High-Speed Target Tracking System Based on a Hierarchical Parallel Vision Processor and Gray-Level LBP Algorithm

Yongxing Yang, Jie Yang, Liyuan Liu, *Member, IEEE*, and Nanjian Wu, *Member, IEEE*

***Abstract*—Visual target tracking has made significant advances in past decades. However, fast and robust vision target tracking systems are still greatly demanded. This paper proposes a novel high-speed target tracking system based on hierarchical parallel vision processor architecture. This system contains three main parts: 1) a CMOS image sensor; 2) a vision processor; and 3) an actuator with two degrees of freedom. The vision processor integrates a pixel-parallel processing element (PE) array, a row-parallel row processor (RP) array, dual-core microprocessor unit (MPU) and motor controller. The PE array and RP array can speed up low-level and middle-level image processing operations by $O(M^2)$ and $O(M)$, respectively. The MPU is responsible for the high-level image processing and the overall chip management. A novel tracking algorithm based on a gray-level local binary pattern descriptor is proposed. The descriptor describes not only local texture feature but also distribution of luminance. The algorithm increases the robustness of the tracking system under low resolution scenery and complex background. It can be carried out by the vision processor with very high efficiency. Experiment results demonstrate that the system can track a fast moving target under complex conditions and the vision processor can achieve over 2000 frames/s processing speed of the target tracking algorithm with 750 × 480 image resolution.**

***Index Terms*—AdaBoost, gray-level local binary pattern (GLLBP), hierarchical, parallel processing, target tracking, vision processor.**

I. Introduction

TRACKING the paths of moving objects is an active field with a long history. People in ancient societies used to track moving prey to hunt and feed their kith and kin, and to track the motion of stars for navigation purposes. Object tracking system recently has become an essential part of our daily lives. It is applied in a wide variety of contexts, such as whale tracking, intelligent video monitoring, industrial automation, traffic monitoring, human–computer interaction, autonomous robot navigation, motion analysis for sports, motion-sensing games, radiosonde-enabled balloon tracking for accurate weather predictions, and cell biology phenomenon [1]–[5].

Simple operations such as background subtraction and segmentation can hardly achieve passable results when the target is tracked under complex scenes or by a moving camera which is controlled by an actuator. As a result, complex algorithms with huge computation are used to achieve robust tracking. However, the response speeds of conventional tracking systems are limited to 30 frames/s due to the bottleneck of serial image data transmission and processing. Obviously, this cannot meet the needs of detecting and tracking high-speed moving object in real scenes.

In order to overcome the restrictions of the conventional tracking systems, many hardware [6]–[21] and algorithms [22]–[30] for high-speed tracking systems have been developed. A parallel processing array called SIMD processing element was proposed to speed up low-level image processing operations [6]–[8], [19]. The 2-D array of processing elements (PEs) operates in pixel-parallel fashion and it can finish simple image processing algorithms within 1 ms, such as edge detection, morphology, and motion detection.

Recently, vision processors based on multiple levels of parallel processors were proposed [20], [21]. These processors consisted of a 2-D PE array, a column of row processors (RPs), and a microprocessor unit (MPU). They partially overcome drawbacks of their predecessors. However, these processors are still unable to meet the needs of high-speed target tracking applications. First, the image capture and processing are carried out serially and extra delay is brought into the tracking system. Next, the processing capacities of these processor are still insufficient for real-time tracking because they can only implement straightforward low-level and some simple middle-level image processing algorithms. They cannot carry out histogram statistics, vector calculations, and machine learning algorithms efficiently, which are inevitable in modern real-time robust visual tracking. Finally, an additional processor is needed to control the actuator directly.

A robust tracking algorithm is the key point to a successful target tracking system. Currently, only several tracking algorithms can be realized by hardware at real time, such as self-window and moment tracking [22], [23]. However, most of them can only be applied to certain scenarios with a clear background and constant illumination. Some other algorithms

Manuscript received July 21, 2015; revised October 9, 2015; accepted December 10, 2015. Date of publication February 19, 2016; date of current version May 15, 2017. This work was supported in part by the National Natural Science Foundation of China under Grant 61234003, Grant 61434004, and Grant 61504141, and in part by the Chinese Academy of Sciences Interdisciplinary Project under Grant KJZD-EW-L11-04. This paper was recommended by Associate Editor B.-F. Wu.

Y. Yang, L. Liu, and N. Wu are with the Institute of Semiconductors, Chinese Academy of Sciences, Beijing 100083, China (e-mail: liuly@semi.ac.cn; nanjian@red.semi.ac.cn).

J. Yang is with the Department of Electrical and Computer Engineering, University of Calgary, Calgary, AB, Canada.

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TSMC.2016.2523907

such as compressive tracking (CT), fragment tracker (Frag), the MIL track, and tracking-learning-detection (TLD) perform better but the computation overhead constrains their speed performance [24]–[27]. Local binary pattern (LBP) is a very efficient texture descriptor for its computational effectivity and tolerance against illumination changes. Many variants of LBP have been proposed for various applications, like tracking and recognition [28]–[30]. However, the LBP of each pixel focuses on the grayscale differences between the center pixel and its neighborhood pixels, the grayscale information of each pixel is ignored. Hence, LBP would fail to track when the resolution of the image is low or the textures of targets and background are similar.

This paper proposes a novel high-speed target tracking system based on a hierarchical parallel vision processor and a gray-level LBP (GLLBP) tracking algorithm. This system mainly contains three parts: 1) a CMOS image sensor; 2) a vision processor; and 3) an actuator. The proposed processor integrates hierarchical parallel array processors that speed up image processing during the whole tracking lifecycle. It contains a pixel-parallel PE array, a row-parallel RP, a dual-core reduced instruction-set computer (RISC) MPU, and a motor controller. The PE array, RP array, and MPU perform low-, middle-, and high-level image processing and chip management, respectively. The pixel-parallel PE array and row-parallel RP array can speed up low-level and middle-level image processing operations by $O(M^2)$ and $O(M)$, respectively. Each RP can randomly access any PE in the same row directly and acquire global information efficiently. The actuator has two degrees of freedom (DOFs): pan and tilt, and makes the camera gaze at the target. A GLLBP tracking algorithm is proposed and realized in this system. This algorithm is more robust than previously proposed high-speed tracking algorithms in aspect of robustness to low resolution scenery and background with similar texture. The algorithm can be carried out by the vision processor with high efficiency. The whole system can realize robust high-speed target tracking in complex scenes.

Rest of this paper is organized as follows. Section II describes the hierarchical parallel vision processor. In Section III, the GLLBP descriptor and tracking algorithm are introduced. Section IV presents the implementation of our tracking system. Section V illustrates the experiment results and correlative analysis. Finally, Section VI concludes this paper and indicates future directions.

II. Hierarchical Parallel Vision Processor

Fig. 1 shows overview of the proposed target tracking system. This system mainly contains a high-speed CMOS image sensor, a vision processor, and an actuator. The image sensor captures the image of the targets at high speed. The processor suppresses the image noise first, then searches the target and extracts its feature by the GLLBP tracking algorithm, next captures the target and finally controls the actuator to track the target. The actuator has two DOFs: pan and tilt. These motors are directly controlled by the vision processor. The proposed system can track high-speed target in a complex environment stably.

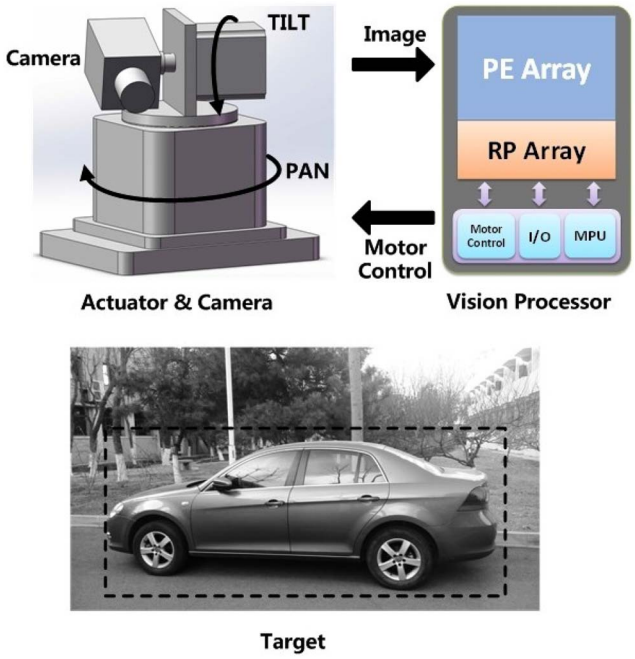

Fig. 1. Overview of high-speed target tracking system.

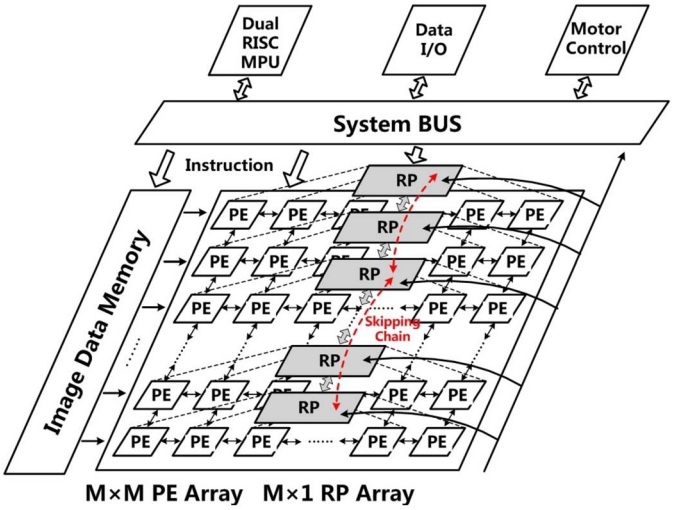

Fig. 2. Architecture of hierarchical parallel vision processor.

A. Architecture of Vision Processor

The architecture of the proposed hierarchical parallel vision processor is shown in Fig. 2. It contains an $M \times M$ single instruction multiple data (SIMD) PE array processor, a row-parallel SIMD $M \times 1$ RP array processor, a dual-core MPU, a motor controller, and data memory. The PE array processor performs pixel-parallel low-level image processing, such as background subtraction, space filtering, and local feature extraction. The PE array processor speeds up the low-level processing by $O(M \times M)$. Each PE element can visit its four nearest-neighbor PEs on east, south, west, and north, and RP of the same row. The RP array processor performs row-parallel middle-level processing to acquire global information. This o1-D SIMD processor can speed up the middle-level image processing by $O(M)$. The dual-core MPU is responsible for

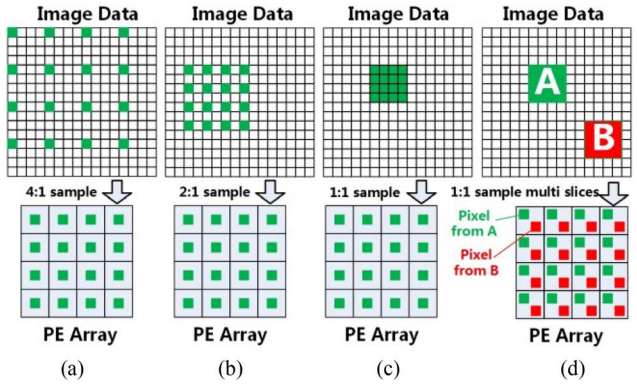

Fig. 3. Mapping scheme between image data and PE array. (a) 4:1 sample. (b) 2:1 sample. (c) and (d) 1:1 sample.

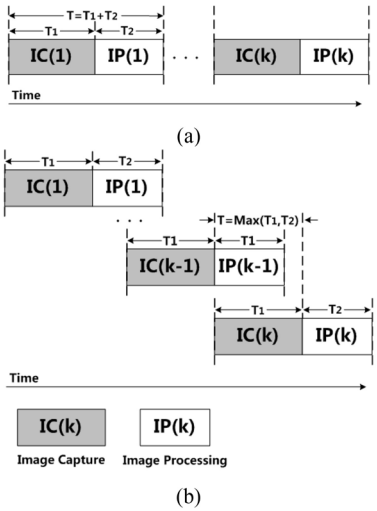

Fig. 4. Frame pipeline. (a) Frame scheme of previous vision processors. (b) Frame pipeline scheme of the proposed hierarchical parallel architecture.

the high-level image processing and the overall chip management. The MPU also controls the motors of the actuator to extend the field of view with the motor control module. This motor control module receives the instruction from MPU and sends pulse to the motor to make the actuator turn toward the desired position.

There are obvious advantages compared with previous vision processors. First, the image resolution is usually larger than the size of PE array, then the PE array can only process one slice of the image at a time. Under such circumstance, PE array can finish the whole image processing by scanning the slices. In our vision processor, there is an efficient and flexible mapping scheme that in each frame the PE array can sample single or multiple image slices with configurable subsample interval. As shown in Fig. 3(a), pixels stored in the image data memory are 4:1 subsampled to the PE array, each PE stores one pixel data from the slice. Similarly, different subsample interval and slice number can be applied, as shown in Fig. 3(b) and (c). In some operations, PE needs to deal with pixels from different slices (for example, slices A and B). Then pixels in slices A and B are sampled to PE array [shown in Fig. 3(d)], and each PE stores two pixel data from slice A and from slice B, respectively.

The number of slices that can be obtained in each frame is mainly determined by the image processing algorithm and PE memory space. Second, the architecture increases the ability of data transmission between the array processors and between the RP array and MPU. Each RP can randomly access any PE in the same row by column index addressing. Likewise, the MPU can randomly access any RP by row index addressing. Thus, the MPU can randomly access any PE. What is more, the image capture and processing can be carried out simultaneously by a frame pipeline scheme, as shown in Fig. 4. When the processor processes the $k-1$th frame image, the data memory captures the kth frame image. Then, the cycle time depends on the maximum of $T1$ and $T2$, as shown in Fig. 4(b). However, the previous vision processors perform image capture and processing alternately. The cycle time is the sum of $T1$ and $T2$, as shown in Fig. 4(a). Therefore, the proposed vision processor can make the tracking system achieve better system-level performance.

TABLE I
SOME FUNCTIONS OF PE ARRAY

Function	Instruction Cycle
$c = a + b$	17
$c = a - b$	18
$c = max(a, b)$	52
$c = min(a, b)$	52
$c = \|a\|$	72
3×3 Gaussian filtering	145
3×3 morphology erosion	19
3×3 morphology dilation	19
$\frac{\partial f}{\partial x}$ or $\frac{\partial f}{\partial y}$	18
$\frac{\partial^2 f}{\partial^2 x}$ or $\frac{\partial^2 f}{\partial^2 y}$	52

B. Circuit Design

The proposed PE circuit consists of a 1-bit arithmetic logical unit (ALU), a piece of local memory, a temporary register, and several multiplexers, as shown in Fig. 5. Each PE element is connected to its four nearest-neighbor PEs on east (E), south (S), west (W), and north (N) to form a 2-D $M \times M$ array. PE element can communicate with the RP of the same row through the PE_Out channel directly. ALU can perform operations such as and, or, addition, and inversion. It takes operands of op1 from 1-bit constant (C), self, and neighbor PEs and op2 from temporary register or 1-bit constant, respectively. The result can be stored in the local memory or temporary register (Temp_reg). PE can complete arithmetic operations of one or two variables with any bit width by repeating the 1-bit ALU operations. Table I lists some common functions that PE can finish and the corresponding instruction cycles, where a, b, and c are 8-bit variables in the PE memory. $\partial f/\partial x$ and $\partial^2 f/\partial^2 x$ are the first and second order of image gradients, respectively.

Fig. 6 shows the schematic of RP. It consists of an 8-bit ALU, a local memory, a register file, and some multiplexers.

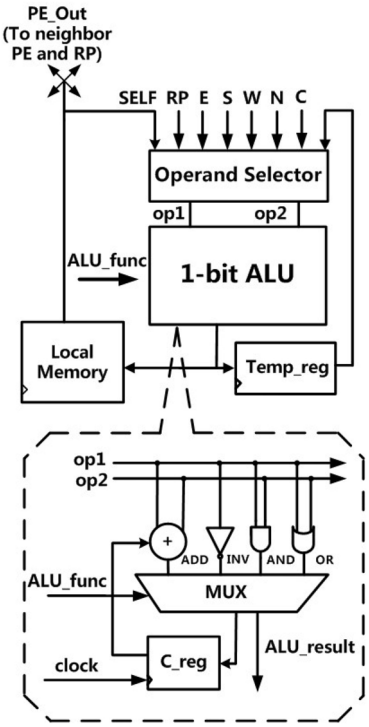

Fig. 5. Schematic of PE circuit.

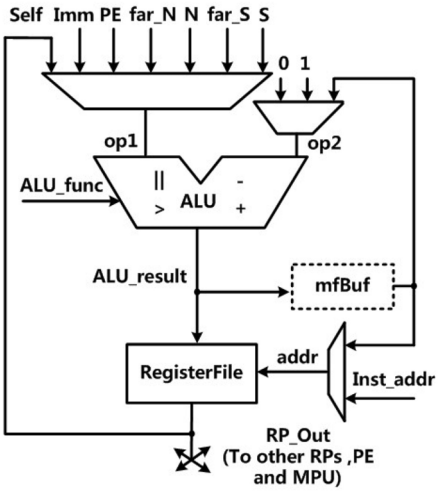

Fig. 6. Schematic of RP.

Each RP can exchange data with its nearest upper (N) and lower neighbor RPs (S), as well as with PEs in the same row directly. Moreover, some RPs are located on a skipping chain (shown with red line in Fig. 2). The RPs can directly visit distant RPs (far_N, far_S) on the same chain and operate their data variables to accelerate some global operations. The ALU can carry on operations such as max/min selection, absolute value calculation, subtraction, and addition. It takes operands of op1 from immediate value (Imm), self, neighbor RPs or PE of the same row, and op2 from buffer (mfBuf) or 1-bit constant, respectively. The results of ALU are stored in the buffer or register file. RP can also perform register index addressing to accelerate middle-level algorithms such as histogram extraction and vector calculation in row parallel manner.

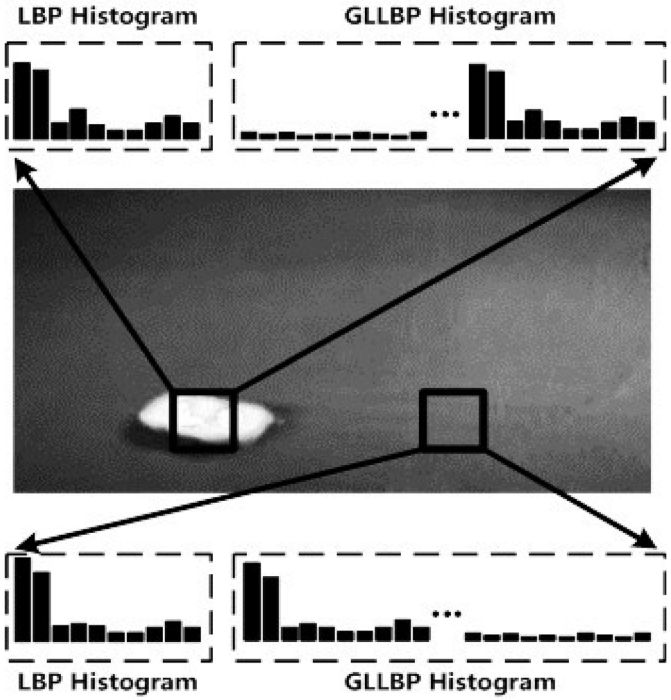

Fig. 7. LBP and GLLBP description.

III. GLLBP Descriptor and Tracking Algorithm

This paper proposes a novel GLLBP descriptor and a corresponding tracking algorithm. They increase the robustness of the tracking system to achieve reliable target tracking. The GLLBP descriptor and tracking algorithm are presented in the following parts.

A. GLLBP Descriptor

LBP is a very efficient local feature descriptor. It has been used in various applications because of its computational effectivity and tolerance against illumination changes [31]. However, because the LBP pattern only focus on the grayscale differences between the center pixel and its neighborhood pixels, it loses the global grayscale information of an image. When the resolution of the image is low or the textures of target and background are similar to each other, tracking algorithms with only texture information are likely to fail. In this case, LBP will degenerate and even cease to be effective. As shown in Fig. 7, the LBP histograms of target and background may be difficult to distinguish. Hence, the descriptor needs extra grayscale information to enhance its robustness.

We propose a novel GLLBP descriptor. Feature of a local region in a monochrome image can be defined as the joint distribution of the grayscale of p image pixels

$$F = f(g_c, g_0, g_1, \ldots, g_{p-1}) \tag{1}$$

where g_c corresponds to the grayscale value of the central pixel and $g_i (i = 0, \ldots, p-1)$ corresponds to the grayscale values of p pixels equally lied on a circle of radius R that form a circularly symmetric neighbor set.

Then we can give (2) by subtracting the grayscale value of the center pixel g_c from the grayscale values of the circularly

symmetric neighborhood

$$F \approx f(g_c, g_0 - g_c, g_1 - g_c, \ldots, g_{p-1} - g_c). \tag{2}$$

Next, based on [31], we assume that the grayscale differences $g_i - g_c$ are independent of center pixel grayscale g_c, which allows us to factorize (2)

$$F \approx f(g_c) f(g_0 - g_c, g_1 - g_c, \ldots, g_{p-1} - g_c) \tag{3}$$

where the distribution $f(g_c)$ describes the overall luminance of the image, and $f(g_0 - g_c, g_1 - g_c, \ldots, g_{p-1} - g_c)$ presents local texture of the image.

In our GLLBP descriptor, LBP describes local texture feature and gray level represents the distribution of luminance. Gray level is generated by dividing the grayscale value of a pixel into different bins. Assuming the number of bins is N, gray level of an 8-bit monochrome image can be calculated as

$$\mathrm{GL}(x, y) = \begin{cases} 0, 0 \le I(x, y) < \dfrac{256}{N} \\ \text{bin}, \dfrac{256}{N} \le I(x, y) < \dfrac{256 \times 2}{N} \\ \ldots\ldots \\ \text{bin} \times (N-1), \dfrac{256 \times (N-1)}{N} \le I(x, y) < 256 \end{cases} \tag{4}$$

where GL(x, y) is the gray level of pixel(x, y), $I(x, y)$ is the grayscale value of pixel(x, y), bin is the step length of GL.

GLLBP can be written as the combination of LBP an GL. Here, the value of GLLBP is equal to the summation of GL and LBP. According to [31], LBP is calculated as

$$\mathrm{LBP}_{P,R} = \begin{cases} \sum_{i=0}^{p-1} s(g_i - g_c) & U \le 2 \\ p + 1 & U > 2 \end{cases} \tag{5}$$

where

$$U = |s(g_{p-1} - g_c) - s(g_0 - g_c)| + \sum_{i=1}^{p-1} |s(g_i - g_c) - s(g_{i-1} - g_c)|$$

$$s(x) = \begin{cases} 1, x \ge 0 \\ 0, x < 0. \end{cases}$$

Assuming that the grayscale information GL is independent of texture information LBP, the number of GLLBP values is $N \times (p+2)$, where N is the number of GL values and $p+2$ is the number of LBP values. On the other hand, GL is from 0 to bin$\times(N-1)$ as shown in (4), and LBP is from 0 to $p+1$. Then, we can know that GLLBP is from 0 to bin $\times\ (N-1) + p + 1$ and the number of GLLBP values is bin $\times\ (N-1) + p + 2$. Thus, we can obtain

$$N \times (p+2) = \text{bin} \times (N-1) + p + 2 \tag{6}$$

and obtain bin $= p + 2$.

Finally, GLLBP is presented as follows:

$$\mathrm{GLLBP}^N_{P,R} = \left[\left(\mathrm{GL}^N_c + \sum_{i=0}^{p-1} s(g_i - g_c) \right) \times \text{flag } U \right] + \left[\left(\mathrm{GL}^N_c + p + 1 \right) \times \overline{\text{flag } U} \right] \tag{7}$$

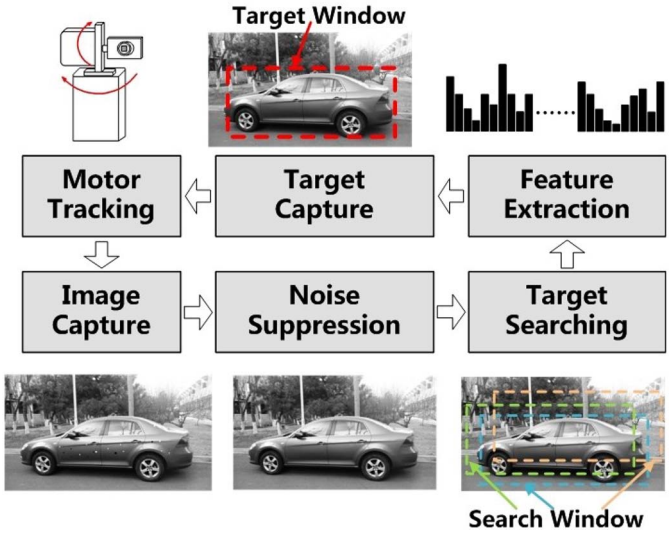

Fig. 8. GLLBP-based high-speed tracking algorithm.

where

$$\mathrm{GL}^N_c = \begin{cases} 0, 0 \le g_c < \dfrac{256}{N} \\ p+2, \dfrac{256}{N} \le g_c < \dfrac{256 \times 2}{N} \\ \ldots\ldots \\ (p+2) \times (N-1), \dfrac{256(N-1)}{N} \le g_c < 256 \end{cases}$$

$$U = |s(g_{p-1} - g_c) - s(g_0 - g_c)| + \sum_{i=1}^{p-1} |s(g_i - g_c) - s(g_{i-1} - g_c)|$$

$$s(x) = \begin{cases} 1, x \ge 0 \\ 0, x < 0 \end{cases} \quad \text{flag } U = \begin{cases} 1, U \le 2 \\ 0, U > 2. \end{cases}$$

If N is set 1, GLLBP is equivalent to LBP. So LBP can be considered as a subset of GLLBP.

Then, we calculate the GLLBP histograms of target and background in Fig. 7. The differences between target and background become obvious. GLLBP descriptor is still tolerant of grayscale changes within 256/N. In high-speed tracking, the illumination changes between adjacent frames will not be violent. By choosing appropriate parameter N, GLLBP can cover the grayscale changes during tracking. In general, N should be larger if luminance distribution is not homogeneous. Smaller N works better when illuminance changes violently. When the texture information is unapparent and LBP becomes invalid, GLLBP describes the target by distribution of grayscale value. Under extreme circumstances that there is no texture information, LBP becomes meaningless but GLLBP can still work as a histogram of grayscale value. With enhanced gray level, GLLBP behaves more robust than LBP.

B. Tracking Algorithm

Fig. 8 shows the flow chart of the tracking algorithm based on the GLLBP feature. It mainly contains operations such as noise suppression, GLLBP feature extraction, and target capture. The details are explained as follows.

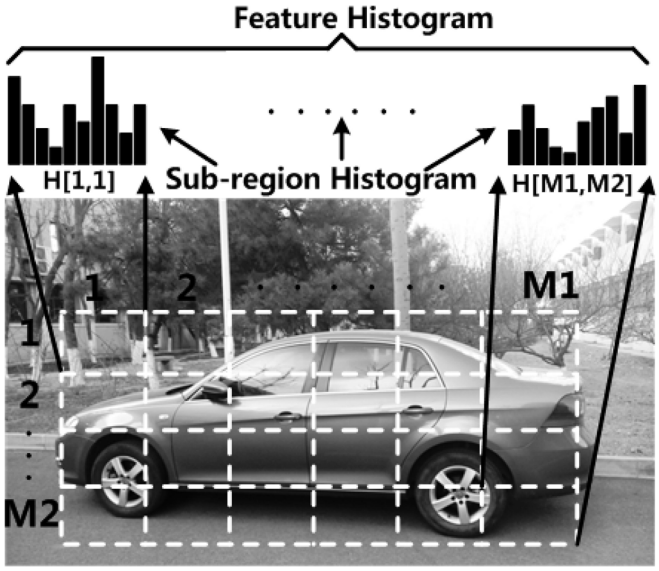

Fig. 9. Extraction of target feature vector.

1) Noise Suppression: In an image, noise is represented as dots whose values change violently. Many methods can be used to suppress the noise. For example,Gaussian filter is an effective method to suppress noise. A 3×3 Gaussian model is implemented as

$$I(x,y) = \frac{1}{8}(I(x+1,y) + I(x-1,y) + 4I(x,y) + I(x,y+1) + I(x,y-1)) \quad (8)$$

where $I(x, y)$ is the grayscale value of pixel (x, y).

2) Feature Extraction: As shown in Fig. 9, assuming that the target region has $N_1 \times N_2$ pixels, the region is first divided into $M_1 \times M_2$ subregions. The size of each subregion is $Q_1 \times Q_2 (Q_1 = N_1/M_1, Q_2 = N_2/M_2)$. The GLLBP histograms of these $M_1 \times M_2$ subregions are then concatenated to form a global feature of the target.

3) Target Searching: In high-speed tracking, the difference between two successive frames will be limited to a few pixels, even with rotation and scale changes. Hence, the search windows are just surrounding the predicted target location, as shown in Fig. 8. The global GLLBP features of the search windows are generated and compared with the feature of the target.

As shown in Fig. 10, there are two searching modes in our tracking algorithm: 1) tracking mode and 2) seeking mode. Usually, the algorithm is under tracking mode; the search windows surround the predicted target location so that the target is within the search region (region which contains all the search windows). When the target moves out of our view (suddenly speeding up or large occlusion), the algorithm turns to seeking mode. In this case, the search region become larger and larger with time until the target is captured again.

4) Target Capture: A classifier is used to identify the target as shown in (9). It combines K weak classifiers $g(i)$ with a specific weight $\alpha(i)$ to form a strong classifier

$$C = \sum_{i=0}^{K-1} \alpha(i) \times g(\mathrm{RV}(i) - V(i)). \quad (9)$$

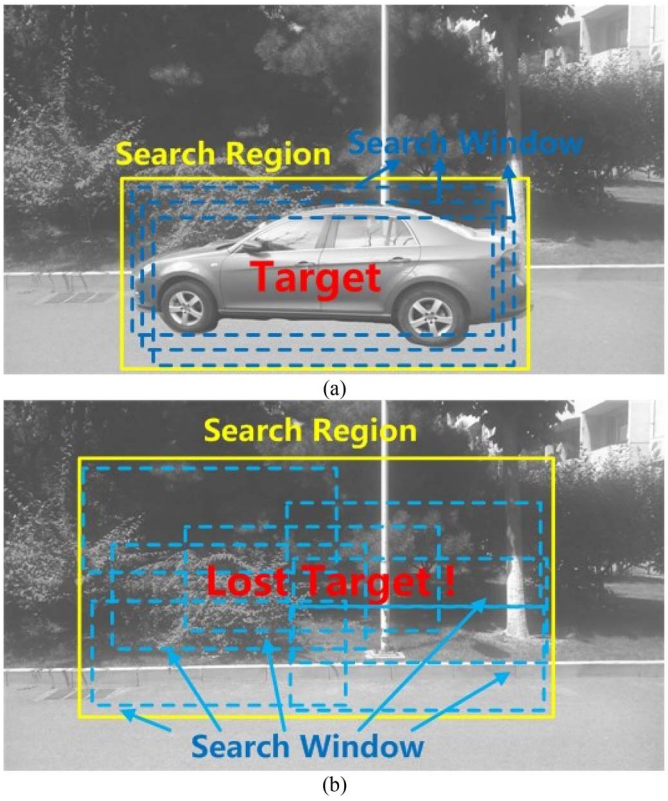

Fig. 10. Two searching methods. (a) Tracking mode. (b) Seeking mode.

The weak classifier $g(x)$ consists of a feature $V(i)$ which corresponds to a single GLLBP histogram bin and a reference vector $\mathrm{RV}(i)$ corresponding to $V(i)$. A threshold T is set to classify the candidate window: if C is smaller than T, then there is a target in this window; otherwise, this candidate window is background. Finally, the search window with the smallest C value is updated as the target window during tracking. (This updating operation only happens when the smallest C is smaller than the threshold T. If the smallest C is larger than T, it means that the target is lost and the search mode of next frame will turn to seeking mode.) The cascade classifier C varies for different applications.

a) Tracking without prior learning: In applications without prior knowledge, the classification (target or background) in every frame can only depend on the features of the past frames. In this case, $\alpha(i)$ is set 1 and $g(x)$ is an absolute operation $g(x) = |x|$. The classifier C in (9) can be written as

$$C = \sum_{i=0}^{K-1} |\mathrm{RV}(i) - V(i)| \quad (10)$$

where V is the GLLBP histogram of the search window and RV is the target window's histogram. So the classifier C represents the similarity of the search window and target. The search window with the smallest C value is updated as the

target window during tracking. The GLLBP feature V with the smallest C value is updated as the reference vector RV. The reference vector RV is updated every frame to adapt to affine and scale changes of the object.

b) Tracking with learning GLLBP bins: In applications where enough images could be got in advance, the parameters of the target are trained before tracking. In this case, the threshold RV(i) and the associated weight for each weak classifier $\alpha(i)$ are obtained with AdaBoost training algorithms [32], [33], and the feature $V(i)$ corresponds to a single GLLBP histogram bin. The weak classifier $g(x)$ here is a sign function as

$$g(\mathrm{RV}(i) - V(i)) = \begin{cases} 1 & \mathrm{RV}(i) - V(i) \geq 0 \\ -1 & \text{otherwise.} \end{cases} \tag{11}$$

Then, the classifier C in (9) can be written as

$$C = \sum_{i=0}^{K-1} \alpha(i) \times k(i) \tag{12}$$

where

$$k(i) = \begin{cases} 1 & V(i) \leq \mathrm{RV}(i) \\ -1 & \text{otherwise.} \end{cases}$$

The value C of search window shows the classification result: target or not. If C is smaller, the search window is more likely to be the target. Then, the search window with the smallest C value is regarded as the target window during tracking.

The choice of target capture method depends on the application scenery. If there is no information of the target before tracking, method *a* is the only choice. Method *b* can offer a more accurate tracking result with offline AdaBoost training. Comparing to method *a*, there is little accumulated error in learning method.

5) Motor Tracking: The target position of next frame is roughly predicted as (13) for the coming target searching

$$\begin{cases} x_{t+1}^P = x_t + dx_t \\ y_{t+1}^P = y_t + dy_t \end{cases} \tag{13}$$

where $dx_t = x_t - x_{t-1}$, $dy_t = y_t - y_{t-1}$, for $t \geq 0$; (x_{t+1}^P, y_{t+1}^P) is the predicted target position of frame $t+1$; (x_t, y_t) is the calculated target location of frame t from target capture operation; (x_0, y_0) is the initial target position, and $(x_{-1}, y_{-1}) = (x_0, y_0)$.

Then, directions and angles for motor rotation are calculated as

$$\begin{cases} P_{\mathrm{STEP}} = \dfrac{x_t - x_c}{\mathrm{STEP}} \\ T_{\mathrm{STEP}} = \dfrac{y_t - y_c}{\mathrm{STEP}} \end{cases} \tag{14}$$

where the absolute value of P_{STEP} corresponds to the angle of pan rotation and the sign presents the direction; the absolute value of T_{STEP} corresponds to the angle of tilt rotation and the sign presents the direction; (x_t, y_t) is the target location of frame t; (x_c, y_c) is the location of image center; and STEP is the motor rotation angle corresponding to one pixel. Then, the processor sends control pulse to motors and makes the camera gaze at the target.

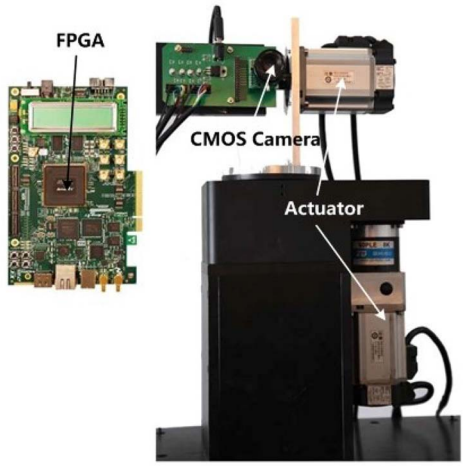

Fig. 11. Photograph of tracking system.

TABLE II
PARAMETERS OF IMAGE SENSOR

Image Sensor	MT9V022
Optical Format	1/3-inch
Active Pixels	752H×480V
Shutter Type	Global Shutter
Maximum Data Rate	26.6MPS
Window Size	128×128
Dynamic Range	>60dB
Frame Rate	60fps (full resolution)
	120fps(128×128)

IV. System Implementation

A. Hardware Implementation

The proposed target tracking system is shown in Fig. 11. It mainly contains three parts: 1) a CMOS image sensor; 2) a vision processor; and 3) an actuator. The information of used CMOS image sensor is listed in Table II. In target tracking applications, valuable target region is about hundreds to thousands pixels and most of the other pixels are useless background. Based on the benchmark sequences commonly used in visual tracking [34], we find that target with 128×128 resolution is enough for most tracking task. So the 128×128 pixels region-of-interest (ROI) after sampling is transmitted to the processor; the sampling rate is configurable and updated during tracking.

In order to evaluate the proposed system, the hierarchical parallel vision processor and other necessary blocks were implemented on an field programmable gate array (FPGA) device. The information about consumed FPGA hardware resource is listed in Table III. In this prototype, it consists of a 128×128 PE array, a 128 RP array, and an MPU; the operating frequency is 100 MHz. To make a deeper analysis, we compiled the vision processor on the same FPGA, in which RP array can only access the easternmost column of PE. We found that the logic resource usage increased about 4.6%. The extra resource usage is some

TABLE III
FPGA RESOURCE ULTILIZATION

FPGA device	ArriaV 5AGXFB3H4F35C4N
Logic utilization (in ALMs)	72,089/ 136,880 (53 %)
Total registers	67023
Total pins	41/ 656 (6 %)
Total virtual pins	0
Total block memory bits	2,350,280/ 17,674,240 (13 %)
Total DSP Blocks	0 / 1,045 (0 %)
Total PLLs	2 / 36 (6 %)
Total DLLs	0 / 4 (0 %)

TABLE IV
PARAMETERS OF ALGORITHM

Parameter	Value	Description
N	4	parameter of GLLBP
P	8	parameter of GLLBP
R	1	parameter of GLLBP
M×M	128×128	Size of ROI image
$N_1 \times N_2$	96×96	Size of target region
$M_1 \times M_2$	6×6	Number of sub-region
$Q_1 \times Q_2$	16×16	Size of sub-region

logic gates introduced to select signal sources. Considering the gained performance of the architecture, we think the additional logic resource is worthwhile. The FPGA device is linked to a personal computer (PC) through the Ethernet interface. Instructions for hierarchical parallel processor and RISC MPU were first developed and compiled on PC, and then sent to memories on FPGA device. The PC was also responsible for video monitor during test. The motor control port is directly connected to the driver of the actuator.

The FPGA board, image sensor, and a lens are mounted on the mobile platform of the actuator. The actuator has two DOFs around the horizontal and vertical axis; each axis has a servo motor (200 W). To meet the response speed of high-speed target tracking, we choose motors with short response time and compact actuators.

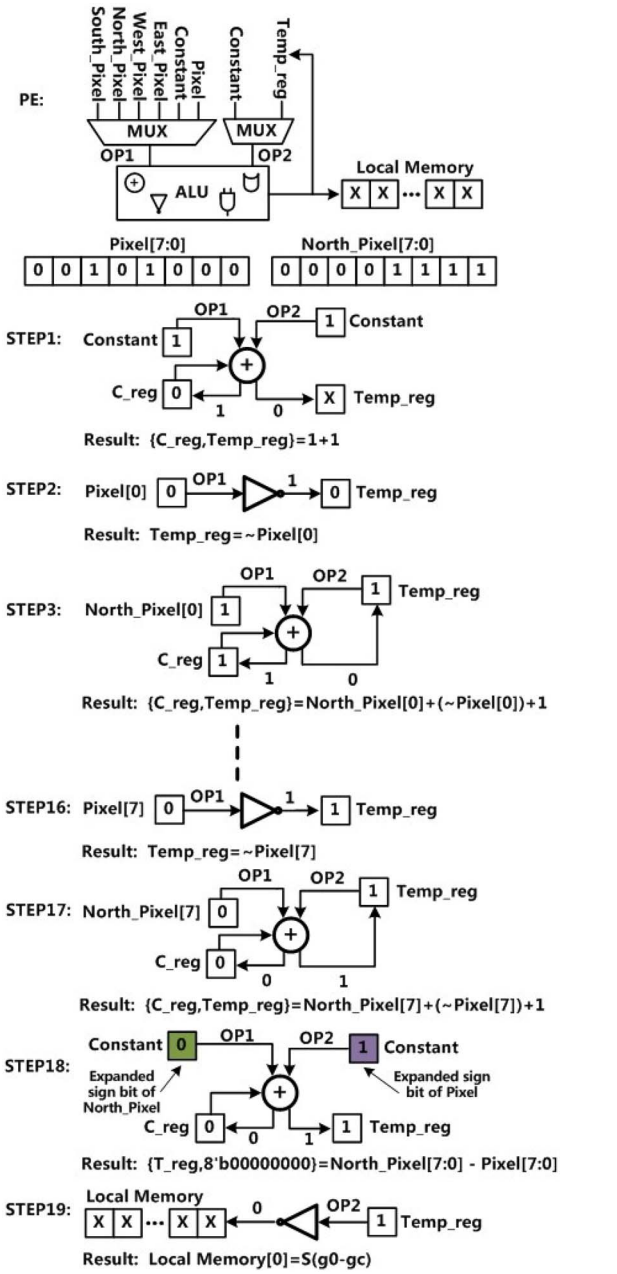

Fig. 12. PE operations of $s(g_0 - g_c)$, assuming that center pixel is 40 (101000) and north pixel is 15 (1111), respectively.

B. Tracking Algorithm Implementation

The proposed tracking algorithm mainly contains operations, such as noise suppression, GLLBP calculation, feature extraction, target capture, and motor tracking. The main parameters are shown in Table IV. The noise suppression and GLLBP calculation only need the grayscale values of the calculated pixel and its surrounding pixels. These operations can be carried out by PE array with pixel parallelism. RP array can extract GLLBP histogram and calculate the classifier C in row-parallel way. The MPU captures the target position and controls the motors tracking.

GLLBP calculation is carried out by PE array in SIMD manner as (7). The details are as follows.

1) *Calculate the Sign Flag of the Center Pixel and Its Neighbor Pixels ($s(g_i - g_c)$, $i \in [0,7]$):* The sign flag can be acquired by judging the sign bit of the difference $(g_i - g_c)$. The difference can be obtained by adding the 2's complements of g_i and $-g_c$, which can be finished by addition and inversion operations. Fig. 12 shows the process of calculating the sign flag of the center pixel and the north pixel $s(g_0 - g_c)$. Other sign flags can be calculated similarly. These flags are added to get the sum $\sum s(g_i - g_c)$.
2) *Calculate U:* This step can be divided into two phases. First, the XOR value of adjacent $s(g_i - g_c)$ is calculated. [$s(g_0 - g_c)$ and $s(g_7 - g_c)$ are also adjacent.] Then, these XOR values are added. The XOR operation can be implemented by addition and eliminating the carry bit.

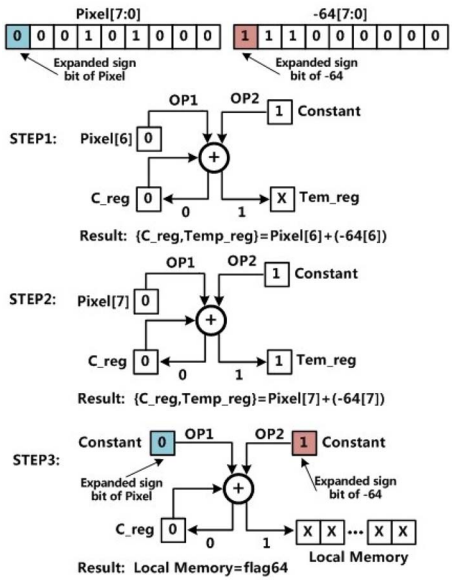

Fig. 13. PE operation of flag 64, assuming that center pixel is 40 (101000).

3) *Calculate Flag U:* Compare the difference between U and 3. If U is not bigger than 2, flag U is set 1.
4) *Calculate GL_c^4:* Compare the pixel value with 64, 128, and 192 and get corresponding flag_{64}, flag_{128}, and flag_{192}, respectively. Similar to $s(g_i - g_c)$ calculation, this compare operation can be carried on by judging the sign bit of the difference. The process can be simplified because the subtrahend is constant value (64, 128, 192). For example, the binary form of -64 is 111000000. The last six bits of -64 have no effect on the sign bit of the difference, so the compare operation starts from the seventh bit, as shown in Fig. 13. In this way, time of flag_{64} operations decreases from 16 to 3 cycles. Then, we can acquire GL_c^4 by several logic operations and addition as

$$\text{GL}_c^4 = \left(\overline{\text{flag}_{64}}\ \&\ \text{flag}_{128}\right) \times 10 + \left(\overline{\text{flag}_{128}}\ \&\ \text{flag}_{192}\right) \times 20 + \overline{\text{flag}_{192}} \times 30 \quad (15)$$

where

$$\text{flag}_X = \begin{cases} 0, g_c \geq X \\ 1, g_c < X. \end{cases}$$

5) *Calculate $GLLBP_{8,1}^4$ as (7):* RP extracts the histogram of GLLBP as a feature vector in a row-parallel way. The 96×96 target region is divided into 6×6 subregions. The GLLBP histogram of each subregion is calculated and the 36 histograms are concatenated to a global vector, as shown in Fig. 9. The GLLBP histogram calculation of a subregion can be divided into two phases: a) GLLBP histogram by row (phase A) and b) sum of rows (phase B). The process is shown in Fig. 14. Although only RPs in one subregion is drown for clarity, in fact, the same operation is simultaneously executed on whole RP array in SIMD manner to accelerate the histogram statistics roughly by $O(M)$.

The classifier C is also calculated in RP. We can reformulate (9) as (14) so that the computation is finished in RP in

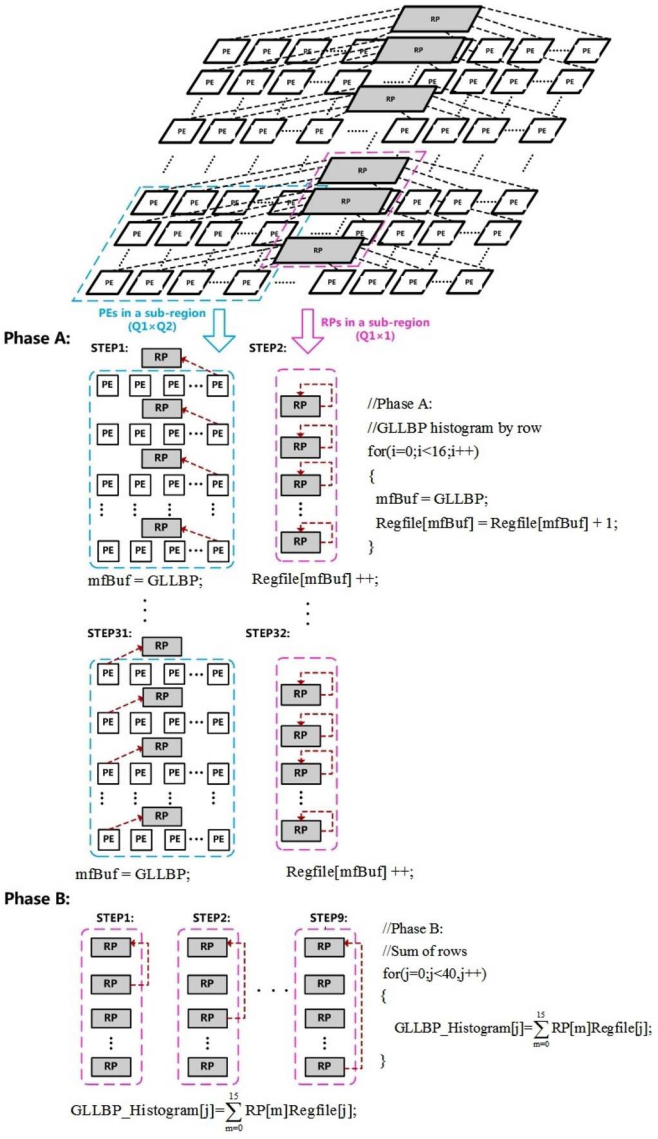

Fig. 14. GLLBP histogram calculation of a subregion in RP.

distributed parallel. In (16), M is the number of RP, and t indicates the number of weak classifier computed in each RP. The corresponding parameters RV and α are stored in RP memory in advance. Row-sum can be calculated simultaneously in RP after GLLBP histogram is computed. The value of C can be acquired by adding these row_sums with the help of RP skipping chan. The whole C calculation can be speeded up roughly by an order of M

$$C = \sum_{j=0}^{M-1} \text{row\_sum}_j \quad (16)$$

where

$$\text{row\_sum}_j = \sum_{i=j \bullet t}^{(j+1) \bullet t-1} \alpha(i) \bullet g(\text{RV}(i) - V(i)).$$

V. Experiment Results

This section presents the system performance of the target tracking system proposed in the previous section.

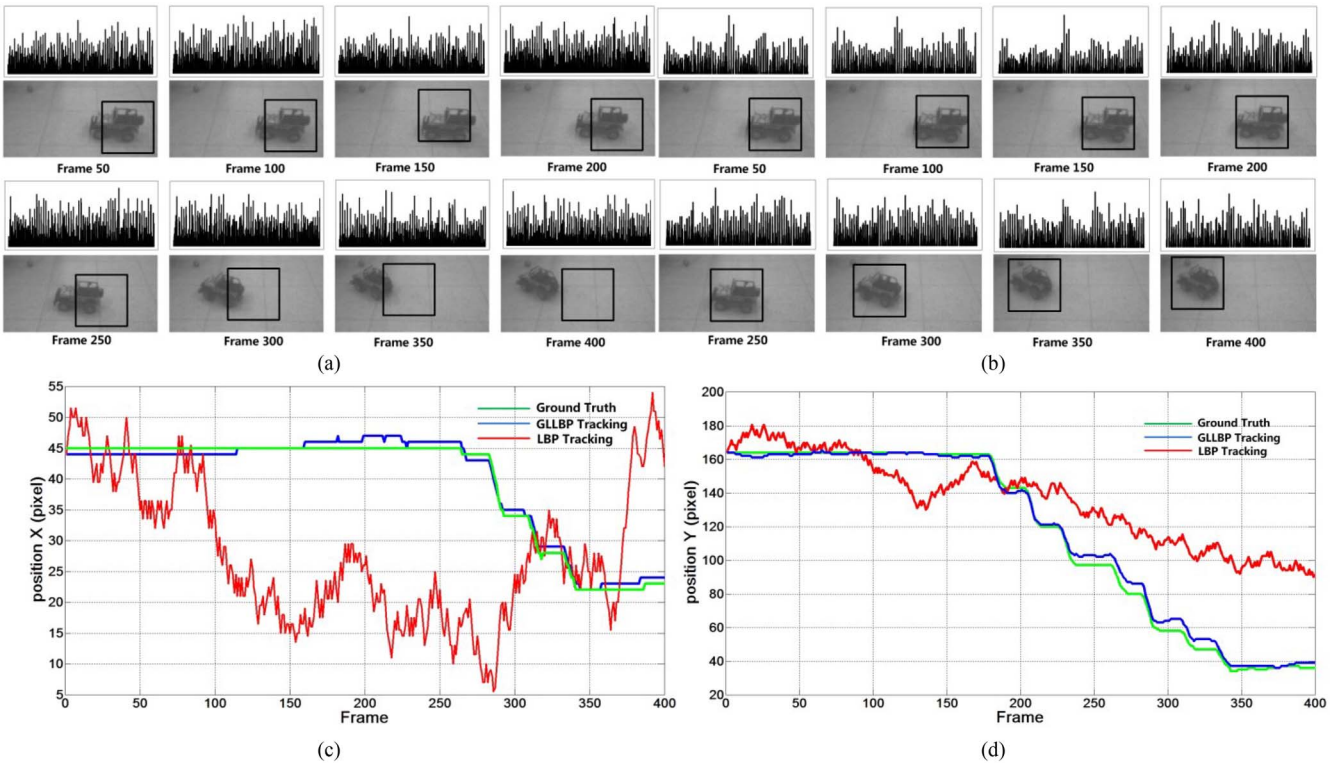

Fig. 15. Test result of tracking algorithm. (The target keeps still from frame 1 to 179 and begins to run from frame 180.) Calculated position and feature histogram of (a) LBP tracking and (b) GLLBP tracking. (c) x coordinate of desired (green), LBP tracking (red), and GLLBP tracking (blue) target center. (d) y coordinate of desired (green), LBP tracking (red), and GLLBP tracking (blue) target center.

First, experiments were carried out to evaluate the basic performance of the tracking algorithm. The same frames of test scenery and similar parameters (P, R, M_1, M_2, N_1, N_2, Q_1, Q_2) are used to test the GLLBP and LBP tracking algorithms. The result is shown in Fig. 15, where the black rectangle represents the calculated target position. As shown in the result, the calculated position of target center by LBP tracking (red line) is gradually drift away from the ground truth (green line). However, the classifier C does not change violently. This means that the LBP feature cannot describe the target precisely. Namely, in the "view" of LBP, the target and background become similar. In GLLBP tracking (blue line), the difference between target and background is more obvious and the tracking is stable.

Then, we evaluate our tracking algorithm with four state-of-the-art methods on six challenging sequences which are publicly available. The four tracking algorithms we compare with are CT, Frag, the MIL track, and TLD. For fair comparison, we use the source or binary codes provided by the authors with tuned parameters for the best performance. Some screenshots of the tracking result are shown in Fig. 16. It can been seen that our tracking algorithm achieves comparative or better performance in most sequences. Furthermore, our tracker runs faster than all of the algorithms with the help of our vision processor. In detail, for the David indoor and car sequence shown in Fig. 16(a) and (e), the illumination and pose of the object both change gradually. The MIL Track, TLD, CT, and our method perform well on this sequence. The object in the Kitesurf and Biker sequence [Fig. 16(c) and (d)] undergoes complicated motion with large degrees out of plane rotation. Only our method and CT perform satisfying in the two sequences; other methods either lose the target or track with large location error. Our algorithm does not work well in Bolt because the target is too small to get enough feature to distinguish with other athletes.

The breakdown of time and computation for the proposed vision processor performing the tracking algorithm are shown in Fig. 17. Thanks to the pixel-parallel PE array, 144 million operations are finished by consuming only 88 μs. Histogram operations are performed by the RP array. It takes 233 μs to finish about 3 million operations. The MPU is used to control the whole system and issue some of the SIMD instructions for the PE array and the RP array. It consumes 26 μs. The total processing speed reaches 2800 frames/s (347 μs). Table V shows the execution time of some kernel functions of the algorithm. A 16×16 patch histogram is finished with 16 RPs and their corresponding 16×16 PEs. It takes 128 PE and 376 RP instruction cycles, respectively. The C value of method a can be finished in 72 RP instruction cycles. Motor control operation costs 300 RISC instruction cycles.

Next, we made some experiments of mechanical tracking of a moving car obtained by controlling the two DOF actuator. Constrained by the limited experimental space, it is difficult to track real vehicle with our system. So we used models to estimate our tracking systems. Fig. 18 shows the environment setup for the experiment. In this experiment, a jeep model runsin front of the tracking system at a speed of 10 km/h; a land rover moves around the jeep as an obstruction.

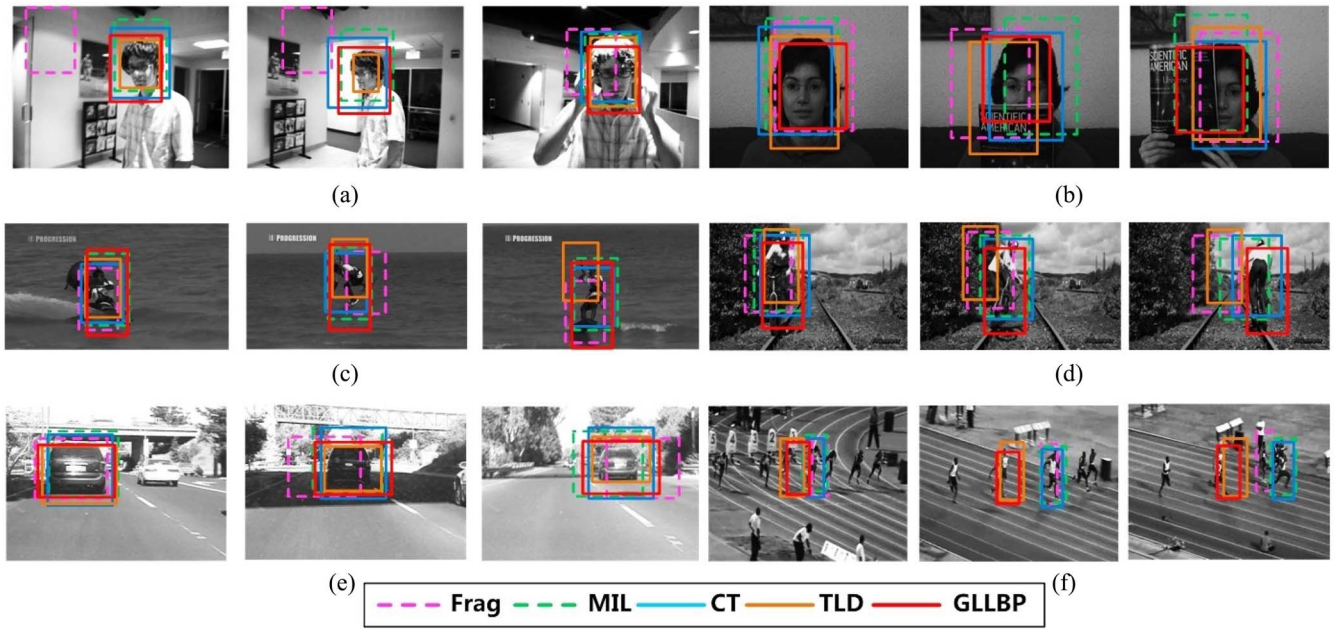

Fig. 16. Screenshots of tracking result highlighting instances of (a) and (e) scale and illumination change, (b) heavy occlusion, (c) out of plane rotation, (d) abrupt motion, and (f) little target with occlusion.

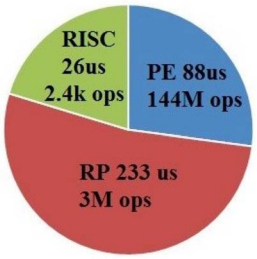

Fig. 17. Breakdown of time and computation.

TABLE V
EXECUTION TIME OF ALGORITHM

Function	Instruction Cycle	Time(microsecond)
$GLLBP^4_{8,1}$	571	5.7
16×16 histogram	504	5
Classifier C of method *a*	72	0.7
Classifier C of method *b*	90	0.9
Motor control	300	3

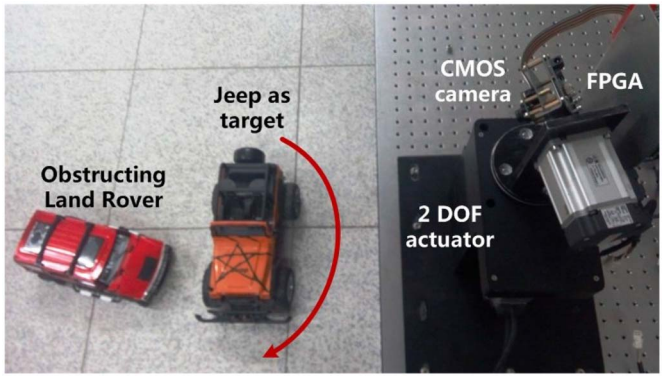

Fig. 18. Environment setup for experiment.

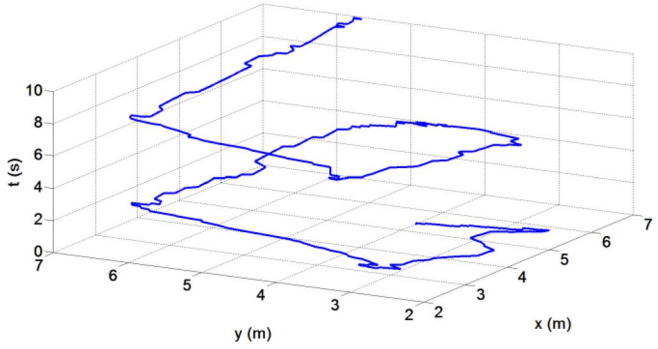

Fig. 19. Trajectory of target.

Fig. 19 gives the trajectory of the target during the experiment. The collection of data points clearly exhibits the motion of the jeep as the target.

Fig. 20 shows the result of the target tracking. The actuator tracks the jeep at the center of view, as shown in the photos. For the target object, Fig. 20(c) and (d) shows the x and y coordinate valves of its center positions over 10 s. It can be seen that the center position of the target image was always controlled around pixel position (64, 64), the center of the camera view. The small deviations were caused by the limitation of motor potential ability. The trajectories of the actuator are also given in Fig. 20(b). The tilt (blue line) and pan (red line) angles were periodically changed about every 6 s. This tendency corresponds to the circle movement

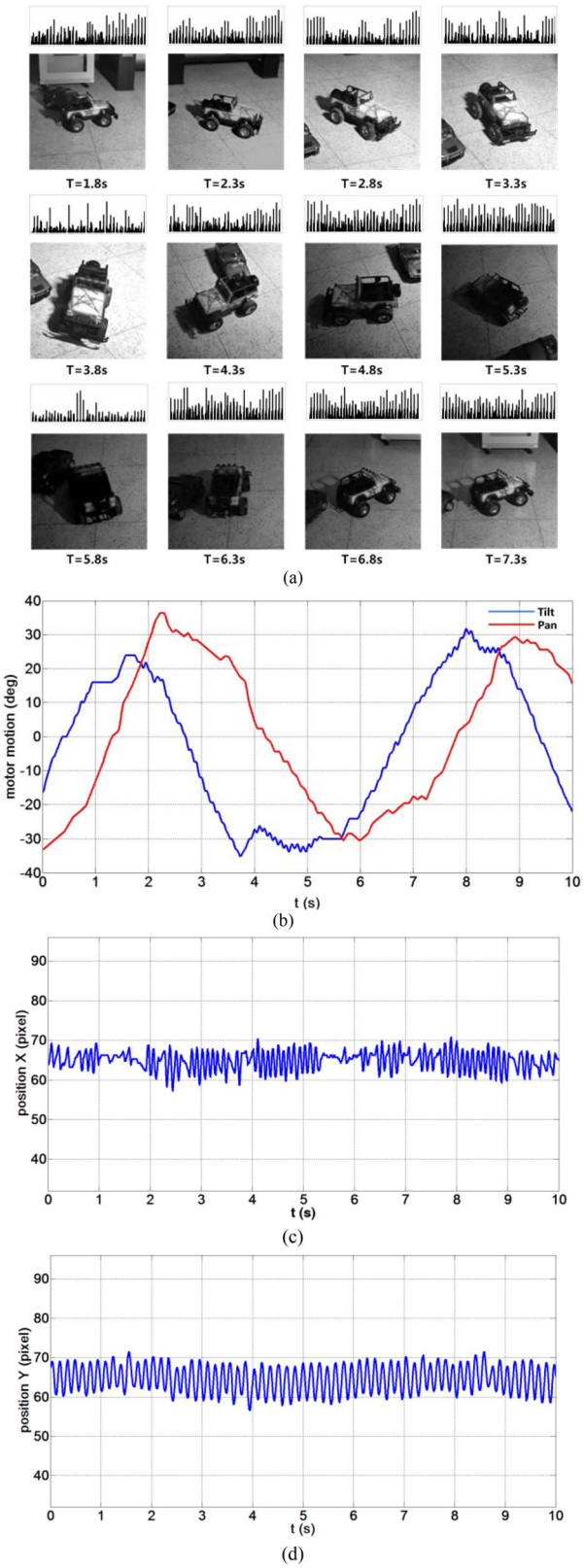

Fig. 20. Target tracking result. (a) Photos and GLLBP histogram captured during tracking. (b) Trajectory of the tilt motor (blue line) and the pan motor (red line). (c) x coordinate of target center. (d) y coordinate of target center.

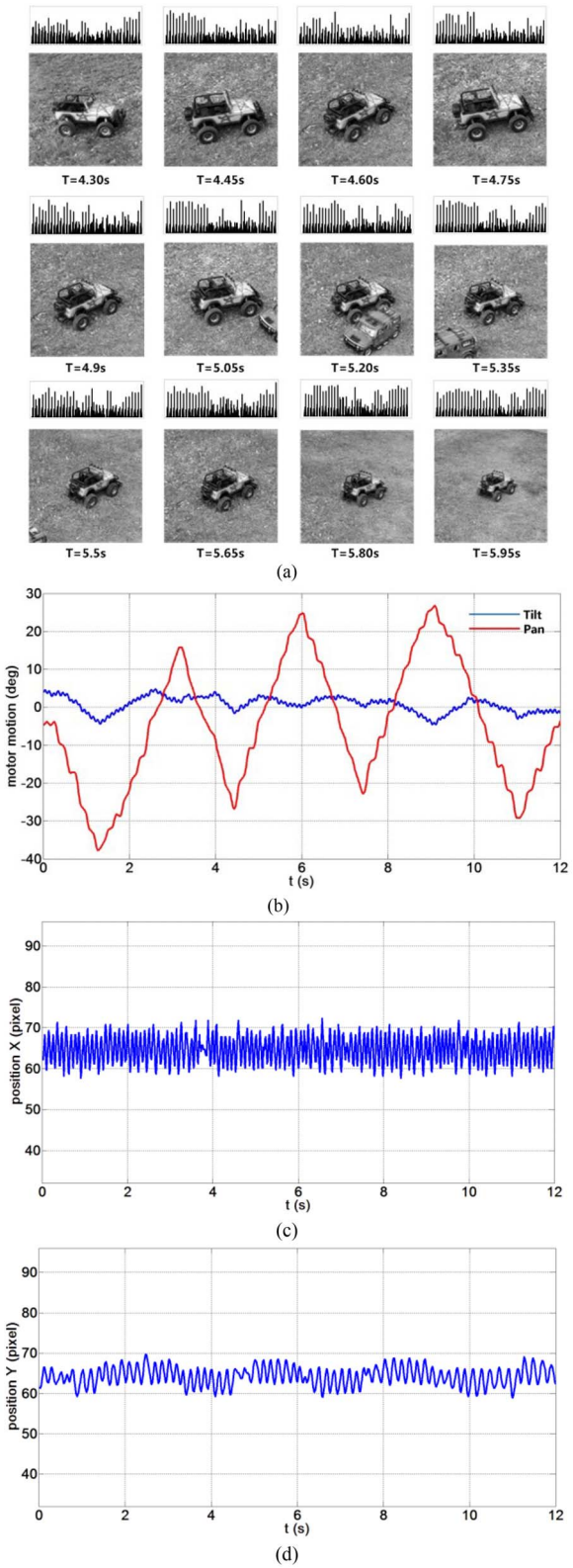

Fig. 21. Results of outdoor tracking. (a) Photos and GLLBP histogram captured during tracking. (b) Trajectory of the tilt motor (blue line) and the pan motor (red line). (c) x coordinate of target center. (d) y coordinate of target center.

of the jeep. The experiment results indicate that the system can achieve successful tracking even if the target moves fast and irregularly.

Then, experiment of outdoor scenery was conducted. The target moved left-and-right, about every 3 s, on the grass in front of the camera. In Fig. 21, the pan motor (red line)

TABLE VI
COMPARISON WITH PREVIOUS STUDIES

Ref	This work	Ref.[20]	Ref.[30]	Ref.[17]	Ref.[16]
Image	8 bit mono	2 bit binary	8 bit mono	8 bit color	8 bit mono
Algorithm	GLLBP based	Binary image based	LBP based	Color tracking	Binary image based
Vector length	1440	N/A	>600	3	N/A
Resolution	750×480	16×16	128×128	512×512	64×64
Hardware setting	Altera 5AGXFB3H4F35C4N	180nm standard CMOS ASIC	FPGA	Xilinx XCVFX60 XC3S5000	UMC 180nm CMOS ASIC
Programmable	Yes	Yes	Yes	No	No
Memory occupied (bit)	536,576	1,024	851,968	N/A	N/A
Environment setting	Outdoor/indoor, complex sceneries, changing illumination	Indoor, clear background, constant illumination	Indoor, certain sceneries, changing illumination	Indoor, certain sceneries, changing illumination	Outdoor/indoor, certain sceneries, constant illumination
Parallelism	Pixel-, row-, thread-parallel	Pixel-,row-parallel	Pixel-, patch-, thread-parallel	N/A	Pixel-parallel
Frame pipeline	Yes	No	No	No	No
Tracking Method	Learning/No learning	No Learning	No Learning	No Learning	No Learning
Frequency (MHz)	100	20	50	151.2	N/A
Processing speed of tracking algorithm (fps)	2800	1000	1000	2000	100
Actuator	2 DOF	1 DOF	2DOF	2DOF	N/A

moved with a period of 3 s, whereas the tilt motor (blue line) did not move as often. This phenomenon corresponds to the left-right movement of the target. The target was always kept around the center of view. The results indicate that the tracking system can still work stably in the open air.

Finally, a tracking experiment of badminton was made to test the speed performance of the system. The maximum speed of the badminton is over 100 km/h and it flies with complicated motion and sudden rotation. The result is shown in Fig. 22. It can be seen that the system still works stably facing this challenging scenery.

Table VI shows a comparison with previously reported tracking systems. Hardware of [16] and [17] is not programmable by software, which limits the application with different sceneries. In [30], 8×8 PEs are connected to a patch processing unit in his architecture. The structure is very inefficient when the size of subregion is not equal to 8×8, which limits the application of tracking algorithms. Thanks to the improved ability of data transmission between processors, the data need only be stored in the PE array and accessed by RP and MPU at a high speed. This allows PE, RP, and MPU to share the same memory. It leads to the fact that the memory occupied by our processor (without considering the memory cost in Ethernet interface) is also smaller than [30], although the resolution is much larger. The algorithms proposed in these researches either utilize a threshold value to obtain a binary image [16] or capture a binary image in the first place [20]. Although in [17] color images are used to track, they also adopt several thresholds

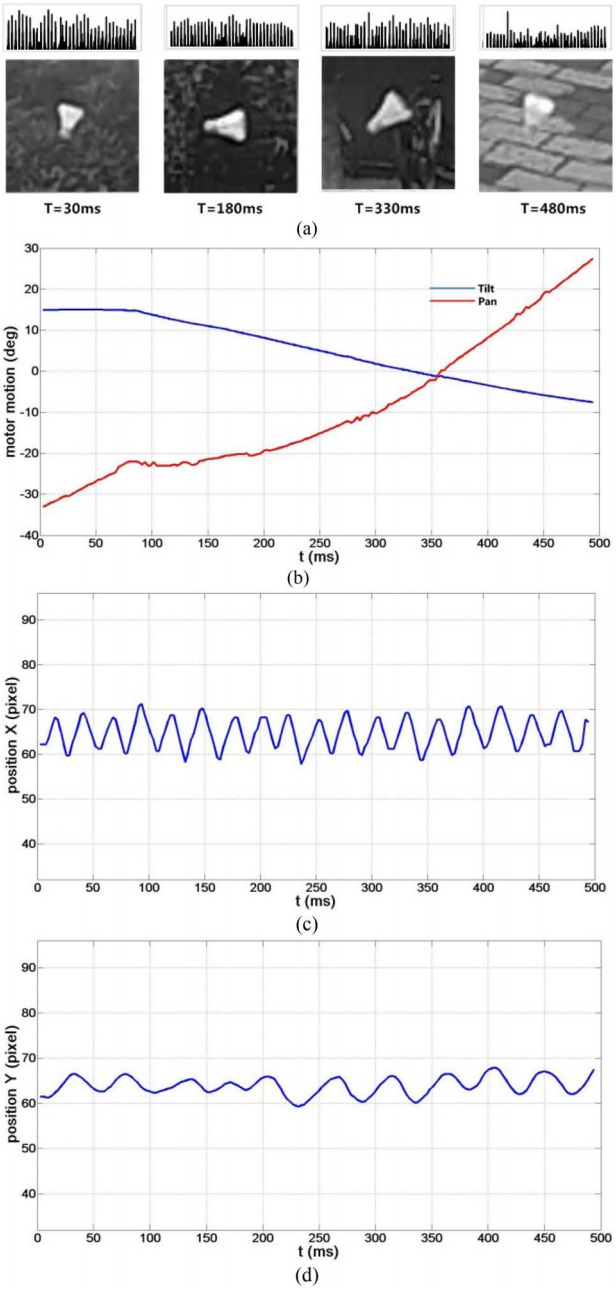

Fig. 22. Results of badminton tracking. (a) Photos and GLLBP histogram captured during tracking. (b) Trajectory of the tilt motor (blue line) and the pan motor (red line). (c) x coordinate of target center. (d) y coordinate of target center.

to convert an HSV image to a binary image. These algorithms can all be classified as binary-image-based methods. The methods can only work in artificial environments with a clear background. An LBP-based tracking algorithm is implemented on a vision processor in [30]. However, LBP-based tracking may degenerate in indistinct environment, as mentioned in Sections II and V. The proposed tracking algorithm with a GLLBP feature can work in much more complex environments. Furthermore, the proposed hierarchical parallel architecture speeds up the algorithm in pixel and row level parallelism.

VI. Conclusion

This paper proposed one novel high-speed target tracking system based on the hierarchical parallel vision processor architecture and the GLLBP tracking algorithm. The proposed processor integrates a pixel-parallel PE array, a row-parallel RP, a dual-core RISC MPU, and a motor controller. The PE array and RP array speed up the low-level and middle-level image processing operations by $O(M^2)$ and $O(M)$, respectively. Each RP can randomly access any PE in the same row directly and acquire global information efficiently. The tracking algorithm based on a novel GLLBP descriptor was proposed. The descriptor describes not only local texture feature but also distribution of luminance. It increases the robustness of the tracking system under low resolution scenery and background with similar texture. This tracking algorithm can be implemented on the vision processor within 0.5 ms. The efficacy of our proposed system was verified in real-time experiment for tracking a fast moving target under complex conditions. In the future, we will extend the use of our tracking system to various application fields in robot control, surveillance, factory inspection, and biometric applications.

Acknowledgment

The authors would like to thank all of the members of the vision processing research team, in particular, Q. Qin, Z. Chen, B. Li, Z. Zhang, and H. Li.

References

[1] A. Yilmaz, O. Javed, and M. Shah, "Object tracking: A survey," *ACM Comput. Survey*, vol. 38, no. 4, 2006, Art. ID 13.
[2] H. Yang, L. Shao, F. Zheng, L. Wang, and Z. Song, "Recent advances and trends in visual tracking: A review," *Neurocomputing*, vol. 74, no. 18, pp. 3823–3831, Nov. 2011.
[3] S. Challa, *Fundamentals of Object Tracking*. Cambridge, U.K.: Cambridge Univ. Press, 2011.
[4] N. Ogawa, H. Oku, K. Hashimoto, and M. Ishikawa, "Microrobotic visual control of motile cells using high-speed tracking system," *IEEE Trans. Robot.*, vol. 21, no. 4, pp. 704–712, Aug. 2005.
[5] T. Hasegawa, N. Ogawa, H. Oku, and M. Ishikawa, "A new framework for microrobotic control of motile cells based on high-speed tracking and focusing," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, Pasadena, CA, USA, 2008, pp. 3964–3969.
[6] M. Ishikawa, K. Ogawa, T. Komuro, and I. Ishii, "A CMOS vision chip with SIMD processing element array for 1 ms image processing," in *IEEE Int. Solid-State Circuits Conf. Tech. Dig. (ISSCC)*, San Francisco, CA, USA, 1999, pp. 206–207.
[7] T. Komuro, I. Ishii, M. Ishikawa, and A. Yoshida, "A digital vision chip specialized for high-speed target tracking," *IEEE Trans. Electron Devices*, vol. 50, no. 1, pp. 191–199, Jan. 2003.
[8] T. Komuro, S. Kagami, and M. Ishikawa, "A dynamically reconfigurable SIMD processor for a vision chip," *IEEE J. Solid-State Circuits*, vol. 39, no. 1, pp. 265–268, Jan. 2004.
[9] A. Rodríguez-Vázquez *et al.*, "ACE16k: The third generation of mixed-signal SIMD-CNN ACE chips toward VSoCs," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 51, no. 5, pp. 851–863, May 2004.
[10] N. Massari, M. Gottardi, L. Gonzo, D. Stoppa, and A. Simoni, "A CMOS image sensor with programmable pixel-level analog processing," *IEEE Trans. Neural Netw.*, vol. 16, no. 6, pp. 1673–1684, Nov. 2005.
[11] N. Massari and M. Gottardi, "A 100 dB dynamic-range CMOS vision sensor with programmable image processing and global feature extraction," *IEEE J. Solid-State Circuits*, vol. 42, no. 3, pp. 647–657, Mar. 2007.
[12] J. Dubois, D. Ginhac, M. Paindavoine, and B. Heyrman, "A 10 000 fps CMOS sensor with massively parallel image processing," *IEEE J. Solid-State Circuits*, vol. 43, no. 3, pp. 706–717, Mar. 2008.

[13] C.-C. Cheng, C.-H. Lin, C.-T. Li, and L.-G. Chen, "iVisual: An intelligent visual sensor SoC with 2790 fps CMOS image sensor and 205 GOPS/W vision processor," *IEEE J. Solid-State Circuits*, vol. 44, no. 1, pp. 127–135, Jan. 2009.
[14] I. Ishii, T. Taniguchi, R. Sukenobe, and K. Yamamoto, "Development of high-speed and real-time vision platform, H^3 vision," in *Proc. IEEE/RSJ Int. Conf. Intell. Robot. Syst. (IROS)*, St. Louis, MO, USA, 2009, pp. 3671–3678.
[15] A. Lopich and P. Dudek, "A SIMD cellular processor array vision chip with asynchronous processing capabilities," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 58, no. 10, pp. 2420–2431, Oct. 2011.
[16] B. Zhao, X. Zhang, S. Chen, K.-S. Low, and H. Zhuang, "A 64×64 CMOS image sensor with on-chip moving object detection and localization," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 22, no. 4, pp. 581–588, Apr. 2012.
[17] Q. Gu, A. Al Noman, T. Aoyama, T. Takaki, and I. Ishii, "A fast color tracking system with automatic exposure control," in *Proc. IEEE Int. Conf. Inf. Autom. (ICIA)*, Yinchuan, China, 2013, pp. 1302–1307.
[18] L. Fiack, N. Cuperlier, and B. Miramond, "Embedded and real-time architecture for bio-inspired vision-based robot navigation," *J. Real-Time Image Process.*, vol. 10, no. 4, pp. 699–722, Jan. 2014.
[19] Y. Yamada and M. Ishikawa, "High speed target tracking using massively parallel processing vision," in *Proc. IEEE/RSJ Int. Conf. Intell. Robot. Syst. (IROS)*, vol. 1. Yokohama, Japan, 1993, pp. 267–272.
[20] W. Miao, Q. Lin, W. Zhang, and N.-J. Wu, "A programmable SIMD vision chip for real-time vision applications," *IEEE J. Solid-State Circuits*, vol. 43, no. 6, pp. 1470–1479, Jun. 2008.
[21] W. Zhang, Q. Fu, and N.-J. Wu, "A programmable vision chip based on multiple levels of parallel processors," *IEEE J. Solid-State Circuits*, vol. 46, no. 9, pp. 2132–2147, Sep. 2011.
[22] T. Komuro, I. Ishii, M. Ishikawa, and A. Yoshida, "High speed target tracking vision chip," in *Proc. 5th IEEE Int. Workshop Comput. Archit. Mach. Percept. (CAMP2000)*, Padua, Italy, 2000, pp. 49–56.
[23] W. Miao, Q. Lin, and N. Wu, "A novel vision chip for high-speed target tracking," *Japanese J. Appl. Phys.*, vol. 46, p. 2220, Apr. 2007.
[24] Z. Kalal, K. Mikolajczyk, and J. Matas, "Tracking-learning-detection," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 34, no. 7, pp. 1409–1422, Jul. 2012.
[25] K. Zhang, L. Zhang, and M.-H. Yang, "Real-time compressive tracking," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, Florence, Italy, 2012, pp. 864–877.
[26] B. Babenko, M.-H. Yang, and S. Belongie, "Robust object tracking with online multiple instance learning," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 33, no. 8, pp. 1619–1632, Aug. 2011.
[27] A. Adam, E. Rivlin, and I. Shimshoni, "Robust fragments-based tracking using the integral histogram," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, New York, NY, USA, 2006, pp. 798–805.
[28] T. Ahonen, A. Hadid, and M. Pietikainen, "Face description with local binary patterns: Application to face recognition," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 28, no. 12, pp. 2037–2041, Dec. 2006.
[29] M. Heikkila and M. Pietikainen, "A texture-based method for modeling the background and detecting moving objects," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 28, no. 4, pp. 657–662, Apr. 2006.
[30] J. Yang, C. Shi, L. Liu, and N. Wu, "Heterogeneous vision chip and LBP-based algorithm for high-speed tracking," *Electron. Lett.*, vol. 50, no. 6, pp. 438–439, Mar. 2014.
[31] T. Ojala, M. Pietikainen, and T. Maenpaa, "Multiresolution gray-scale and rotation invariant texture classification with local binary patterns," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 24, no. 7, pp. 971–987, Jul. 2002.
[32] P. Viola and M. J. Jones, "Robust real-time face detection," *Int. J. Comput. Vis.*, vol. 57, no. 2, pp. 137–154, May 2004.
[33] C. Shan, "Learning local binary patterns for gender classification on real-world face images," *Pattern Recognit. Lett*., vol. 33, no. 4, pp. 431–437, Mar. 2012.
[34] Y. Wu, J. Lim, and M.-H. Yang, "Online object tracking: A benchmark," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Portland, OR, USA, 2013, pp. 2411–2418.

Yongxing Yang received the B.S. degree from Tsinghua University, Beijing, China, in 2011. He is currently pursuing the Ph.D. degree with the Institute of Semiconductors, Chinese Academy of Sciences, Beijing.

His research interests include visual target tracking, image processing, computer vision, and deep learning.

Jie Yang was born in Chongqing, China, in 1987. He received the B.S. degree in electronic science and technology from Tianjin University, Tianjin, China, in 2010, and the Ph.D. degree in microelectronics from the Institute of Semiconductors, Chinese Academy of Sciences, Beijing, China, in 2015.

He is currently a Post-Doctoral Fellow with the Department of Electrical and Computer Engineering, University of Calgary, Calgary, AB, Canada. His research interests include algorithms and very large-scale integration architecture of image processing, computer vision, and deep learning.

Liyuan Liu (M'11) received the B.S. and Ph.D. degrees in electrical engineering from the Institute of Microelectronics, Tsinghua University, Beijing, China, in 2005 and 2010, respectively.

He is currently an Associate Professor with the Institute of Semiconductors, Chinese Academy of Sciences, Beijing, to study high-performance analog front-end design for CMOS image sensors.

Nanjian Wu (M'05) was born in Zhejiang, China, in 1961. He received the B.S. degree in physics from Heilongjiang University, Harbin, China, in 1982, the M.S. degree in electronic engineering from Jilin University, Changchun, China, in 1985, and the D.S. degree in electronic engineering from the University of Electronic Communications, Tokyo, Japan, in 1992.

In 1992, he joined the Research Center for Interface Quantum Electronics and the Faculty of Engineering, Hokkaido University, Sapporo, Japan, as a Research Associate. In 1998, he was an Associate Professor with the Department of Electro-Communications, University of Electronic Communications. Since 2000, he has been a Professor with the Institute of Semiconductors, Chinese Academy of Sciences, Beijing, China. In 2005, he was a Visiting Professor with the Research Center for Integrated Quantum Electronics, Hokkaido University. Since 2009, he has been an Honorable Guest Professor with the Research Institute of Electronics, Shizuoka University, Shizuoka, Japan. His current research interests include mixed-signal large-scale integrated design and semiconductor quantum devices.