Real-time Object Tracking Method based on Multi-Core DSP

Lidan Li¹*, Yi Zhang¹, Qianya Guo¹, Qing Lin¹, Jing Zhou¹, Yanyong Wang¹, Bo Zhao¹

¹ China North Vehicle Research Institute, Beijing, China
* e-mail: consultant@noveri.net

Abstract. In this paper, we propose a fast object tracking method used in real-time application. The tracking method is based on compact kernelized correlation filters. Traditional, gray values are used to describe target feature. In order to achieve relatively fast performance, we adopt Principal Component Analysis (PCA) to simplify Histogram of Gradient (HOG) as feature descriptor, which can significantly reduce computation resources. Our method is implemented on Texas Instruments (TI) multi-core digital signal processor (DSP) TMS320 C6657, which can achieve 60 frame per-second (fps). Finally, we conduct both quantitative and qualitative experiment to show the robustness and real-time performance of our method.

1. Introduction

Object tracking refers to automatically determine object location and scale in successive frames, given object position and size in first frame. Object tracking is an active research area in computer vision for many years, which has widely application in automatic driving, surveillance, etc. Although tremendous progress has made in recent years, especially with the prosperous of deep learning, object tracking still has a lot of problem in application. First, it is hard to achieve robust tracking for commonly used tracking method under complex background where it is hard to discriminate target and background. Secondly, most of target tracking algorithms are running on high-performance platform, where powerful CPUs or GPUs are used. However, it is impossible to use these power-consumed devices in some power deficient platform.

In recent years, top performance tracking methods include correlation filter based methods[1-4] and deep learning based methods[5-7]. MOSSE[1] refer to minimizing the output sum of squared error, which presents a new type of correlation filter to produce stable correlation filters when initialized using individual image. It can achieve super real-time but poor tracking performance. Kernelized Correlation Filter (KCF)[2] is proposed to extend the MOSSE tracker by employing different kernels to correlation filter. Instead of gray image, histogram of gradient (HOG) is used to feature boost tracking performance. Discriminative Scale Space Tracking (DSST)[3] further improve the tracking performance by building dual correlation filter to estimate target position and scale separately. Fast Discriminative Scale Space Tracking (fDSST)[4] investigate a novel scale adaptive tracking approach by learning separate discriminative correlation filter by sampling target appearance at a series of different scales. Additionally, they investigate strategies to reduce the computational cost. DSST is the winner of the VOT 2014 challenge[5]. With the prosperous of deep neural network, deep learning based tracking method show excellent results compared to correlation filter based tracker. These methods include MDNet[6], the winner of VOT2015, C-COT[7], the winner of VOT 2016, and other excellent work[8,9]. However, these algorithms are extremely complicate, huge computation and memory is needed which is not applicable for real time situation.
In the paper, we build a real-time target tracker based on correlation filter. Our work is built upon KCF by adding scale estimation and some simplified tricks. We also implement proposed method in DSP TMS320C6657 platform in order to be applied to some storage and computation limited occasion.

2. Correlation filter and target tracking

2.1. Correlation filter

Correlation filter is a concept from signal processing which is successfully employed to target tracking by Bolme [1]. Self-correlation and inter-correlation are two types of correlation. In image processing, we often refer to inter-correlation which can be defined as:

\[
(f \otimes g)(\tau) = \int_{-\infty}^{+\infty} f^*(t)g(t+\tau)dt
\]

(1)

\[
(f \otimes g)(m) = \int_{-\infty}^{+\infty} f^*[m]g(m+n)
\]

(2)

where \( f^* \) is the complex conjugate of \( f \). Correlation is used to determine the similarity of the signals \( f \) and \( g \) at time \( \tau \).

Correlation filter trackers build model for target appearance using filters trained on first image patches and update in following images. Then, the filters are used to measure the similarity of filters and successive images. Figure 1 show full tracking procedure.

![Figure 1. Tracking procedure](image)

2.2. Target tracking based on correlation filter

A filter is used to find maximum response of target position. At the beginning, it needs a series of training images \( f \) and training results \( g \). \( g \) often generated from initial image, which often has a compact 2D gaussian shape centered at the target in training images. We use capital letters \( G \), \( F \), \( H \) denote Discrete Fourier Transform (DFT) of corresponding functions, correlation takes the following form.

\[
g = f \otimes h
\]

(3)

\[
G = F \otimes H^*
\]

(4)

where \( g, f, h \) denote desired correlation output, input image, correlation filter. By using Fast Fourier Transform (FFT), convolution can be turned into dot product of the two signals, which can achieve low computation cost. The correlation filter can be obtained through minimizing the out sum of squared errors, which can be expressed as follow:

\[
\min_{H} \sum_{i}[F_i \otimes H^* - G_i]^2
\]

(5)

The above equation is a optimization problem which can be solved by setting the partial difference of equation to zero to find closed form solution (more details about solving process can be found in literature [2]). In this way, correlation filters can be expressed as follow:

\[
H = \frac{\sum_{i}F_i^*G_i^*}{\sum_{i}F_i^*F_i^*}
\]

(6)

The filter is updated in every to deal with scale change, appearance change, illumination change, etc. Thanks to FFT, the correlation filter based methods can achieve high real-time while maintain relatively good performance.
3. Improved and simplified strategy

3.1. Simplified HOG feature
Traditionally, gray feature is used to depict target. In reference [2], Histogram of gradient (HOG) is used to describe target feature, which achieve fairly good results. HOG is often used in pedestrian detection, where whole image is divided into small cells to compute histogram of gradient orientation. The representation can be formed by combination of these histograms. The computation process can be divided into the following steps:
- Image normalization.
- Divided to small cells.
- Compute gradient in each cell.
- Compute histogram of gradient in each cell.

In recent year, HOG has been successfully applied to object detection, target tracking and so on. Compared to gray feature, HOG make great improvement to tracking and detection. However, it demand large computational and memory resource, which limits its application. We make some simplification to original HOG. Principal component analysis [5] is used to reduce the dimensional of HOG, which is also reduce storage requirement.

3.2. Algorithm improvement
Based on correlation filter, we use the following equation to update target position and scale.
\[
A_t^l = (1 - \eta)A_{t-1}^l + \eta \bar{G}_F^l \\
B_t = \eta \sum_{k=1}^{d} \bar{F}_k^l F_k^l + (1 - \eta)B_{t-1}
\]  
(7) (8)
After each iteration, new position of target is determined. Considering feature map of input image \(f\) which can be represented as \(f^l(l \in \{1,...,d\})\). In every new image, target position can be obtained by maxing Eq. (9):
\[
Y_t = \frac{\sum_{l=1}^{d} A_{t-1}^l \bar{F}_l}{B_{t-1} + \tau}
\]  
(9)

4. Algorithm implementation

4.1. Hardware Platform
The diagram of hardware platform is shown in Figure 2.

![Figure 2. Diagram of hardware platform](image)

Photoelectric Conversion Board receives video data from the video source and transfers optical signals into electric signals which is switched to FPGA on the Tracking Board. FPGA transmits the video data to DSP via SRIO. The algorithm is implemented in DSP which calculates the target positions and scales and returns them to FPGA. FPGA draws tracking boxes according to tracking results on the video. Then FPGA returns the video with the tracking boxes to Photoelectric Conversion
Board. Finally, the processed video is displayed on the screen. You can observe tracking results on the screen directly.

For the large storage and computational requirement, we choose TI DSP TMS320C6657[10-11] to implement our algorithm. TMS320C6657 which uses TI’s KeyStone Multicore Architecture contains two CPU cores and frequency up to 1.25GHz for each core, 1024K Byte Local L2 Per Core, 1MB Multicore Shared Memory Controller, 1GB DDR3, four lanes of SRI O 2.1. It supports OpenMP Programming. FPGA is chosen from the Xilinx Kintex-7 family[12].

4.2. Algorithm Implementation

The proposed algorithm is implemented using Code Composer Studio 5.1.

The configuration and initialization of DSP can be done using GEL file. FPGA transmits video data to extended DDR3 of DSP. The address of DDR3 starts from 0x80000000 which is accessible to DSP. DSP compute HOG features and build initial model based on input video data. The target position and scale, which is sent to FPGA for display, can be obtained by translation filter and scale filter. Meanwhile, the filter model is updated according to new position and scale.

The implementation of the algorithm in DSP is based on OpenMP API which is supported in the KeyStone Multicore Architecture. The OpenMP API is a portable, scalable model that provides developers utilizing TI’s multicore processors a simple yet flexible interface for developing parallel applications in high-performance computing disciplines. It’s an efficient parallel programming method without increasing software complexity. OpenMP is a thread-based programming language. The master thread executes the sequential parts of a program. When the master thread encounters a parallel region, it forks a team of worker threads that execute in parallel with the master thread.

In order to implement the algorithm with OpenMP, the most important issue is the allocation of the memory space. As the internal memory resources in DSP are limited and the data storage location influences greatly on real-time tracking, it’s very vital to allocate memory reasonably. The details of memory allocation can be found in .cmd file,

- heap and stack size: -heap 0x1000, -stack 0x1000;
- library:
  -  ti_runtime_rts6000_debug_e66.ae66
  -  ti_runtime_device_c6657_debug_e66.ae66
  -  ti_runtime_openmp_c6657_debug_e66.ae66
- Memory instruction:
  MEMORY
  {
  L2SRAM(RW) o = 0x00800000  l = 0x00020000
  L2SRAM_CODE(RW) o = 0x10820000  l = 0x000DFD20
  MSMC_NC_VIRT(RW) o = 0xA0000000  l = 0x00020000
  MSMC_NC_PHY(RW) o = 0x0C000000  l = 0x00020000
  MSMCSRAM o = 0x0C020000  l = 0x000E0000
  DDR3 o = 0x80800000  l = 0x28000000
  DDR3_CORE0 o = 0x90000200  l = 0x08000000
  }
- SECTIONS instructions:
  SECTIONS
  {
  .Entry >MSMCSRAM START(BootMagic)
  .text > L2SRAM_CODE
  .cinit > DDR3
  .const > DDR3
  .switch > MSMCSRAM
  .stack > L2SRAM

5. Experiments

In this section, both quantitatively and qualitatively experiments are conducted to demonstrate the high performance of our algorithm. At first, we use some ground truth datasets to perform quantitative comparison with other algorithms. Then, more video sequences taken from real scene are used to evaluate the tracker performance. These videos include background clutter, scale change, similar target and so on.

Experimental setup: The experiments are performed on Intel i5 eight core 1.60GHz CPU with 8GB RAM, Windows 10 Operating System. Software environment: Matlab 2019b plus OpenCV 4.0.

At first, we choose video sequences from VOT2019 to conduct quantitatively evaluation different methods. Target position is labeled as ground truth bounding box. The tracking performance is measured through center location error (CLE), which is defined as the average Euclidean distance between the center of the estimated target and the ground-truth. The results are reported in Table 1. Our method achieves relatively good results while maintaining high speed.

| DSST\[4\] | MOSSE\[1\] | KCF\[2\] | ECO\[7\] | Ours |
|---|---|---|---|---|
| Median CLE | 15.1 | 35.1 | 23.5 | 12.9 | 14.3 |
| Median FPS | 43 | 289 | 62 | 1.5 | 103 |

Figure 3 and Figure 4 show more results of our tracking method. These videos contain occlusion, scale change, illumination change and so. Our algorithm can successful track the target and achieve satisfied results.

Figure 3. The video includes scale change

Figure 4. More Tracking Results
6. Conclusion
This paper focuses on fast target tracking method used in real-time application. The method is based on simplified kernelized correlation filter. In stead of gray feature, we used HOG feature descriptor to describe target feature to boost tracking stability. In order to improve tracking performance, PCA is used to simplify the HOG computation, which can reduce computation and storage resource abundantly. Our algorithm is implemented to Texas Instruments (TI) multi-core digital signal processor (DSP) TMS320 C6657 platform, which can achieve more than 60 fps. A series quantitative and qualitative experiments are carried out to show superior of proposed method.

References
[1] D. S. Bolme, J. R. Beveridge, B. A. Draper, Y. M. Lui. (2010) Visual object tracking using adaptive correlation filters. In: IEEE Computer Society Conference on Computer Vision and Pattern Recognition. San Francisco. pp. 2544-2550.
[2] J. Henriques, R. Caseiro, P. Martins, J. Batista. (2015) High-speed tracking with kernelized correlation filters. In: IEEE Transactions on Pattern Analysis and Machine Intelligence. pp. 583-596.
[3] M. Danelljan, G. Hager, F. Shahbaz Khan, M. Felsberg. (2014) Accurate scale estimation for robust visual tracking. In: BMVC.
[4] M. Danelljan, G. Hager, F. Shahbaz Khan, M. Felsberg. (2017) Discriminative Scale Space Tracking. In: IEEE Transactions on Pattern Analysis and Machine Intelligence.
[5] M. Danelljan, A. Leonardis, J. Matas, et. Al. (2016) The Visual Object Tracking 2016 Challenge Results. In: European Conference on Computer Vision work-shop.
[6] H. Nam, B. Han. (2015) Learning Multi-domain Convolutional Neural Networks for Visual Tracking. In: CoRR. pp. 4293-4302.
[7] M. Danelljan, A. Robinson, F. Shahbaz Khan, M. Felsberg. (2016) Beyond Correlation Filters: Learning Continuous Convolution Operators for Visual Tracking. In: European Conference on Computer Vision. pp. 472-488.
[8] Yuxuan Sun, Chong Sun, Dong Wang, You He, Huchuan Lu. (2019) ROI Pooled Correlation Filters for Visual Tracking. In: CVPR 2019. pp. 5783-5791.
[9] Kenan Dai, Yunhua Zhang, Dong Wang, Jianhua Li, Huchuan Lu, Xiaoyun Yang. (2020) High-Performance Long-Term Tracking with Meta-Updater. In: CVPR2020.
[10] TEXAS INSTRUMENTS. (2012) TMS320C6655/57 Fixed and Floating-Point Digital Signal Processor Data Manual. https://www.ti.com.cn/cn/lit/ds/symlink/tms320c6655.pdf.
[11] TEXAS INSTRUMENTS. (2012) Multicore Programming Guide. https://www.ti.com.cn/cn/lit/an/sprab27b/sprab27b.pdf.
[12] Xilinx. (2019) 7 Series FPGAs Packaging and Pinout Product Specifications User Guide V1.18. https://www.xilinx.com/support/documentation/user_guides/ug475_7Series_Pkg_Pinout.pdf.