low-resolution environment. Under considering the prerequisite condition of computational performance, we are preparing to fuse other features, e.g., texture, edge, simple shape and motion information to allow more robustness and more flexibility in the object-tracking. Besides, as the number of particles required in multiple objects tracking is increased many times comparing with single object tracking, which make our DSP platform difficult to meet the real-time requirement when used to track multiple objects, especially more than three objects. To solve the two problems presented above, we are designing a new parallel reconfigurable Multi-Processors architecture, which is composed of four DSP processors and one Intel(Marvell) Xscale/ARM processor-PXA270, and the DSP we selected is DAVINCI digital medial processors-TMS320DM6437(C64x+ core, 600MHZ) instead of TM320DM642.

ACKNOWLEDGMENT

The authors would like to thank Lei Ma and Yuan Tian for their help in experiments, and thank Xinbo Deng for his help in schematic and PCB design. This research is funded by the National Hi-Tech R&D Program (no.2007AA809505B), and Youth Innovation Foundation of Institute of Automation, Chinese Academy of Sciences (Grant DI07J01).

REFERENCE

[01] C. Jacek, R. Branko, M. Benoit, "A Particle Filter for joint detection and tracking of color objects," Image and Vision Computing, Vol. 25, Issue. 8, pp. 1271-1281, Aug. 2007.

[02] P. Brasnett, L. Mihaylova, D. Bull, N. Canagarajah, "Sequential Monte Carlo tracking by fusing multiple cues in video sequences," Image and Vision Computing, Vol. 25, Issue. 8, pp. 1217-1227, Aug. 2007.

[03] J. Cho, S. Jin, X. Pham, J. Jeon, J. Byun, H. Kang, "A real-time object tracking system using a Particle Filter," Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems, Beijing, pp. 2822-2827, Oct. 2006.

[04] A. Sankaranarayanan, R. Chellappa, A. Srivastava, "Algorithmic and architectural design methodology for Particle Filters in hardware," Proceedings of the 2005 International Conference on Computer Design, pp. 275-280, Oct. 2005.

[05] T. Bando, T. Shibata, K. Doya, S. Ishii, "Switching Particle Filters for efficient real-time visual tracking," Proceedings of the 17th International Conference on Pattern Recognition, Vol. 2, pp.720-723, Aug.2004.

[06] S. Zhou, R. Chellappp, B. Moghaddam, "Visual tracking and recognition using appearance-adaptive models in Particle Filters," IEEE Transactions on Image Processing, Vol. 12 ,Issue. 11, pp. 1491-1504, Nov. 2004.

[07] P. Perez, C. Hue, J. Vermaak and M. Gangnet, "Color-based probabilistic tracking," Proceedings of 7th European Conference on Computer Vision (ECCV), pp. 661-675, May. 2002.

[08] K. Deguchi, O. Kawanaka, T. Okatani, "Object tracking by the Mean-shift of regional color distribution combined with the Particle-filter algorithm," 17TH International Conference on Proceedings of the Pattern Recognition, Vol. 3, pp. 506-509, Aug. 2004.

[09] E. Maggio, A. Cavallaro, "Hybrid Particle Filter and Mean shift tracker with adaptive transition model," Proc. of IEEE Signal Processing Society International Conference on Acoustics, Speech, and Signal Processing (ICASSP), Philadelphia, Vol. 2, pp. 221-224, Mar. 2005.

[10] M. Bolic, Architectures for efficient implementation of Particle Filters, Ph.D. Dissertation, State University of New York at Stony Brook, 2004.

[11] S. Hong, M. Bolic, "An efficient fixed-point implementation of residual resampling scheme for high-speed Particle Filters," IEEE Signal Processing Letters, Vol. 5, pp. 482-485, Nov. 2004.

[12] M. Boloc, P. Djuric, S. Hong, "New resampling algorithms for Particle Filters," IEEE International Conference on Acoustics, Speech, and Signal Processing, Vol. 2, pp. 589-592, Apr. 2003.