FPGA-based Real-Time Object Tracking for Mobile Robot

Xiaofeng Lu[1][2], Diqi Ren[2], Songyu Yu[1]
[1] *Shanghai Jiao Tong University,* [2] *Shanghai University*
Luxiaofeng_sjtu@sjtu.edu.cn

Abstract

*This paper proposes an embedded vision system for real-time moving object tracking using modified mean-shift algorithm for mobile robot application. This design of modified mean-shift algorithm fully utilizing the advanced parallelism of Field Programmable Gate Arrays (FPGA) is capable of processing real-time PAL video of 720*576 at 25 fps. This hardware implementation realizes time-consumed color space transformation using pipeline operations, which completely removes the dependence of off-chip RAM memory. In addition, this system incorporates adaptive kernel based mass center calculation method to balance the tracking complication and precision. Finally, the time division multiplex (TDM) technology obviously saves the FPGA hardware resources. As a result, the feasibility and tracking robustness of this implementation have been demonstrated through real-time tracking videos.*

1. Introduction

Because of complicated tracking algorithm and limited embedded computing ability, embedded real-time object tracking is still an important challenge in computer vision and embedded system for mobile robot or embedded navigation. Due to the advanced technologies in digital signal processing, such as high speed, programmable flexibility and parallel ability, Field Programmable Gate Arrays (FPGA) plays an increasing role in embedded image or video processing. Therefore, many researchers have proposed kinds of image processing solutions based on FPGA architecture to achieve real-time and efficient image content extraction and object feature analysis [1][2][3][4].

Schlessman in [5] discusses a practical design based on a hardware/software co-design to realize an optical flow tracking system, which puts the KLT portion that consumes much processing time to FPGA-based hardware implementation as optimization. Usman in [6] presents an object tracking solution based on a soft RISC processor running mean-shift algorithm in FPGA. This system needs a great of off-chip DDR RAM as memory access. Soft processor frequency is 50MHz, so it just supports 360*288 video resolutions. Christopher in [7] introduces an object tracking hardware design based on color features. And by tracking information system can realize some system control applications.

This paper introduces the implementation of a real-time moving object tracking on large scale FPGA hardware architecture based on modified mean-shift algorithm. And our novel design focuses on real-time processing, without any additional off-chip memory spending and optimizing FPGA logic elements (LE) as much as possible. Specifically, the novel algorithm improvements based on FPGA hardware structure including three facets: (1) Pipelined architecture of modified color space transformation model realizes effective pre-processing and just uses little on-chip RAM memory. (2) kernel-based mass center calculation can balance the calculating complication and precision, and its achieving flow is fit for FPGA computing architecture. (3) The time division multiplex (TDM) technology in algorithm organization optimization obviously saves FPGA LE resources.

The rest of the paper is organized as follows. Section 2 presents the framework of mean-shift object tracking algorithm, and section 3 introduces the implementation details of modified mean-shift tracking algorithm on FPGA. In section 4, experimental platform and results of some live sequences are presented. Finally, section 5 concludes this paper.

2. Mean-shift tracking algorithm

Mean-shift is a nonparametric density gradient statistical method, which has been widely applied to image segmentation, clustering and tracking.

Firstly, $K(x): x \to R$ is defined as a search window. If its profile function $k(x):[0,\infty] \to R$ exists, $K(x)$ is

called a kernel [8]. And if $K(x)$ is a symmetrical kernel, it can be defined as follows,

$$K(x) = ck\left(\|x\|^2\right) \tag{1}$$

Where c is a normalized constant.

Secondly, in N-dimensional Euclidean space X, if $x_i : 1 \le i \le n$ means an independent sample set, from literature [9] and [10], multivariate kernel probability density estimate at location x is proposed as

$$f_{h,k}(x) = c_h \sum_{i=1}^{n} k\left[\left\|\frac{x - x_i}{h}\right\|^2\right] w(x_i) \tag{2}$$

Where $w(x_i)$ is the weight coefficient regarded as prior probability of x_i, and c_h is a normalized constant.

Finally, from [8] and the summarized conception we can defined the mean-shift vector weighted by kernel function as follows,

$$\overline{m}_k(x) = \frac{\sum_{i=1}^{n} x_i g\left[\left\|\frac{x - x_i}{h}\right\|^2\right]}{\sum_{i=1}^{n} g\left[\left\|\frac{x - x_i}{h}\right\|^2\right]} - x \tag{3}$$

Where $g(x) = -k(x)$. As a result, shifting each data point to the local maximum probability along the maximum mean-shift vector direction is the key point of mean-shift object tracking process, and this iteration will converge relying on local maximum or iterating times.

3. FPGA based algorithm architecture

In this mobile robot tracking system, Cyclone series FPGA EP1C6 of Altera Co. is used as processing unit. A block diagram of this implementation is shown below in figure 1.

System captures analog composite PAL video with a CCD camera, and it is converted to digital RGB video data by video decoder chip SAA7111 produced by Philips. From video decoder module in FPGA, we can obtain a stream of 16-bit interlaced RGB (5:6:5) successive fields, which is compatible with ITU-R BT.656 standard. Simultaneously, this module grabs vertical and horizontal synchronous signals as timing benchmark for video processing, system status transferring and video display. When operator chooses a red rectangle on real-time video as target location by keyboard, tracking system begins to enable color space transformation module, modified mean-shift module, and tracking result display module to accomplish object tracking process at 100MHz frequency. In addition, FPGA hardware control system can choose different English font programmed in FPGA ROM memory beforehand as system status indication. Lastly, tracking video data need to convert to LVDS signal for LCD display from LCD driver module and LVDS transformation module.

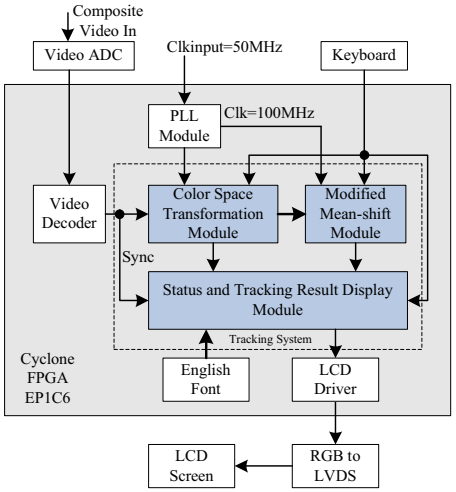

Figure 1. Block diagram of tracking system

3.1. Color space transformation module

Color space transformation plays an important role in mobile robot tracking system as a pre-processing module. In traditional image processing field, several color spaces have been widely used, including RGB, YUV, YCbCr and HSV (Hue, Saturation, value) [4]. Commonly, within RGB space, little intensity change of any color class will result in a diagonal movement of RGB cube, and that can lead great object location error. Christopher [7] used YUV space to overcome this intensity interdependence problem. In YUV space, chrominance information, which is coded in two dimensions U and V, is separated from intensity information. As a result, YUV space is more dependable than RGB for object feature extraction [4].

Up to now, several researchers have approved that H component of HSV space is more insensitive to illumination variety and more intuitive to human vision system, going with its expensive computation. In this paper, we extend the standard RGB cube to HSV cone space transformation matrix in FPGA system by multiplying a constant 16 to H operation for FPGA hardware architecture migration, which alters H value from [-1, 5] to [-16, 80]. So, from modified calculation equation (4) to equation (6), we can obtain H value of every pixel in region of interest (ROI), and its complement code denotation is [0, 79] and [112, 127].

$$MAX = \max(R, G, B) \tag{4}$$

$$MIN = \min(R, G, B) \tag{5}$$

$$H = \begin{cases} \dfrac{G-B}{MAX-MIN} \times 16, if \rightarrow R = MAX \\ 32 + \dfrac{B-R}{MAX-MIN} \times 16, if \rightarrow G = MAX \\ 64 + \dfrac{R-G}{MAX-MIN} \times 16, if \rightarrow B = MAX \end{cases} \tag{6}$$

And then, the H component calculation of every pixel needs two comparing operations, two subtraction operations, one division operation, one multiplication operation and one addition operation. In PAL digital video format, pixel clock is 13.5MHz. That means it is almost impossible to complete these six operations for real-time processing in one clock.

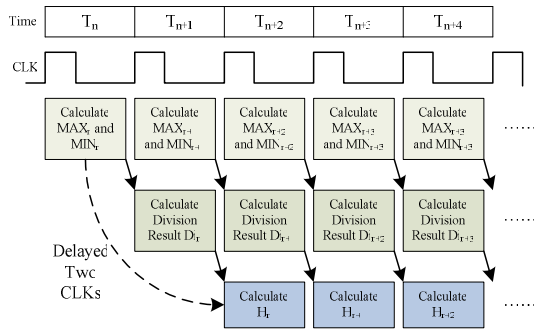

Figure 2. Pipelined color space transformation structure

Fortunately, in this implementation, we propose a pipelined color space transformation structure to improve calculating efficiency which is shown in figure 2. This pipeline flow utilizes the parallelism of FPGA skillfully. In first clock, the MAX and MIN of RGB components of current pixel is calculated. In second clock, the algorithm can use results of two subtractions to gain the division result. And the H component value can be obtained at third clock. Up to now, through delaying two clocks, this implementation can calculate H component for every pixel in real-time.

As we know, black (V=0, H and S have no definition) and white (V=1, S=0, H has no definition) in HSV color space have no H information, but these two colors will appear with high probability in real environment. So in the optimization of color space transformation, we define the projection of grayscale (black and white) to [80, 111]. In one hand, this projection perfects the H mapping of all color components. In the other hand, this projection is suitable to the FPGA calculation structure. The optimization is defined as equation (7), and the optimized result is shown in figure 3.

$$H = \begin{cases} 80 + MAX, if \rightarrow MAX = MIN \\ \dfrac{G-B}{MAX-MIN} \times 16, if \rightarrow R = MAX \\ 32 + \dfrac{B-R}{MAX-MIN} \times 16, if \rightarrow G = MAX \\ 64 + \dfrac{R-G}{MAX-MIN} \times 16, if \rightarrow B = MAX \end{cases} \tag{7}$$

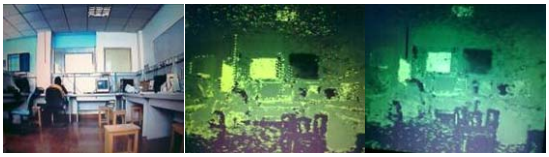

Figure 3. Colour space transformation module optimization result. Left is original image. Middle is non-optimized. Right is optimized. (H component is mapped to G component for display)

3.2. Modified mean-shift tracking module

After color space transformation, the algorithm enters a tracking cycle. In initial state, system waits start signal from keyboard. And object current location is estimated through object mass center of previous field and mean-shift process.

In an object window, tracking system samples every effective pixel value of H channel to form Hue histogram (His) as object reference histogram model, and calculates color probability histogram model (H(i)) from His. When system enters tracking state, algorithm examines the H(i) to check the probability that current pixel belongs to target in search window.

After the back projection based on the probability of searched pixel, normal image is changed to color probability distribution image. The area has much higher probability belonging to target if the luminance is much higher in the area of projection image.

In mean-shift algorithm, if we define (x, y) as a pixel in search window, and $I(x, y)$ is the pixel value in back projection image of the physical position (x, y), then we can calculate the zeroth moment M_{00} and first moment M_{10} and M_{01}. Through the equation (8), we can obtain the mass center of the search window in frame n and move the search window to

mass center. This calculation cycle will not finish until the moving distance or the calculating time is less than user definition.

$$(x_n, y_n) = \left(\frac{M_{10}}{M_{00}}, \frac{M_{01}}{M_{00}} \right) \quad (8)$$

In this paper, we enhance above mass center calculation method by introducing a kernel function $\alpha(x, y)$ as equation (9) to improve the balance of computing precision and complication [9]. Figure 4 obviously shows the object window and search window definition. If the current pixel is in object window, moments kernel weight is 1, correspondingly if it is in search window, the kernel weight is defined 1/2. Initial object window location is chosen by keyboard, our algorithm automatically extends 100 pixels in horizontal and 40 pixels in vertical to form the search window for object tracking system. The new moments calculation equations are defined in (10), (11), and (12). This novel mass center calculating method can solve the problem of mass center excursion caused by background information in search window, and the problem of tracking robustness caused by the lack of object information.

$$\alpha(x, y) = \begin{cases} 1, if\ (x, y) \in ObjectWindow \\ \frac{1}{2}, if\ (x, y) \in SearchWindow \\ 0, if\ (x, y) \in OtherArea \end{cases} \quad (9)$$

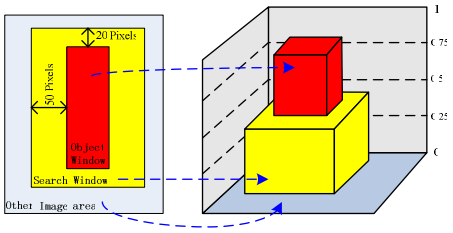

Figure 4. Kernel based object search window definition

$$M_{00} = \sum_x \sum_y I_n(x, y)\alpha(x, y) \quad (10)$$

$$M_{01} = \sum_x \sum_y y I_n(x, y)\alpha(x, y) \quad (11)$$

$$M_{10} = \sum_x \sum_y x I_n(x, y)\alpha(x, y) \quad (12)$$

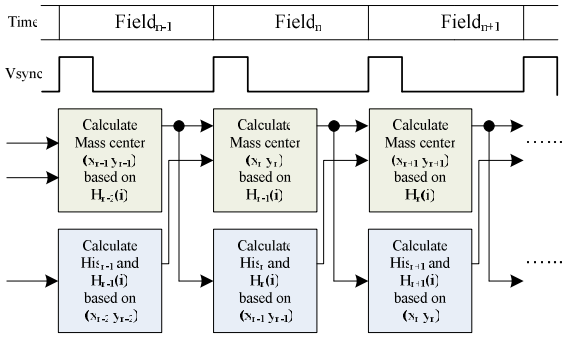

Figure 5. Pipeline based mass center calculation flow

Like above color space transformation calculation, we also design a pipeline based parallel mass center hardware architecture, which is shown in figure 5. Vertical sync is the time benchmark. In field n period, firstly we calculate Hue histogram His_n and color probability histogram $H_n(i)$. Based on the mass center (x_{n-1}, y_{n-1}) and search window of previous field, we synchronously calculate current mass center (x_n, y_n) by previous probability histogram $H_{n-1}(i)$. This simplified pipelined structure does not need additional cache and RAM memory consumption.

3.3. Status and result display module

Status and result display module achieves video and control information display, including state information, red object rectangle, and video signal timing, which is shown in figure 6. System state scans English font stored in FPGA inside ROM as OSD information, and these state information are controlled by user operation as man-machine interaction.

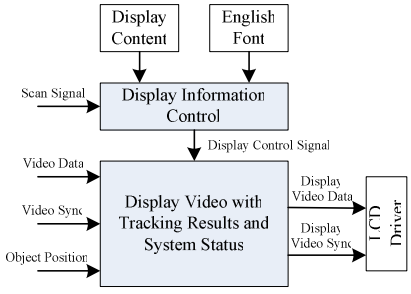

Figure 6. Tracking results and status display module

3.4. FPGA LE resource optimization

In the process of object tracking design in FPGA, there are many 16-bits multiplications and divisions. But Cyclone EP1C6 FPGA does not have ready

embedded hardware multipliers and divisions. So FPGA resource consumption is depending on the LE used for these complicated operations. Through experiments we find that multiplications and divisions are not used at exact same time. Obviously, the TDM technology of hardware resource allocated to multipliers and divisions can save FPGA hardware LE greatly. The experimental result is shown in table 1.

Table 1. System optimization result

	Before Optimized	After Optimized	Optimization Efficiency
LE No.	4093	3284	24.8%

4. Experimental results

In this section, the performance of FPGA-based object tracking implementation is presented. We initialize the object location by keyboard in first field, and the target estimated in following experiments is shown as a red rectangle. And some results about tracking robustness are discussed in details.

In this algorithm implementation, we utilize the SOPC platform, which is an embedded development kit in our previous work. From the CCD camera, analog PAL video data can be captured by the FPGA option board, which is developed for this design and is the key research point in this tracking system. And this board is based on a low cost and low power consumption FPGA for parallel data processing.

4.1. Tracking results

4.1.1. Tracking validity.

Figure 7 presents the tracking validity of our modified algorithm and FPGA-based structure. We can find that this implementation can realize real-time object tracking in actual lab environment. Therefore, this embedded system design for real-time object tracking is feasible.

Figure 7. Tracking results of a brown notebook in lab

4.1.2. Illumination.

In order to demonstrate the efficiency of this implementation to illumination variety, we also give the tracking results in different illumination environment of our algorithm as figure 8 shown. These results can indicate that this design is not so sensitive in different light conditions.

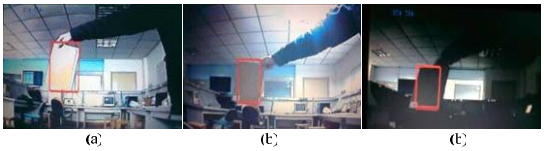

Figure 8. Tracking in different illumination. (a) is in sunlight. (b) is in lamplight. (c) is in sombrous state.

5. Conclusion

A modified mean-shift real-time object tracking algorithm and its embedded FPGA-based implementation have been proposed in this paper. Simultaneously, we present FPGA-based parallel algorithm architecture, especially the pipeline structure and TDM-based optimization of our design is analyzed in details. Finally, many pointed experimental results demonstrate the correctness and robustness of our implementation.

Acknowledgement

This research was supported in part by NSFC (60702044), Innovation foundation of Shanghai University (A10-0107-09-902).

Reference

[1] Zhaoyi Wei, Dah-Jye Lee, and Brent E. Nelson, FPGA-based Real-time Optical Flow Algorithm Design and Implementation, Journal of Multimedia,Vol.2, No.5, 2007.

[2] K Benkrid and Samir Belkacemi, Design and Implementation of a 2D Convolution Core for Video Application on FPGAS. International Workshop on Digital and Computational Video. 2002.

[3] D.G. Bariamis, D.K. Iakovidis, and D.E. Maroulis. An FPGA-based Architecture for Real Time Image Feature Extraction. Proceedings of the 17th International Conference on Pattern Recognition, p 801–804, ICPR2004.

[4] Qingrui Zhou, Kui Yuan, Wei Zou, and Huosheng Hu, FPGA-Based Colour Image Classification for Mobile robot Navigation, Proceedings of the IEEE International Conference on Industrial Technology, p 921-925, 2005.

[5] Schlessman, J., Chen, C. Y., Ozer, B., Fujino, K., Itoh, K., and Wolf, W., Hardware/software Co-design of an FPGA based Embedded Tracking System, in Proceedings of the IEEE Conference on Computer Vision and Pattern

Recognition Workshop, p 123-133, 2006.
[6] Usman Ali, M.B. Malik, and Khalid Munawar, FPGA/Soft-processor based Real-time Object Tracking System, 5th Southern Conference on Programmable Logic, p 33-37, 2009.
[7] Christopher T. Johnston, Kim T. Gribbon, and Donald G. Bailey, FPGA based Remote Object Tracking for Real-time Control, 1st International Conference on Sensing Technology, 2005.
[8] Yizong Cheng, Mean Shift, Mode Seeking, and Clustering, IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol.17, No.8, 1995.
[9] D. Comaniciu, V. Ramesh, and P. Meer, Real-time Tracking of Non-rigid Objects Using Mean Shift, Proc. IEEE Conference on Computer Vision and Pattern Recognition, Vol. 2, p 142-149, 2000.
[10] Qi Sumin and Huang Xianwu, Non-rigid Object Tracking by Anisotropic Kernel Mean Shift, Transactions of Tianjin University, Vol. 13, No. 5, 2007.