Modified Visual Target Tracking Algorithm and Its FPGA Implementation

Zhi-Ying Du[1], Shu-Hui Wang[2], and Dong-Bin Pei[3]

School of Communication and Information Engineering, Shanghai University
No.99, Shangda Road, Shanghai, China 200072

[1]duzhiying@live.com
[2]wongshgogo@gmail.com
[3]dongbinpei@126.com

Abstract— **Visual target tracking is the key problem in intelligent video processing. CamShift and Particle Filtering are classic and effective in visual target tracking algorithms, but they both need to analyse a large amount of probability statistic, leading to high algorithm complexity and low calculation efficiency. FPGA provides a competitive alternative for hardware acceleration to these applications. In this paper, we modify CamShift and Particle Filtering algorithms and propose a FPGA-based hardware accelerating architecture. Experiments show the embedded architectures have good performance and the Particle Filtering algorithm shows better robustness and real-time performance.**

***Keywords*— target tracking; CamShift; Particle Filtering; FPGA; embedded system**

I. INTRODUCTION

Visual target tracking is the key problem in intelligent video processing applications. It is related to the image processing and pattern recognition. In military, precision guided system [1] uses the accurate tracking technique to get target location. In transportation, intelligent transportation system [2] can analyse the traffic flow statistics and anomaly detection by tracking cars. In security, the intelligent monitoring system [3] will raise the alarm if the image captured by camera is anomalous.

Target tracking was proposed firstly by Wax in 1955. CamShift and Particle Filtering are two widely used algorithms in many target tracking techniques because of their excellent performance in real time and robustness.

Bradski proposed the CamShift algorithm [7-10] on basis of MeanShift algorithm [4-6]. CamShift algorithm is characterized by color histogram, analyses the dynamic probability statistics of the features frame by frame and changes the search box adaptively. Therefore, CamShift algorithm is also called self-adaption MeanShift algorithm.

Particle Filtering algorithm [11-12] is based on Bayesian estimation and Monte Carlo method, and it is suitable for the non-linear and non-Gaussian systems presented by any state space model that is approximate to the optimal estimation [16]. It is a great improvement to Kalman Filtering algorithm [13-15].

CamShift and Particle Filtering are well developed on PC. PC shows powerful computing performance, but it has great disadvantage of large size, high cost and power consumption. Compared to PC systems, embedded systems are less powerful in storage space and computing capability, but they are flexible and less consuming. Considering the cost, embedded system is a better choice for target tracking algorithm implementation.

Lots of researchers use DSP, ARM or FPGA to implement target tracking algorithm. Pan Lian [17] proposed a pedestrian tracking system on ARM. Li Yanbin [18] modified Particle Filtering on DSP. Deng Minxi [19] built a real-time target tracking system on DSP. Saeed Ahsan [20] used the parallel structure of FPGA to track the moving objects and proposed the FPGA-based real-time target tracking on a mobile platform. Jung [21] used the adaptive color histograms to implement the FPGA-based real-time target tracking system.

Compared to PC, ARM and DSP, FPGA has a great performance in parallel structure. If there are 100 particles need to be computed, PC will cost plenty of time due to the serial processing pattern. FPGA processes the 100 particles in parallelism and shorten the computing time.

DE2-115 board provides Cyclone IV with enough logic elements and multiplying units, and peripheral units to satisfy the Multimedia IC designs. This paper proposes FPGA-based hardware architecture of CamShift and Particle Filtering algorithms.

II. VISUAL TARGET TRACKING ALGORITHMS

A. CamShift Algorithm

To analyse the continuous dynamic color histograms is the key process of CamShift algorithm. From the histograms, we judge the location of the target. CamShift shows a strong robustness and real-time performance. The algorithm flow chart is shown in Fig.1.

1) Set the tracking object and search box; initialize the procedure.

2) Count the color probability of the target box and generate the color histogram. Compared with RGB color space, HSV color space is less sensitive to the light. We extract the H-component to analyse the color statistics to

weaken the light influence on tracking effect and simplify the computing.

3) Back project the color histogram to the image. Color of the target values more than the background.

4) Compute the zeroth moment and first moment and get the centroid position of the search box. Set (x, y) as the pixel position of the search box and $I(x, y)$ as the back projection of (x, y) . Define the zeroth moment M_{00} and first moment M_{01}, M_{10} as follows:

$$M_{00} = \sum_x \sum_y I(x, y) \tag{1}$$

$$M_{01} = \sum_x \sum_y yI(x, y) \tag{2}$$

$$M_{10} = \sum_x \sum_y xI(x, y) \tag{3}$$

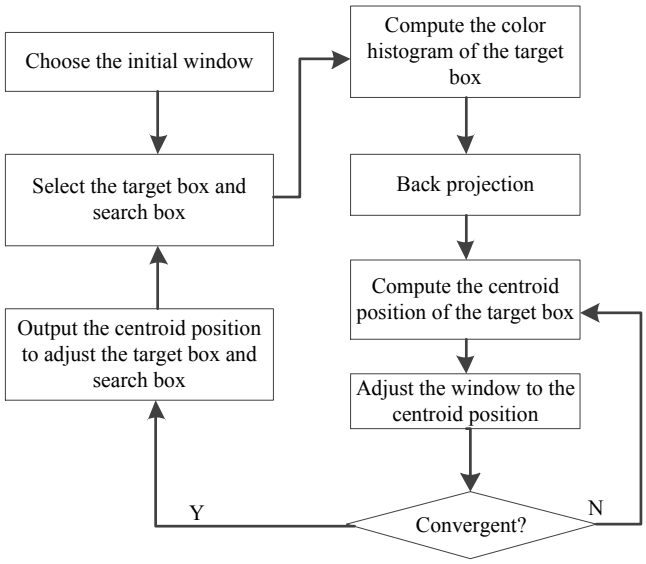

Fig. 1 Flow chart of CamShift algorithm

The centroid position of the search box:

$$(x_k, y_k) = \left(\frac{M_{10}}{M_{00}}, \frac{M_{01}}{M_{00}} \right) \tag{4}$$

5) Adjust the window to the centroid position and judge the convergence. If it is convergent, output the centroid position, adjust the target box and search box and track the next frame.

B. Particle Filtering Algorithm

Particle Filtering is an optimal algorithm based on Bayesian estimation and Monte Carlo method. It uses sequential process to measure data by recursive fashion, so it is unnecessary to store and process the previous data and saves storage space. The algorithm flow chart is shown in Fig.2.

The steps of the Particle Filtering are shown as follows:

1) Initialization: $t = 0$

Collect the particle set $\{ x_0^{(i)}, i = 1,2,...,N \}$ from the prior distribution $p(x_0)$.

2) If $t \geq 1$, $\{x_{t-1}^{(i)}, w_{t-1}^{(i)}\}_{i=1}^{N}$ is the particle set from posteriori distribution at the $t-1$ moment.

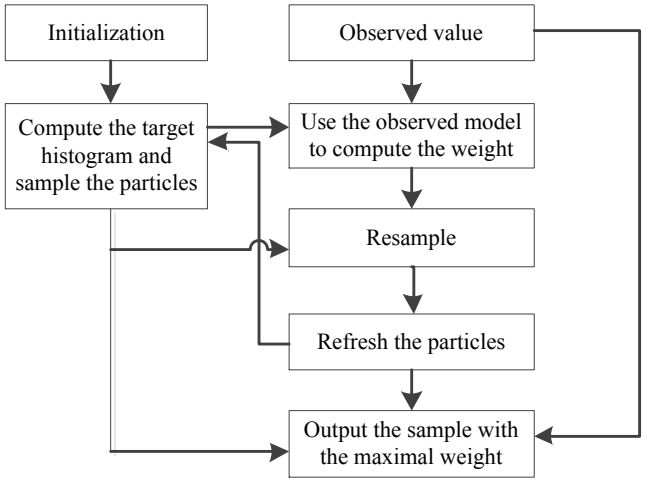

Fig. 2 Flow chart of Particle Filtering algorithm

Importance sample:

Sample the particles $\tilde{x}_t^{(i)} \sim I(\tilde{x}_t^{(i)} \mid x_{t-1}^{(i)}, y_t)$ from the importance distribution $I(x_t \mid x_{t-1}^{(i)}, y_t)$.

At the moment y_t , compute the weight of the particle $\{\tilde{x}_t^{(i)}\}_{i=1}^{N}$:

$$w_t^{(i)} = w_{t-1}^{(i)} \frac{p(y_t \mid \tilde{x}_t^{(i)}) p(\tilde{x}_t^{(i)} \mid x_{t-1})}{I(\tilde{x}_t^{(i)} \mid x_{t-1}, y_t)} \tag{5}$$

Normalize the particle weight:

$$\tilde{w}_t^{(i)} = \frac{w_t^{(i)}}{\sum_{i=1}^{N} w_t^{(j)}} \tag{6}$$

3) Resample

$$\hat{N}_{eff} = \sum_{i=1}^{N} (\tilde{w}_t^{(i)})^2 \tag{7}$$

If $\hat{N}_{eff} < N_{th}$, then define the resample particle as

$$\{x_t^{(i)}, \frac{1}{N}\}_{i=1}^{N} \tag{8}$$

If $\hat{N}_{eff} \geq N_{th}$, then define the resample particle as

$$\{x_t^{(i)}, w_t^{(i)}\}_{i=1}^{N} = \{\tilde{x}_t^{(i)}, \tilde{w}_t^{(i)}\}_{i=1}^{N} \tag{9}$$

4) Output the particle set $\{x_{0:t}^{(i)} : i = 1,2,\ldots,N\}$

Posteriori probability:

$$p(x_{0:t} \mid y_{1:t}) = \hat{p}(x_{o:t} \mid y_{1:t}) = \frac{1}{N}\sum_{i=1}^{N}\delta_{x_{0:t}^{(i)}}(dx_{0:t}) \quad (10)$$

State approximation: $\hat{x}_t = \sum_{i=1}^{N} \tilde{w}_t^{(i)} x_t^{(i)}$

According to the law of large numbers, define the expectation of the function $g_t(x_{0:t})$ as

$$E(g_t(x_{0:t})) = \int g_t(x_{0:t}) p(x_{0:t} \mid y_{1:t}) \approx \frac{1}{N}\sum_{i=1}^{N} g_t(x_{0:t}^{(i)}) \quad (11)$$

At the moment of $t-1$, the posteriori probability $p(x_{t-1} \mid y_{1:t-2})$ of the target state variable is represented by approximate particle set $\{\tilde{x}_{t-1}^{(i)} : i = 1,2,\ldots,N\}$. The weight of the particles is $1/N$. Compute the weight $\tilde{w}_{t-1}^{(i)}$ of the particle $\tilde{x}_{t-1}^{(i)}$. The particles with high accuracy have high weight, while those with big error have low weight. Then, we get the particle set $\{\tilde{x}_{t-1}^{(i)}, \tilde{w}_{t-1}^{(i)}\}_{i=1}^{N}$. Particles with high weight will derive more particles than those with low weight. The derived particles show uniform distribution as well. Therefore, we get the particle set $\{\tilde{x}_{t-1}^{(i)}, 1/N\}_{i=1}^{N}$ to predict next particle set $\{\tilde{x}_t^{(i)}, 1/N\}_{i=1}^{N}$.

III. The Implementation Architecture of the Visual Target Tracking Algorithms

We use the DE2-115 board, camera, VGA display and IR remote controller, and propose the hardware accelerating architecture of CamShift and Particle Filtering algorithms on FPGA.

A. Hardware Implementation of CamShift Algorithm

Combining CamShift with FPGA features, the module partition of the hardware architecture is shown in Fig.3.

CamShift includes RGB to HSV, median filtering and CamShift implementation. CamShift implementation consists of selecting the initial target, color histogram statistics, back projection, computing the zeoroth and first moment, outputting the centroid and displaying the tracking box. Because of the low complexity, CamShift is easy to implement on FPGA. The performance of FPGA implementation of CamShift will be analysed in the next Chapter.

1) RGB to HSV: As mentioned in the previous chapter, HSV color space is little sensitive to the light. To simplify the computation, we extract the H-component to count the color histogram. As shown in Fig.4, RGB to H-component needs division, while FPGA doesn't support it directly, so we use the divider in Quartus II. Each division occupies more than one pixel clocks. It leads to the FPGA timing problem. Hence, we use the flow line to solve this problem. Delay two pixel clocks at the output terminal and output the H-component at each beat.

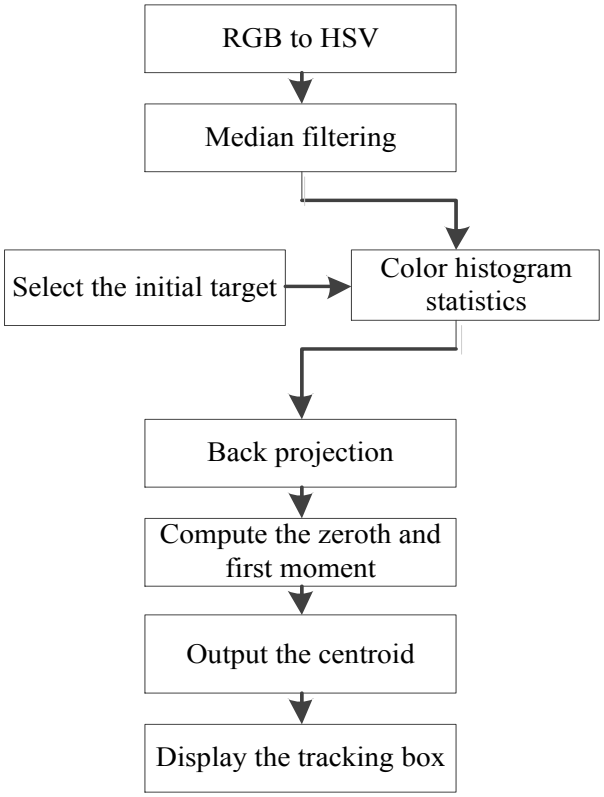

Fig. 3 Module partition of CamShift hardware architecture

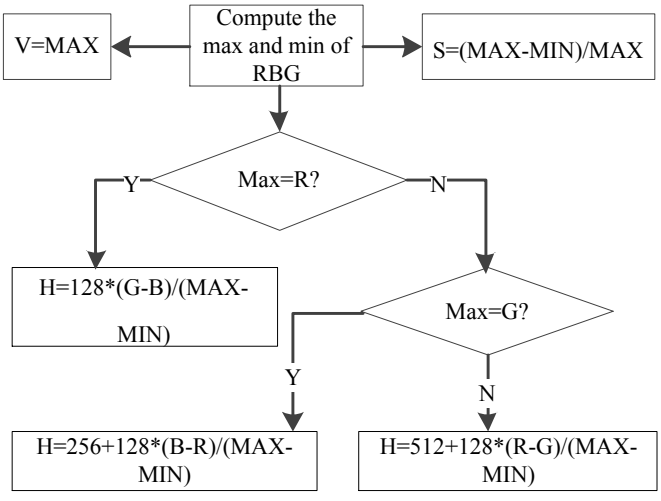

Fig. 4 RGB to HSV

2) Median Filtering: The H-component image is denoised smoothly by median filtering. Output to the input port of the CamShift module.

3) CamShift Algorithm: This module consists of initialization, color histogram statistics, back projection, computing zeroth and first moment, calculating the centroid and displaying the tracking box.

Initialization: Select a target and put the target into the target box.

Color histogram statistics: Put the color histogram of the first frame into RAM. Each frame will be compared with the first frame by searching the color histogram in the RAM in order to judge whether this pixel is the target pixel.

Back projection: The target weights more, non-target weights less. Back projection will help us to locate the target position.

Compute the zeroth and first moment: We use the formulas mentioned in the previous Chapter to compute zeroth and first moment. We use the multipliers in Quartus II to do multiplication.

Output the centroid: We use the formulas mentioned in the previous Chapter to compute the centroid. We use the dividers in Quartus II to do division as well.

Display the tracking box: Set the centroid as the geometric centre of the tracking box and display the tracking box.

We use state machine to implement the controlling and state transition of each sub module. State machine overcomes the inflexible controlling of the pure digital hardware architecture and makes the timing and logic more accurate. The clear structure contributes to debugging and modification.

B. Hardware Implementation of Particle Filtering Algorithm

Combining Particle Filtering with FPGA features, the module partition of the hardware architecture is shown in Fig.5. It includes initialization, target selection, color histogram statistics of the target and each particle, particle weight computation, target capture, resampling and RAM. We use the initialization to clear RAM. Then, set the size and location of the target. The display module is to output the original image and computing result.

1) Feature Model of Particle Filtering: We choose the H-component histogram as the feature model to track the target, shown in Fig.4.

2) Color Histogram Statistics of the Target: If the input image is the first frame, this system generates the color histogram and prepares for the weight computing.

We use the H-component as the RAM address to do read-write operation. The read data equals the written data adding one.

3) Color Histogram Statistics of Each Particle: Particle Filtering computes the posteriori probability distribution of the target state variables by a series of random particles. We use the method shown in Fig.6 to produce random particles (x, y).

Given the coordinate of each particle, we use the method mentioned in the previous section to get the color histogram of each particle.

4) Compute the weight of the particles: Computing the weight of the particles is the key to Particle Filtering, involving lots of multiplication, division, radication and superposition. We can take advantage of the FPGA parallel structure to compute the weight of each particle in parallelism, but we still need compute enough particles to have a great experimental effect. In this paper, we use 128 particles, so it costs lots of hardware consumption. Considering the slot time between each frame, we use the time multiplexing to compute the weight of the particles, as shown in Fig.7.

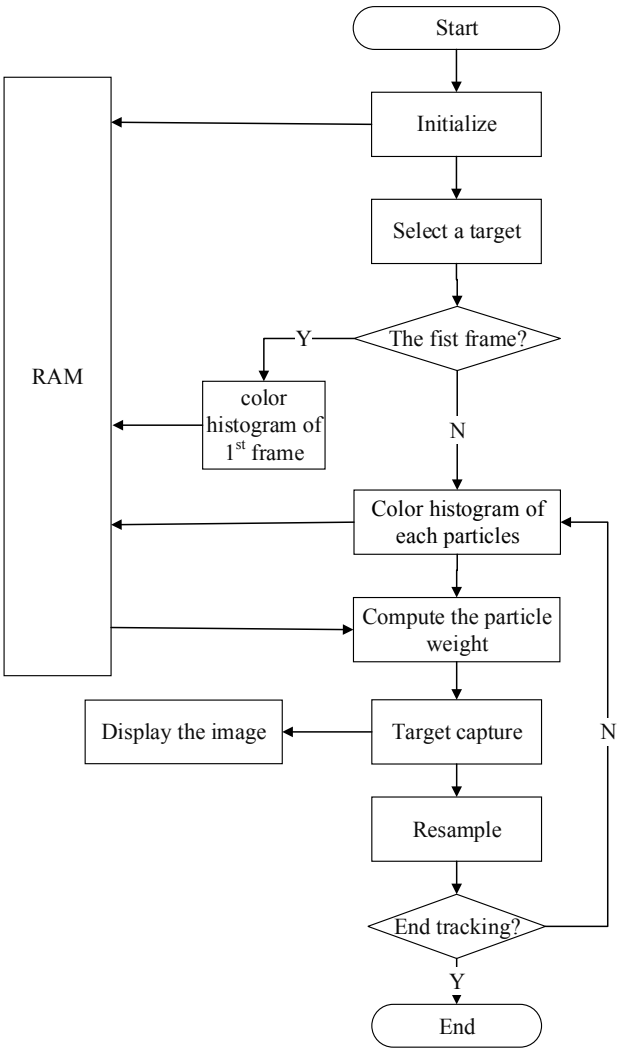

Fig. 5 Module partition of Particle Filtering hardware architecture

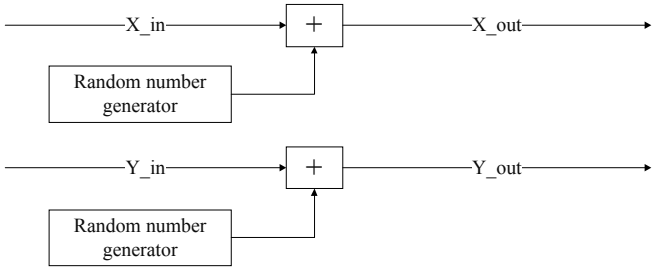

Fig. 6 Particles generator

5) Target Capture: After getting the weight of each particle, we sort them and get the particle with the biggest weight. Choose the particle ordinate as the target ordinate.

6) Resample: Particles are random in the Particle Filtering. Particles with small weight contribute little, but they lead to severe degeneracy. Resampling deletes the particles with small weight and replaces them with large weighted particles.

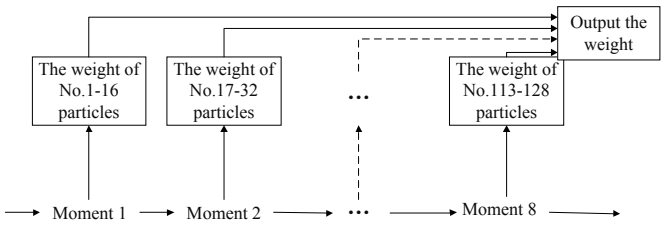

Fig.7 Compute the weight of the particles

7) RAM: We use the RAM on FPGA to store the color histograms of the target and each particle.

IV. EXPERIMENTAL RESULTS

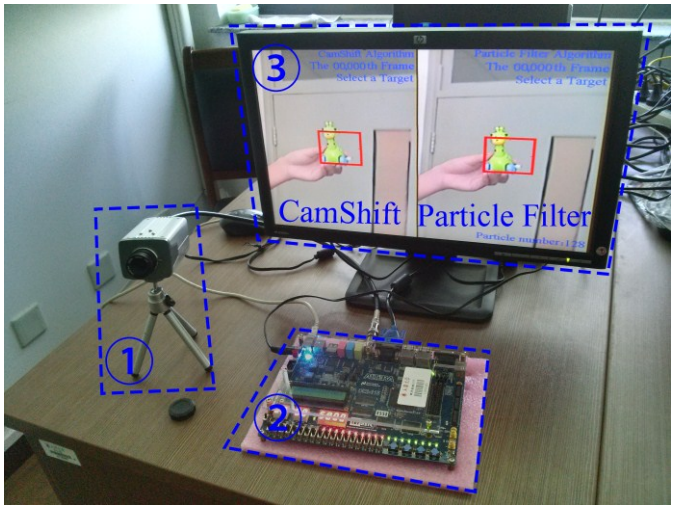

Fig.8 FPGA-based visual target tracking system

As shown in Fig.8, we use the multimedia circuits design resources on the DE2-115 to build this FPGA-based target tracking system. The camera is to collect the video signal. We use the digital-to-analog conversion, Cyclone IV and SDRAM to process the video signal and implement the CamShift and Particle Filtering algorithms. The display is to output the processed video signal. The left part of the display is experimental results of CamShift algorithm and the right part is Particle Filtering algorithm.

A. Qualitative Experimental Results

1) In a simple scenario

In a simple scenario, we set the human face as the target. The experimental effect is shown in Fig.9. First, we select a target and initialize the system. From the 111th, 188th and 272nd frames, CamShift and Particle Filtering algorithms both meet the demand of target tracking. Observe the 188th and 272nd frames carefully. CamShift can track the target but the tracking box shifts partially. Particle Filtering still has a great performance in a simple scenario.

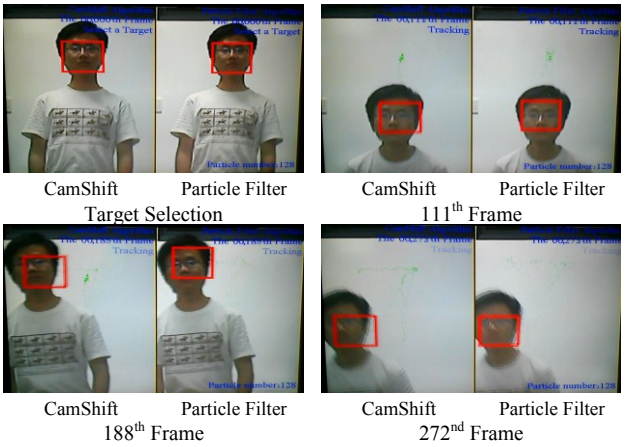

Fig.9 Target tracking effect in a simple scenario

2) In a complex scenario

In a colorful and complex scenario, the tracking effect of CamShift and Particle Filtering algorithms are shown in Fig.10. Both algorithms choose the color as their features, so colorful scenario causes the interference to target tracking. Experiment shows CamShift and Particle Filtering are still effective in complex scenario, but the shift extent of CamShift is more serious than Particle Filtering.

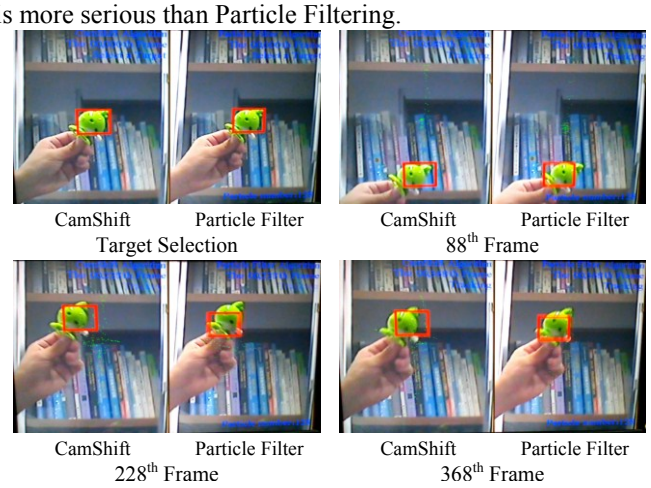

Fig.10 Target tracking effect in a complex scenario

3) With a complex target

As shown in Fig.11, we hold two toys in hand as the target and compare the performance of both algorithms. From the 123rd, 272nd and 340th frames, CamShift loses the target, while Particle Filtering can track the target accurately. When we track the target with only one color, CamShift shows a great performance. If the target includes complex color information, the tracking effect is poor. However, Particle Filtering still shows a strong performance with a complex target.

4) In a dark scenario

Light is an important factor in image processing. The tracking effect in a dark scenario is shown in Fig.12. From 72nd frame, CamShift loses the target. In the followed frames, it still loses the target. When the target is close to the tracking box, it can continue tracking, but the shift extent is serious, shown in 130th frame. From the 263rd frame, it loses the target again. Particle Filtering still works in a dark scenario. Experiment approves that light has a great influence on the CamShift, but Particle Filtering is still robust.

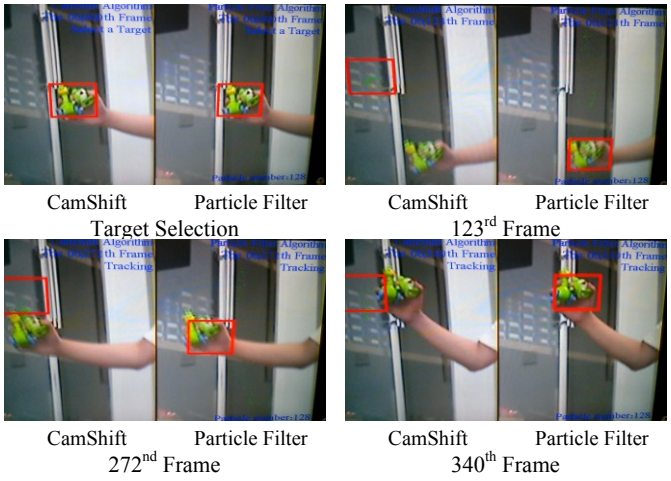

Fig.11 Target tracking effect with a complex target

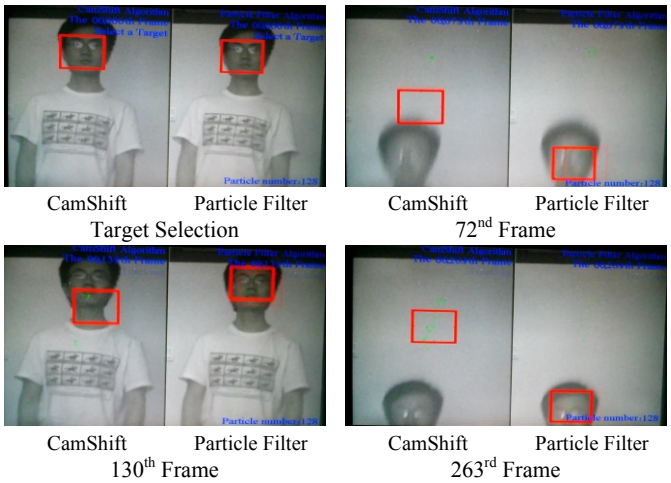

Fig.12 Target tracking effect in a dark scenario

5) With an obstruction

In practical application, if a car is waiting during the red light and a man is crossing the road, the man will occlude the car. Therefore, we need the algorithm to be anti-occluded. As shown in Fig.13, from 296th frame, CamShift loses the target, while Particle Filtering still has a great performance.

6) Velocity of target

A good algorithm should still be effective, when the target moves fast. As shown in Fig.14, from the 128th, 238th and 332nd frames, when the target moves at 20 pixels/frame, both algorithms can track the target. From 458th, 470th, 483rd and 492nd frames, when the target moves at 40 pixels/frame, CamShift loses the target. From 458th frame to 470th frame, the tracking box of the CamShift shifts to the target slowly. When the target moves fast and goes through the tracking box, CamShift is effective again. From the 483rd frame to the 492nd frame, the tracking box does not change the position. Experiment shows the refreshing of CamShift is slow, while Particle Filtering has a better real-time performance.

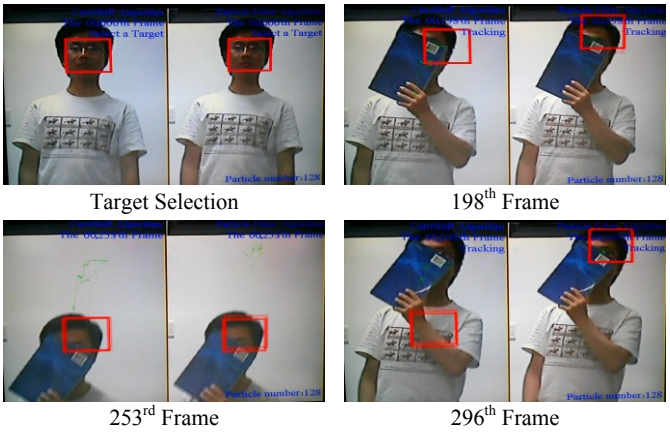

Fig.13 Target tracking effect with an obstruction

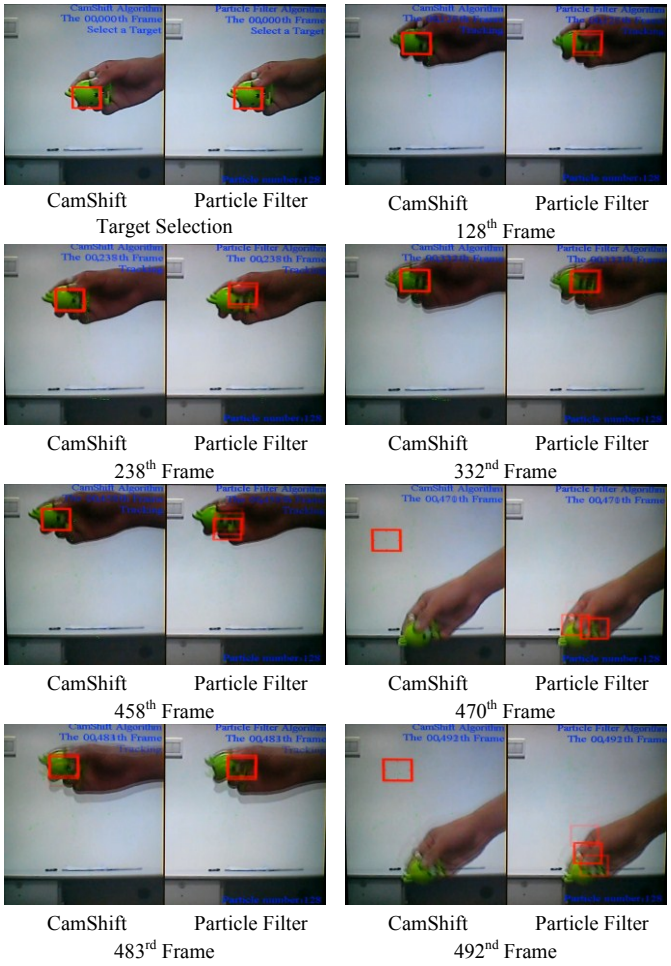

Fig.14 Impact of velocity on the target tracking

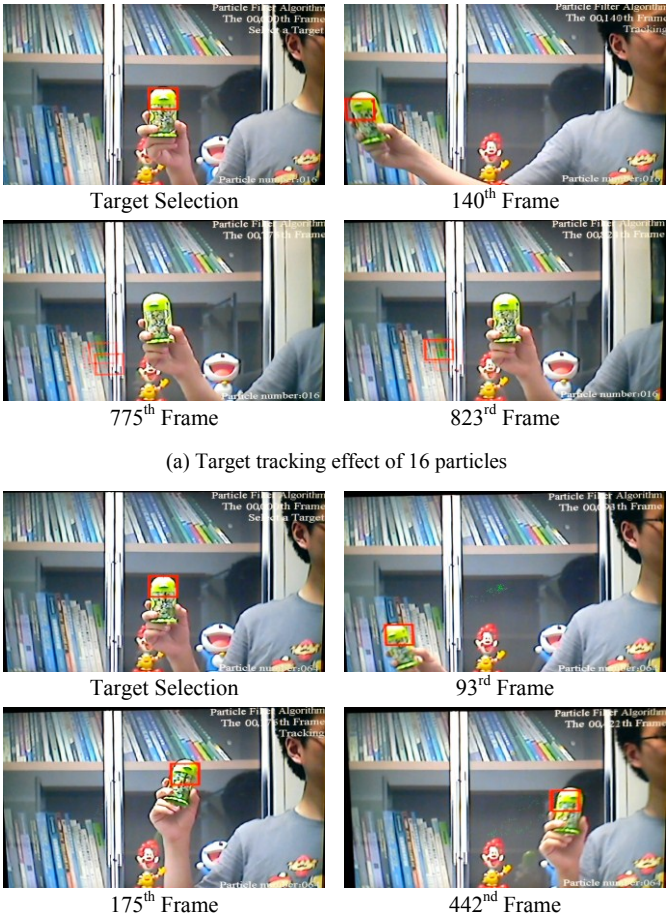

Fig.15 Target tracking effect with different particle numbers

7) *With different particles*

As shown in Fig.15, from the 775th and 823rd frames in Fig.15(a), Particle Filtering with 16 particles has a poor tracking performance. If the particles increase to 64 particles, the accuracy of the tracking effect improves obviously. Experiment shows the number of particles has a great influence on Particle Filtering. If we compute more particles, the tracking effect will be better.

B. *Quantitative Experimental Results*

1) *Influence of particle numbers on the operation time*

Table 1 shows the operation time to compute one frame with different particle numbers. It will cost a long time if we have too many particles to compute on PC, while FPGA costs the same time with different particle numbers. With the increasing of the particle numbers, PC will compute them one by one. However, with the help of FPGA parallel structure, Particle Filtering can compute all particles in parallelism. Experiment shows compared to PC systems, the operation time has been largely accelerated by the hardware architecture. When we have a large number of particles to be computed, this advantage will be obvious.

TABLE I
OPERATION TIME OF PARTICLE FILTERING ALGORITHM ON PC AND FPGA

Particles numbers	PC operation time(s)	FPGA operation time(s)
8	0.0191	0.0167
16	0.0342	0.0167
32	0.0493	0.0167
64	0.1029	0.0167
128	0.1920	0.0167

TABLE II
RESOURCE CONSUMPTION OF FPGA

Resource Consumption	Available	CamShift Used	Particle Filter Used
Combinational LE with no register	114,480	5,703 (4.98%)	46,948 (41.01%)
Sequential LE	114,480	2,832 (2.47%)	6,479 (5.66%)
Combinational LE with a register	114,480	4,245 (3.71%)	15,603 (13.63%)
Dedicated logic registers	114,480	7,077 (6.18%)	22,082 (19.29%)
LABs	7,155	948 (13.25%)	5,654 (79.02%)
M9Ks	432	36 (8.33%)	241 (55.79%)
Block memory bits	3,981,312	243,834 (6.12%)	1,571,413 (39.47%)
Embedded Multiplier 9-bit elements	532	22 (4.14%)	58 (10.90%)
PLLs	4	1 (25%)	1 (25%)

2) Power dissipation and resource consumption of FPGA

As shown in table 2 and table 3, Particle Filtering has large power dissipation and consumes many resources. From the qualitative experiments, the performance of the CamShift is not as good as Particle Filtering. Experiments show CamShift and Particle Filtering algorithms have their own advantages on different conditions. If the power dissipation is limited, CamShift is a better choice. If we have a demand on accurate tracking, Particle Filtering is an efficient and reliable target tracking algorithm.

TABLE III
POWER DISSIPATION OF FPGA

Sort	CamShift Dissipation (mW)	Particle Filter Dissipation (mW)
Total thermal power dissipation	381.51	585.47
Core dynamic thermal power dissipation	87.07	267.73
Core static thermal power dissipation	103.27	108.44
I/O power dissipation	191.18	209.10

V. CONCLUSION

In this paper, we propose a FPGA-based hardware architecture on basis of CamShift and Particle Filtering. Experiments show the proposed system can meet the target tracking demands. On the condition of complex target, dark scenario and fast movement, Particle Filtering has more robust anti-jamming capability. The performance of Particle Filtering can be improved by increasing particles. The FPGA parallel structure can improve the real time performance of Particle Filtering algorithm. Particle Filtering has a great performance at the cost of resource consumption. On condition of limited resource, CamShift algorithm is a better choice. FPGA parallelism processing largely accelerates the operation time. Experiments show the proposed hardware accelerating architecture has a great performance.

REFERENCES

[1] Xiujuan Yao, Xiaole Peng, Yongke Chung. Precise guidance techniques [J]. Laser and Infrared. 2006,36(5):338-340

[2] Betke M., Haritaoglu E., Davis L.S. Real-time vision system for automatic traffic monitoring[J]. Image and Vision Computing. 2000,18(10):781-794

[3] Mohamed F. A., Rama C., Zheng Q. Integrated motion detection and tracking for visual surveillance[A]. Proceeding of the Fourth IEEE International Conference on Computer Vision System[C]. New York: IEEE, 2006:28

[4] Fukunage K and L.D.Hostetler. The estimation of the gradient of a density function with application in pattern recognition[J]. Information Theory. 1975,21(1):32-40

[5] Comaniciu D, Ramesh V, Meer P. Real-Time Tracking of Non-Rigid Objects using Mean Shift[J]. Computer Vision and Pattern Recognition. 2000,2:142-149

[6] Yizong Cheng. Meanshift, mode seeking, and clustering[J]. Pattern Analysis and Machine Intelligence. 1995,17(8):790-799.

[7] Bradski, G.R. Real time face and object tracking as a component of a perceptual user interface[A]. Applications of Computer Vision[C] 1998: 214-219

[8] Collins R T, LIU Yan-Xi. On-Line selection of discriminative tracking features[J]. IEEE Transactions on Pattern Analysis and Machine Intelligence. 2005,27(10): 1631-1643.

[9] Allen J G, Richard Y D and Jin J S. Object tracking using CamShift algorithm and multiple quantized feature spaces[A] Pan-Sydney Area Workshop on Visual Information Processing [C]. Australian: Sydney. 2003: 1-5.

[10] Zhou S K, Chellappa R, Moghaddam B. Visual tracking and recognition using appearance-adaptive models in particle filters[J].IEEE Transactions on Image Processing. 2004,13(11):1491-1506.

[11] Arulampalam M S, Maskell S, and Gordon N. A tutorial on particle filters for online nonlinear/non-Gaussian Bayesian tracking[J]. IEEE Transactions on Signal Processing. 2002,50(2): 174-188.

[12] Doucet A, Gordon N J, Krishnamurthy V. Particle filters for state estimation of jump Markov linear systems[J]. IEEE Transactions on Signal Processing. 2001, 49(3): 613-624.

[13] R.E.Kalman. A New Approach to Linear Filtering and Prediction Problems[J]. Transactions of the ASME--Journal of Basic Engineering. 1960, 82(D):35-45

[14] S. Julier, J. K. Uhlmann, H. F. Durrant-Whyte. A New Method for the Nonlinear Transformation of Means and Covariances in Filters and Estimators[J]. IEEE Transactions on Automatic Control. 2000, 45(3): 477-482

[15] R. P. Wishner, J. A. Tabaczynski, M. Athans. A Comparison of Three Non- linear Filters[J]. Automatica. 1969,5(5):487-496

[16] D. Crisan and A.Doucet. A Survey of convergence results on particle filtering methods for practitioners [J]. IEEE Trans. Speech and Audio Proc., 2002,10(3): 173-185

[17] Pan Lian, and Liu Xiaoming. The study of target tracking based on ARM embedded platform[J]. Journal of Computers, 2012, 7(8): 2015-2023

[18] Li Yan-Bin, Cao Zuo-Liang, Liu Chang-Jie. Particle filter algorithm for target tracking based on DSP[J]. Journal of Optoelectronics Laser, 2009, 20(6): 771-774

[19] Deng Minxi, Guan Qing, Xu Sheng. Intelligent video target tracking system based on DSP[A]. International Conference on Computational Problem-Solving[C], 2010

[20] Saeed, Ahsan, Amin, Adeel, Saleem, Shehzad. FPGA based real-time target tracking on a mobile platform[A]. International Conference on Computational Intelligence and Communication Networks[C], 2010

[21] Jung Uk Cho, Seung, Hun Jin, Xuan, Dai Pham. FPGA-based real-time visual tracking system using adaptive color histograms[A]. International Conference on Robotics and Biomimetics[C], 2007.