Proceedings of the 2006 IEEE/RSJ
International Conference on Intelligent Robots and Systems
October 9 - 15, 2006, Beijing, China

A Real-Time Object Tracking System Using a Particle Filter

Jung Uk Cho, Seung Hun Jin,
Xuan Dai Pham, Jae Wook Jeon
School of
Information and Communication Engineering
Sungkyunkwan University
Suwon, Korea
{ichead, coredev, pdxuan}@ece.skku.ac.kr,
jwjeon@yurim.skku.ac.kr

Jong Eun Byun
Research and Development
Center
Nexteye Company
Anyang, Korea
jebyun@nexteye.com

Hoon Kang
School of
Electrical and Electronics Engineering
Chungang University
Seoul, Korea
hkang@cau.ac.kr

***Abstract* - Particle filters have attracted much attention due to their robust tracking performance in cluttered environments. Particle filters maintain multiple hypotheses simultaneously and use a probabilistic motion model to predict the position of the moving object, and this constitutes a bottleneck to the use of particle filtering in real-time systems due to the expensive computations required. In order to track moving objects in real-time without delay and loss of image sequences, a particle filter algorithm specifically designed for a circuit and the circuit of the object tracking algorithm using the particle filter are proposed. This circuit is designed by VHDL (VHSIC Hardware Description Language), and implemented in an FPGA (Field Programmable Gate Array). All of the functions of the proposed particle filter used to track moving objects are implemented in the FPGA. The object tracking system employing this circuit is implemented and then its performance is measured.**

Index Terms – Object tracking, particle filter, FPGA, VHDL

I. INTRODUCTION

Object tracking has been an active research area in the vision community in recent years. It has many potential applications in the fields of intelligent robots [1], monitoring and surveillance [2], human computer interfaces [3], smart rooms [4][5], vehicle tracking [6], biomedical image analysis [7], video compression [8], etc [9][10][11]. Numerous approaches have been proposed to track moving objects in image sequences, such as the Kalman filter, the extended Kalman filter, and the particle filter. Among these different approaches, the particle filter has been successfully applied to the tracking of moving objects in cluttered environments. The basic idea of the particle filter is that the posterior density is approximated by a set of discrete samples (called particles) with associated weights [11][12].

The particle filter is used to apply a recursive Bayesian filter based on the propagation of a sample set over time, maintain multiple hypotheses at the same time and use a stochastic motion model to predict the position of the object. Maintaining multiple hypotheses allows the tracker to handle clutter in the background, and recover from failure or temporary distraction. However, there are expensive computational demands in this approach, and this constitutes a bottleneck to the application of particle filtering to real-time systems [13].

In order to track moving objects in real-time without delay and loss of the image data, a particle filter algorithm specifically designed for a circuit and the circuit of the object tracking algorithm using the particle filter are proposed. This circuit is designed by VHDL and implemented in an FPGA. An object tracking system with this FPGA is implemented and its performance is measured.

This paper is organized as follows; in section II, the basic particle filter algorithm and particle filter for the circuit are explained. In section III, the circuit used for the tracking of moving objects using a particle filter is designed by VHDL, and this circuit is explained in detail using a block diagram. In section IV, the tracking circuit is implemented in an FPGA and its performance is measured. Finally, the conclusion is given in section V.

II. PARTICLE FILTER

Particle filters provide robust tracking of moving objects in a cluttered environment. They are used in the case of non-linear and non-Gaussian problems where the interest lies in the detection and tracking of moving objects. This section explains the particle filter algorithm.

Particle filters use multiple discrete "particles" to represent the belief distribution over the location of a tracked object.

In the Bayes filtering paradigm, we recursively update the posterior distribution over the current state X_t given all observations $Z^t = \{Z_1,..,Z_t\}$ up to and including time t, as follows:

$$p(X_t \mid Z^t) = kp(Z_t \mid X_t)p(X_t \mid Z^{t-1}) \quad (1)$$
$$= kp(Z_t \mid X_t)\int_{X_{t-1}} p(X_t \mid X_{t-1})p(X_{t-1} \mid Z^{t-1}),$$

where the likelihood $p(Z_t|X_t)$ expresses the *measurement model* and $p(X_t|X_{t-1})$ is the *motion model*. In a particle filter, we approximate the posterior $p(X_{t-1}|Z^{t-1})$ recursively as a set of N weighted samples $\{X_{t-1}^{(i)}, w_{t-1}^{(i)}\}_{i=1}^{N}$, where $w_{t-1}^{(i)}$ is the

weight for particle $X_{t-1}^{(i)}$. Given this equation, we can use a Monte Carlo approximation of the integral and get:

$$p(X_t \mid Z^t) \approx kp(Z_t \mid X_t)\sum_{i=1}^{N} w_{t-1}^{(i)} p(X_t \mid X_{t-1}^{(i)}) . \quad (2)$$

One way to view a particle filter is as an importance sampler for this posterior distribution. Specifically, N samples $X_t^{(s)}$ are drawn from the *proposal distribution*

$$q(X_t) = \sum_{i=1}^{N} w_{t-1}^{(i)} p(X_t \mid X_{t-1}^{(i)}) , \quad (3)$$

and then weighted by the likelihood, i.e.

$$w_t^{(s)} = p(Z_t \mid X_t^{(s)}) . \quad (4)$$

This results in a weighted particle approximation $\{X_t^{(s)}, w_t^{(s)}\}_{s=1}^{N}$ for the posterior $p(X_t|Z^t)$ at time t [14][15].

The general operation of the particle filter tracker for the tracking of objects in video images is illustrated in Fig. 1. Figure 2 shows a flowchart of the general operation of the particle filter. Each particle which tracks moving objects is represented by a point, so the group of particles is referred to as a point cloud. The principal steps in the particle filter algorithm include: [11][16]

1) Initialization
   Draw a set of particles for the prior $p(X_0)$ to obtain $\{X_0^{(i)}, w_0^{(i)}\}_{i=1}^{N}$, let k=1.
2) Sampling
   (a) For i=1,…,N
   Sample $X_k^{(i)}$ from the proposal distribution $p(X_k^{(i)} \mid X_{k-1}^{(i)})$.
   (b) Evaluate the new weights
   $$w_k^{(i)} = p(Z_k \mid X_k^{(i)}), \quad i=1,\ldots,N \quad (5)$$
   (c) Normalize the weights
   $$w_k^{(i)} = \frac{w_k^{(i)}}{\sum_{j=1}^{N} w_k^{(j)}}, \quad i=1,\ldots,N \quad (6)$$
3) Output
   Output a set of particles $\{X_k^{(i)}, w_k^{(i)}\}_{i=1}^{N}$ that can be used to approximate the posterior distribution as $p(X_k \mid Z^k) \approx \sum_{i=1}^{N} w_k^{(i)} \delta(X_k - X_k^{(i)})$ and the estimate as $E_{p(\mathrm{g}Z^k)}(f_k(X_k)) \approx \sum_{i=1}^{N} w_k^{(i)} f_k(X_k^{(i)})$, where $\delta(\mathrm{g})$ is the Dirac delta function.
4) Resampling
   Resample particles $X_k^{(i)}$ with probability $w_t^{(i)}$ to obtain N independent and identically distributed random particles $X_k^{(j)}$, approximately distributed according to $P(X_k|Z^k)$.
5) k=k+1, go to step2.

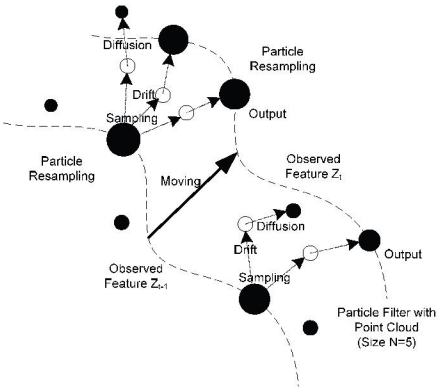

Fig. 1. Visualization of the particle filter.

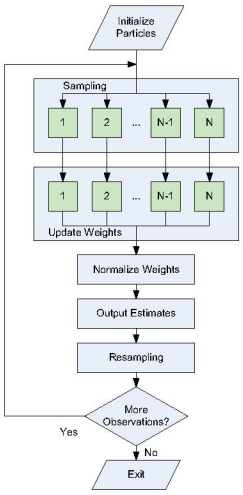

Fig. 2. Flowchart of the particle filter.

III. Circuit Used for Object Tracking

The object tracking of visual features in cluttered environments is a challenging research area in the vision community. Due to their sample-based representation, particle filters have been applied with success to a variety of state estimation problems including visual tracking. The increased representational power of particle filters, however, comes at the expense of higher computational complexity.

In previous particle filter applications, the prevalent approach in real-time applications was to update the filter as often as possible and to discard the image data that arrives during the update process. Obviously, this approach is prone to lose valuable image data. At first sight, the sample based representation of particle filters appears to offer an alternative approach which is similar to an anytime implementation: i.e. whenever a new observation arrives, sampling is interrupted and the next observation is processed. Unfortunately, such an approach can result in too small sample sets, thus causing the filter to diverge [17]. Although the real-time tracking of a moving object is possible using a particle filter, this approach results in most of the system resources being monopolized by the tracking process.

Herein, a circuit is specially designed for an object tracking system using a particle filter, in order to resolve the abovementioned problem. It is impossible for the algorithm of the particle filter to be applied to the circuit without some modification, due to the restrictions imposed by the hardware, such as the timing, resources, size, and power consumption. Therefore, a modified architecture is proposed for the circuit implementation of the particle filter.

In order to apply the particle filter to the process of tracking moving objects in real-time without delay and loss of image data, a specially designed particle filter circuit is proposed. This circuit is shown in Fig. 3 and consists of five modules: the Camera Controller, Image Store Memory, Image Subtractor, Noise Reduction Filter, and Particle Filter. It is designed by VHDL and implemented in an FPGA in order to track moving objects in real-time. In Fig. 3, the gray box, consisting of the Image Subtractor, Noise Reduction Filter, and Particle Filter modules, represents the implemented circuit in an FPGA.

A. Camera Controller

In the Camera Controller module, the Sync Separator generates the signals used for controlling the camera and the A/D Converter converts the analog image data into digital image data. The image sync signals, *Sync*, and digital image data, *Raw Pixels*, of the Camera Controller module are used in all of the modules of the object tracking circuit.

B. Image Store Memory

Based on the sync signals, *Sync*, obtained from the Sync Separator in the Camera Controller module, the Image Store Memory module stores the image data, *Raw Pixels*, arriving from the A/D Converter in the Camera Controller module frame by frame. This module transfers the image data of the previous and current frames, *Stored Pixels*, to the Image Subtractor module in order to generate the IFDs, *IFD*, for each Particle Filter modules. Since it is difficult and inefficient to store the image of a frame in an FPGA, the FIFO (First In, Fist Out) memories are used to store the image of a frame.

C. Image Subtractor

The Image Subtractor module generates the IFDs, *IFD*, by pixel by pixel differencing between the images of the previous and current frames, *Stored Pixels*, obtained from the Image Store Memory module. These IFDs, *IFD*, $\Delta P_t(x,y)$ are determined as

$$\Delta P_t(x,y) = \left| I_{t-\Delta}(x,y) - I_t(x,y) \right|, \tag{7}$$

where $I_t(x,y)$ is the value at the $(x,y)^{th}$ pixel that has the range [0, 255] at time t, and $I_{t-\Delta}(x,y)$ is the value at the $(x,y)^{th}$ pixel that has the range [0, 255] at time t-Δ. The range [0, 255] is the same as that of a gray level image. The IFD, *IFD*, is used in the Particle Filter module as a tracking feature after the noise is reduced in the Noise Reduction Filter module.

D. Noise Reduction Filter

Since the IFDs, *IFD*, from the Image Subtractor module often contain noise which causes errors during image processing, a Noise Reduction Filter block is included to reduce it by applying a threshold, where the noise reduction technique is described by

$$\Delta P_t(x,y) = \begin{cases} \left| I_t(x,y) - I_{t-\Delta}(x,y) \right| & , \left| I_t(x,y) - I_{t-\Delta}(x,y) \right| > V_{TH} \\ 0 & , \left| I_t(x,y) - I_{t-\Delta}(x,y) \right| \le V_{TH} \end{cases} \tag{8}$$

where, V_{TH} is a threshold value used to reduce the noise that is set in advance. The noiseless IFD without noise, *Noiseless IFD*, is used in the Particle Filter module to track moving objects.

E. Particle Filters

The Particle Filter module in the proposed object tracking circuit is based on the technique described in section 2.2, which can track moving objects. Its inputs, *Noiseless IFD* and *Sync*, represent the IFD, whose level of noise is reduced in the Noise Reduction Filter module, and the image sync signal from the Camera Controller module, respectively. Its outputs, *Tracking Image* and *Trajectory Data*, represent the tracking image data and the trajectory data of the tracking object, respectively. Therefore, this circuit can obtain the tracking image and trajectory information of the tracked object in real-time.

The operation of the particle filter circuit is illustrated in the flowchart shown in Fig. 4. The principal steps in the particle filter circuit include:

1) Initialization
   Draw a set of particles uniformly.
2) Sampling
   (a) Sample the position of the particles from the proposal distribution.
   (b) Find the features of the moving object (IFD).
   (c) Update the weight of the particles.
   (d) Normalize the weight of the particles.
3) Output
   Output the mean position of the particles that can be used to approximate the posterior distribution.
4) Resampling
   Resample the particles with probability to obtain independent and identically distributed random particles.
5) Go to the sampling step.

Figure 5 shows the architecture of the Particle Filter module. It consists of five blocks: the Particle Initiator, Particle Sampler, Coordinates Comparator, Particle Normalizer, Data Output, and Particle Selector.

Figure 6 shows the Particle Initiator block. The Particle Initiator block initializes the particles when it starts to track objects for the first time or when it attempts to find a new moving object after the previously tracked object has disappeared from view. Whenever the IFD of some pixel is larger then the initial value, V_{INIT}, the coordinates of its position are stored in the Initialization Lookup Table. This stored data is used for the purpose of repositioning the particles rapidly for the purpose of tracking the newly detected moving object.

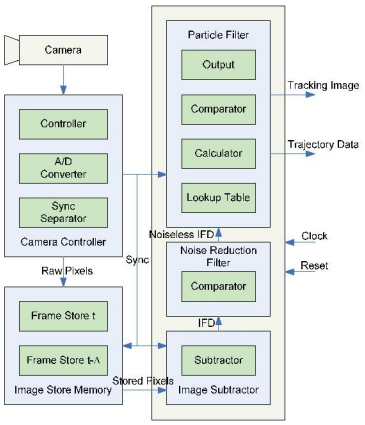

Fig. 3. Block diagram of the object tracking circuit using a particle filter.

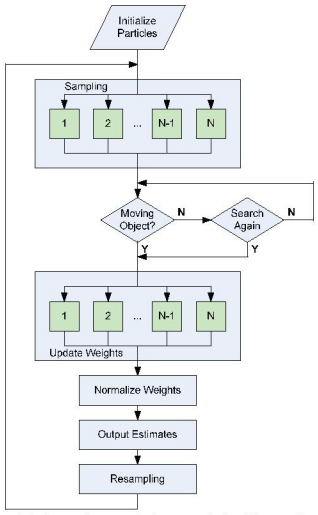

Fig. 4. Flowchart of the particle filter circuit.

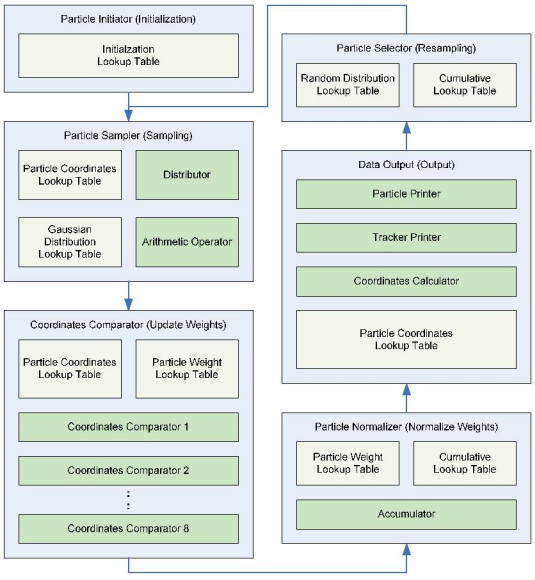

Fig. 5. Block diagram of the particle filter module.

The Particle Sampler block performs the sampling step of the particle filter algorithm. It samples the new position of the particle according to the resampled particles and Gaussian white noises from the Particle Selector and Gaussian Distribution Lookup Table, respectively. The Gaussian white noises contained in the Gaussian Distribution Lookup Table were calculated using the Gaussian Function of the C program language and stored in advance. The coordinates of the new particles are stored in the Particle Coordinates Lookup Table. This operation is illustrated in Fig. 7.

Figure 8 shows the Coordinates Comparator block used to update the weights of the particles. The weights of the particles are calculated using the IFD of the pixels. The coordinates of the particles are compared with one of the current pixels in order to obtain the weight of each particle. The IFD becomes the weight of each particle, so those particles with a high IFD have high weights. In order to perform this operation in real-time, the eight comparators operate simultaneously in parallel. Therefore, during 1 pixel clock cycle, the coordinates of all of the particles are compared with those of one of the current pixels. The Particle Coordinates Lookup Table stores the coordinates of each particle and the Particle IFD Lookup Table stores the weight (IFD) of each particle. The Particle Coordinates Lookup Table and the Particle weight Lookup Table store the coordinates and the weight (IFD) of each particle, respectively.

The Particle Normalizer block normalizes the weights of the particles, as shown in Fig. 9. In order to accomplish this, it generates a Cumulative Lookup Table according to the data contained in the Particle IFD Lookup Table using the Accumulator. This cumulative lookup table contains the cumulative distribution of the weights of the current particles. The range of this cumulative distribution in the Cumulative Lookup Table is $[0, \sum_{i=1}^{N} IFD]$.

In the general particle filter algorithm, this distribution has to be normalized, and this requires dividing by the sum of the weights. However, division is not an efficient operation in a hardware implementation, because it needs an additional circuit and involves a lengthy procedure. Therefore, whenever possible, division operations are avoided when designing hardware circuits. Therefore, the division operation in the weight normalization step is eliminated and this operation is compensated for in the Particle Selector (Resampling) by using a variable resampling range.

Figure 10 shows the method used to obtain the trajectory of the tracked object in the Data Output block. It calculates the mean position of the particles. This circuit has a parallel structure consisting of a lookup table, nine adders and nine dividers, in order to save processing time by executing the operations concurrently.

The Particle Selector block performs the resampling step in the particle filter algorithm. It selects new particles among the current particles according to the uniform random distribution obtained from the Random Distribution Lookup Table and the cumulative function from the Cumulative

Lookup Table in the Particle Normalizer block. The uniform random distribution has the range [0, 1] and the cumulative distribution has the range [0, $\sum_{i=1}^{N} IFD$]. The resampling range varies according to the range of the cumulative distribution in the Cumulative Lookup Table. New particles are selected randomly in the range [0, $\sum_{i=1}^{N} IFD$] according to the random numbers in the range [0, 1] obtained from the Random Distribution Lookup Table. Herein, the method of the variable resampling range is to magnify the range of the random distribution as much as the range of the cumulative distribution and this method is proposed and implemented in this block. This selection method is similar to the roulette wheel selection method used in Genetic Algorithm. Therefore, those particles with high a weight are likely to be selected more than those with a low weight. The resampling step plays an important role in the particle filter algorithm, since it avoids the production of particles with a dominating weight. This operation is shown in Fig. 11.

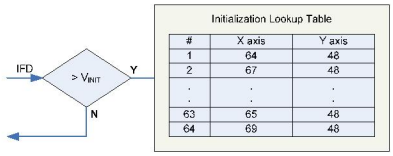

Fig. 6. The particle initiator block.

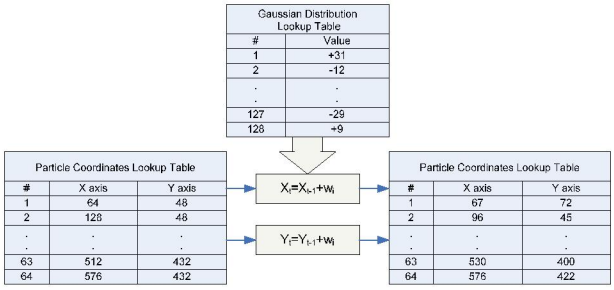

Fig. 7. The particle sampler block.

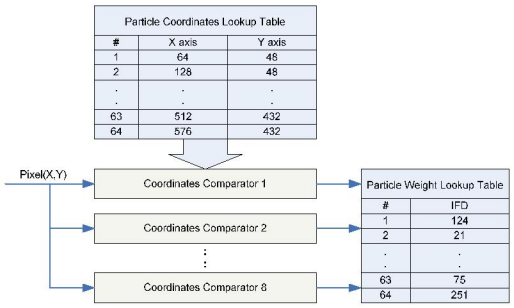

Fig. 8. The coordinates comparator block.

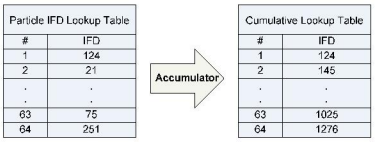

Fig. 9. The particle normalizer block.

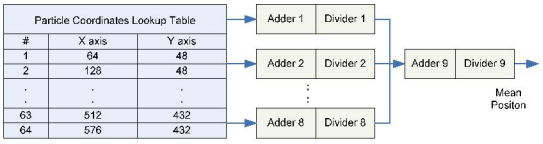

Fig. 10. The data output block.

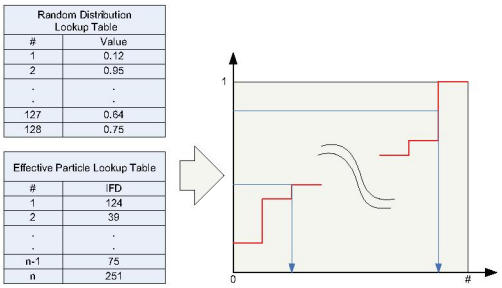

Fig. 11. The particle selection block.

IV. Experiment

The circuit of the object tracking algorithm using the particle filter is designed by VHDL and is implemented in an XC2V6000-4CFF1152 FPGA, which has 6M system gates, 76,032 logic cells, and 1,104 I/O pins [18]. A summary of the device utilization characteristics for the circuit of the object tracking algorithm using the particle filter is shown in Table I. A vision system using this FPGA is built, as shown in Fig. 12. It can acquire images from a camera and send both the raw and processed images and the object trajectory data generated by the particle filter to a PC through PCI (Peripheral Component Interconnect) bus.

A high frame processing rate and low latency are important for many applications that must provide quick decisions based on events in the scene [19]. When the vision system with the circuit of the proposed object tracking using particle filter is applied to an RS-170 camera which produces images consisting of 640×480 pixels, it can process the images at speeds of up to 56.52 fps (frames per second) to track a moving object and send these processed images to the PC. The performance of this circuit is measured by evaluating the average maximum delay time. Fig. 13 shows the results obtained from the tracking of a moving object in a video sequence. There is some delay between the images shown in Fig. 13-(a), (b), (c), (d). In Fig. 13, the white points represent the particles and the square with the cross in the center represents the trajectory of the moving object. The 64 particles are used to track a moving object in this experiment.

TABLE I
DEVICE UTILIZATION CHARACTERISTICS FOR THE OBJECT TRACKING CIRCUIT USING A PARTICLE FILTER

Device utilization characteristics				
Number of Slices:	8945	out of	33792	26%
Number of Slice Flip Flops:	6603	out of	67584	9%
Number of 4 input LUTs:	11603	out of	67584	17%
Number of bonded IOBs:	143	out of	824	17%
Number of MULT18X18s:	1	out of	144	0%
Number of GCLKs:	3	out of	16	18%

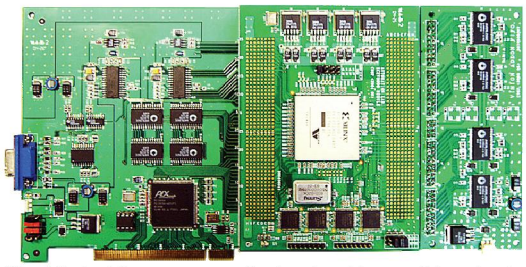

Fig. 12. A vision system employing the object tracking circuit using a particle filter.

(a) The moving object tracking image *bicycle* at frame t (b) The moving object tracking image *bicycle* at frame $t+\Delta$ (c) The moving object tracking image *bicycle* at frame $t+2\Delta$ (d) The moving object tracking image *bicycle* at frame $t+3\Delta$

Fig. 13. The result of the object tracking circuit using a particle filter.

V. CONCLUSION

In this paper, we described the development of the object tracking circuit using a particle filter for the purpose of proving efficient object tracking in real-time without delay and loss of image data. Since the proposed object tracking circuit using the particle filter is designed by VHDL and implemented in an FPGA, no additional system is needed to implement the object tracking system itself. Since only a small percentage of the system resources are allocated for object tracking, the rest can be assigned to the preprocessing stages or to high level tasks such as recognition, trajectory interpretation, and reasoning. This real-time object tracking circuit can be applied to a wide range of applications.

ACKNOWLEDGMENT

This research was performed for the Intelligent Robotics Development Program, one of the 21st Century Frontier R&D Programs funded by the Ministry of Commerce, Industry and Energy of Korea.

REFERENCES

[1] C. Kwok, D. Fox, M. Meila, "Adaptive Real-Time Particle Filter for Robot Localization," *Proceedings of Robotics and Automation*, Vol. 2, pp. 2836-2841, Sept. 2003.

[2] Y. Cui, S. Samarasekera, Q. Huang, M Greiffenhangen, "Indoor Monitoring Via the Collaboration Between a Peripheral Sensor and a Foveal Sensor," *Proceedings of Visual Surveillance*, pp. 2-9, Jan. 1998.

[3] G. R. Bradski, "Real Time Face and Object Tracking as a Component of a Perceptual User Interface," *IEEE workshop on Applications of Computer Vision*, Princeton, pp. 214-219, 1998.

[4] S. S. Intille, J. W. Davis, A. F. Bobick, "Real-Time Closed-World Tracking," *Proceedings of Vision and Pattern Recognition*, pp. 697-703, 1997.

[5] C. R. Wren, A. Azarbayejani, T. Darrel, A. P. Pentland, "Pfinder: Real-Time Tracking of the Human Body," *IEEE Transactions on Pattern Analysis and Machine intelligence*, Vol. 19, pp. 780-785, July 1997.

[6] M. Betke, E. Haritaoglu, S. S. Davis, "Multiple Vehicle Detection and Tracking in Hard Real-Time," *Proceedings of Intelligent Vehicles Symposium*, pp. 351-356, Sept. 1996.

[7] L. Pan, J. L. Prince, J. A. C. Lima, N. F. Osman, "Fast Tracking of Cardiac Motion Using 3D-HARP," *IEEE Transactions on Biomedical Engineering*, Vol. 52, pp. 1425-1435, Aug. 2005.

[8] A. Eleftheriadis, A. Jacquin, "Automatic Face Location Detection and Tracking for Model-Assisted Coding of Video Teleconference Sequence at Low Bit Rates," *Signal Processing – Image Communication*, Vol. 7, pp. 231-248, 1995.

[9] Blake, M. Isard, *Active Contours*. Springer-Verlag, 1998.

[10] D. Comaniciu, V. Ramesh, P. Meer, "Real-Time Tracking of Non-Rigid Objects Using Mean Shift," *IEEE Conference on Computer Vision and Pattern Recognition*, South Carolina, Vol. 2, pp. 142-149, 2000

[11] P. Li, T. Zhang, "Visual Contour Tracking Based on Particle Filters," *Proceedings of Generative-Model-Based Vision*, Copenhagen, pp. 61-70, June 2002.

[12] S. Blackman, *Multiple-Target Tracking with Radar Applications*. Dedham, MA: Artech House, 1986.

[13] C. Shan, Y. Wei, T. Tan, F. Ojardias, "Real Time Hand Tracking by Combining Particle Filtering and Mean Shift," *Proceedings of Automatic Face and Gesture Recognition*, pp. 669-674, May 2004.

[14] B. Ristic, S. Arulampalam, N. Gordon, *Beyond The Kalman Filter*. Boston·London, Artech House, 2004.

[15] C. Hue, J. L. Cadre, P. Perez, "A Particle Filter to Track Multiple Objects," *IEEE Workshop on Multi-Object Tracking*, pp. 61-68, July 2001.

[16] C. Hue, J. L. Cadre, P. Perez, "Sequential Monte Carlo Methods for Multiple Target Tracking and Data Fusion," IEEE Transactions on Signal Processing, Vol. 50, No. 2, pp. 309-325, Feb. 2002.

[17] C. Kwok, D. Fox, M. Meila, "Real-Time Particle Filters," *Proceedings of the IEEE*, Vol. 92, Issue 3, pp. 469-484, March 2004.

[18] Xilinx Inc., "Virtex-II Platform FPGAs: Complete Data Sheet," available from www.xilinx.com, March 2005.

[19] J. I. Woodfill, G. Gordon, R. Buck, "Tyzx DeepSea High Speed Stereo Vision System," *Conference on Computer Vision and Pattern Recognition*, pp. 41-45, June 2004.