Implementation and Optimization of Template Matching Algorithm for FT-M6678

Jiangtao Hu, Jiandong Shang, Yun Chen, Mengyuan Wang, Xiaonan Chai

ABSTRACT

Feiteng FT-M6678 (hereinafter referred to as M6678) DSP is a multi-core high-performance DSP with completely independent intellectual property rights. M6678 adopts the Harvard architecture and the new KeyStone multi-core architecture that store instructions and data separately. In general, the multiple classic image processing classic algorithms including correlation matching are not efficient for M6678 DSP architecture. To promote the application of domestic DSP chips in the field of image processing and artificial intelligence. Here, the correlation matching algorithm based on the correlation coefficient is transplanted to the M6678 platform, combining the algorithm characteristics and the architecture characteristics of the target platform, performance optimization in terms of parallelism and locality is carried out. The performance of the program is obviously improved, and the unique computing resources of the platform are more fully utilized, which is meaningful for the transplantation and optimization of other image processing algorithms on the platform.

KEYWORDS
Fei DSP, FT-M6678, Related Matching Algorithm, Implementation, Optimization.

INTRODUCTION

With the rapid development of science and technology, more and more new technologies have emerged in the field of information processing, especially image information processing. Image matching technology is one of the most important technologies. Its application is very extensive. Recently, image matching technology has been applied in computer vision technology, aerial mapping, missile guidance, and medical image registration[1].

With the continuous development of DSP technology, a large number of data operations of image matching processing can be realized under real-time conditions, and the technology of image target extraction, detection, recognition and tracking can be realized. TI's series of products is the current leader in the DSP field. The main function of TI's Ti6678 is to apply visual processing systems in the embedded field[2].

Jiangtao Hu, Jiandong Shang, Yun Chen, Mengyuan Wang, Xiaonan Chai
School of information Engineering, Zhengzhou University, Zhengzhou, China
Many excellent DSP microprocessor manufacturers have also been born in China, one of which is a completely self-made high-performance DSP chip called FT-M6678, which uses the new KeyStone multi-core architecture and enhanced core M66x, and supports fixed-point/floating-point processing. Contains 8 FT-M66x DSP cores, each core operating frequency is 1GHz. Fixed-point arithmetic performance: 32 GMAC@1GHz, floating-point arithmetic performance: 16 GFLOPS@1GHz. Domestic DSP boards are in the early stages of development. It is of great significance to continue the corresponding software environment and the adaptation and optimization of the function library for the realization of complete independent control of the hardware and software[3].

The main research work of this paper is to realize the transplantation and optimization of the relevant matching algorithm on the new high-performance multi-core DSP chip of the Feiteng platform[4]. Through the research and analysis of the Feiteng platform and the transplantation and optimization of related matching algorithms, the verification of the correctness of the domestic high-performance chip application and the current popular image processing field and the optimization of the program for the domestic platform have been completed. Explore the alternative controllability of TI chips and chips[5].

INTRODUCTION TO RELATED MATCHING ALGORITHMS

Suppose the target image to be searched is T, the length and width of the image are m and n; the image to be searched (large image) is I, and the image length and width are M and N; and M is greater than or equal to m, N Less than or equal to n. Assuming that the length and width of the sub-image of the image to be searched are m and n, and the upper left vertex of the sub-image is (x, y), assuming that the sub-image is, the template coefficients of the sub-block and the template image are defined as[6]:

\[
\rho(x, y) = \frac{\sum_{k=1}^{m} \sum_{l=1}^{n} I_{(k,l)} - \bar{I}_{x,y} [T(k,l) - \bar{T}]}{\sqrt{\sum_{k=1}^{m} \sum_{l=1}^{n} (I_{(k,l)} - \bar{I}_{x,y})^2} \sqrt{\sum_{k=1}^{m} \sum_{l=1}^{n} [T(k,l) - \bar{T}]^2}}
\]

(1)

Where \( I_{x,y,(k,l)} \) represents the gray value of the sub-image \( I_{x,y} \) at (k, l) point, \( T(k,l) \) represents the gray value of the target image T at (k, l) point, and \( \bar{I}_{x,y} \) represents the gray value of the sub-image, \( \bar{T} \) respectively represents the gray value of the template image T Mean.

It can be seen from Equation (1) that the large calculation amount of the template matching algorithm lies in the fixed point (x, y). During the sliding of the image to be searched, the template coefficients need to be recalculated. The number of template coefficient calculations is \((M-m+1)\times(N-n+1)\). We simplify the numerator and denominator in Equation (1) by product and double integration respectively to obtain Equation (2).
\[
\rho(x,y) = \frac{R(x,y) - TR(x,y)}{\sqrt{R(x,y) - \frac{1}{mn} R^2(x,y) \sum_{k=1}^{m} \sum_{l=1}^{n} T^{*}(k,l) - mnT}}
\]

(2)

Equation (2) and the search position \((x, y)\) template have the following three formulas, as follows

\[
R_1(x, y) = \sum_{k=1}^{m} \sum_{l=1}^{n} I(x + k, y + l) T(k, l)
\]

(3)

\[
R_2(x, y) = \sum_{k=1}^{m} \sum_{l=1}^{n} I(x + k, y + l)
\]

(4)

\[
R_3(x, y) = \sum_{k=1}^{m} \sum_{l=1}^{n} I^2(x + k, y + l)
\]

(5)

\(R(x,y)\) of Equation (3) is the correlation between the sub-image \(I_{x,y}\) and the target image \(T\). According to the correlation theorem, the expression (2) can be converted into the product of the conjugate of the Fourier transform of the image \(I\) to be searched and the Fourier transform of the target image \(T\). Then find the inverse Fourier transform of the product, as shown in the following formula:

\[
R_1(x, y) = \text{IFFT}\{I(u,v) T^*(u,v)\}
\]

(6)

In the above formula \(I(u,v)\), it is the Fourier transform of the image \(I\) to be searched, and \(T^*(u,v)\) is the conjugate of the Fourier transform of the target image \(T\). Because the gray value in the image is a real number, you can quickly calculate and and construct a complex function image

\[
Z(x, y) = I(x, y) + iT(x, y)
\]

(7)

According to the conjugate symmetry principle of the two-dimensional FFT, the following formula can be derived

\[
I(u, v) = [Z(u,v) + Z^*(M-u, N-v)]/2
\]

(8)

\[
T(u, v) = -i[Z(u,v) - Z^*(M-u, N-v)]/2
\]

(9)

Through the simplification operations of formulas (8) and (9), it can be seen that and can be obtained by adding and subtracting the complex function image \(Z\), and the operation process is simplified. Compared with the traditional algorithm, every time the matching position is changed, it needs to be calculated once, and all the
template coefficients can be obtained at once by the complex function image. It is the integral of the gray value of the sub-image pixel and the integral of the square of the target image. The algorithm transplanted in this paper uses integral graphs to simplify calculations, and the calculation process will not be described in detail here.

**ALGORITHM IMPLEMENTATION AND PROGRAM OPTIMIZATION**

**Platform Overview**

FT-M6678 is a high-performance military multi-core floating-point DSP. It is a flagship product of multi-core DSP launched by the National University of Defense Technology. It supports fixed-point/floating-point processing[7]. The DSP has a wealth of coprocessors and high-performance I/O (Input/Output) components and storage. FT-M6678 contains 8 cores, each core operating frequency is 1GHz. Fixed-point arithmetic performance: 32 GMAC@1GHz, floating-point arithmetic performance: 16 GFLOPS@1GHz. It has certain advantages in performance, and can be used in radar signal processing, precision guidance, image processing, electronic countermeasures, sonar detection, secure communications and flight control[8].

FT-M6678 adopts the new KeyStone multi-core architecture and enhanced core M66x. FT-M66x CorePac DSP block diagram is shown in Figure 1 below. It is mainly divided into 8 parts in function: CorePac core, extra-core storage system, interconnection network, high-speed interface, low-speed interface, integration Peripherals, global control registers, bootstrap reset[9].

![Figure 1. FT-M6678 FT-M66x DSP block diagram.](image-url)
Program Implementation

The flow chart of program implementation of this algorithm is shown in the following figure:

![Flow chart of program implementation](image)

Figure 2. Implementation flowchart of related matching algorithm.

After the program is implemented, the main function modules are as follows:

- **void main(void):** Define the real-time image (that is, the reference image) and the template image, and call the MatchTemplate() function in an infinite loop and time it.

- **TemplateMatch(…):** The core function implements the above algorithm. Note that the number of rows and columns of the template coefficient image has been calculated in the main function, so pass the number of rows and columns as a parameter to MatchTemplate. The core of TemplateMatch is also the most time-consuming function in theory. It is used to calculate the mutual template image of two images, that is, the image composed of the value of the R1 expression. The input and output are the same as TemplateMatch, but only the partial result is stored in the result address. First calculate the DFT of the template image, and then calculate the DFT of each real-time image sub-block in turn and perform the inverse Fourier transform of the product of the two DFTs to obtain the mutual template value. The core function of this function is the MulSpectrums(…) function. The MulSpectrums(…) function multiplies two frequency domain images (all elements are complex numbers).

- **IntegralGraph(…):** Called by MatchTemplate to calculate the integral graph (integral image) and square integral graph of real-time image (the integral graph obtained by taking the square of each element and calculating)

- **IntegralGraph(…):** Called by MatchTemplate to calculate the integral graph (integral image) and square integral graph of real-time image (the integral graph obtained by taking the square of each element and calculating)
MeanSdv(...): Called by MatchTemplate to calculate the mean and standard deviation of the template image. The purpose is to calculate the T mean above the formula and the content in the bottom right root. The specific calculation is implemented by the internal Mean_StdDev_8u_C1R(...) function.

CorrCross(...): Use DFT to calculate the template matrix (R1).

Principle of Image Block Optimization

In formula (10) of this chapter, \( R^{(x,y)} \) in the calculation of the correlation coefficient matrix analyzed by the correlation matching algorithm, the relevant image is first padded to the dimension of the image to be searched, and then \( T^{(u,v)} \) the conjugate of the matrix after the DFT transformation of the relevant image is obtained, and then the searched \( I^{(u,v)} \) image is obtained. After the DFT transformation of the matrix, the two matrices are subjected to matrix multiplication. Finally, according to the correlation theorem, the inverse fast Fourier transform of the matrix multiplied by the matrix multiplication results in a matrix of correlation coefficients \( R^{(x,y)} \). The formula is as follows: 512*512.

\[
R^{(x,y)} = \text{IFFT} \{ I^{(u,v)} T^{(u,v)} \} 
\]

(10)

This method of finding all correlation coefficient matrices at once is simple to implement, but the shortcomings are also very obvious. You need to fill in 0 for the related images first, adding redundant calculations. In the process of calculating the entire image, the data is large. When the image data exceeds the FT-M6678's L2 cache size[11], the locality of memory access is poor, and the cache hit rate will be relatively low. In order to solve this problem, this paper proposes the image segmentation optimization for FT-M6678[10].

When the image to be searched is large, and the image data itself exceeds the FT-M6678's L2 cache size, the locality of memory access becomes worse, and the cache hit rate will be lower[11]. In order to solve this problem, this paper proposes an image blocking optimization for FT-M6678 based on the cache line size of FT-M6678. The following figure shows the implementation framework and schematic diagram of image segmentation optimization for domestic platforms. Suppose this paper expands the related image to p*p dimension, and then finds the conjugate of the matrix after the DFT transform of the expanded related image; \( M_1 \) is expressed as the lower right of the (x, y) point in the image to be searched, The
lower right corner is a rectangular sub-image block \( M_1(x, y) \) of length and width \( p \). FFT transformation of \( M_1(x, y) \) is obtained by multiplying the two matrices, and inverse fast Fourier transform of the result matrix of the matrix multiplication results in the correlation matrix \( \text{corr}(n_2\cdot i, n_2\cdot j) \). The length and width of the matrix are both \( n_2 \); where \( n_2=m_1-64+1 \), \( i \) and \( j \) represent the control variables of the rows and columns in the image partition, \( i \) and \( j \) and the nested loop control variables, the initial value of 0, the step of the control variable The length is \( n_2 \), and the loop-out condition is that \( i \) is greater than or equal to \( M+1-m[12] \).

![Input with search image I, template image T](image)

Select the image block dimension \( m_1 \times m_1 \) dimension according to the band search image \( M \) and template image

Extend the template image to 0, dimension and perform FFT transform to get \( T(u, v) \)

Loop to determine whether to traverse the entire search image

YES

FFT THE SUB-IMAGE TO GET \( I_{M_1}(x, y) \)

\( T(u, v) \) and \( I_{M_1}(x, y) \) are multiplied by two matrices

Find the correlation coefficient matrix of sub-image and template image by IFFT

End

Figure 4. Image blocking frame.
Vector Optimization of Core Algorithms

The FT-M6678 platform supports SIMD extension technology. Its extended instruction component is provided with 16 128-bit vector registers, which are integer and floating-point common registers. Each vector register can store two 32-bit floating-point or integer data. Therefore, after SIMD vectorization, one instruction can simultaneously access two floating-point data, which improves the execution power of the code and the parallelism of the data. The Feiteng platform provides rich SIMD instructions. You can use the SIMD instruction to rewrite the source program to the vectorization of the Feiteng platform. For example, use the _mem8_f2 instruction to simultaneously access two float-type data, and use the _dmpysp instruction to complete the multiplication of two float-type data:

The SIMD instruction is rewritten as follows:
```c
int i;
float *ary1 = (float*)0x0c300000;
float *ary2 = (float*)0x0c380000;
for(i=0;i<1024;++i){
    ary1[i]=i;
    ary2[i]=i*3;
    ary2[i]=ary1[i]*ary2[i];
}
```

The example source program is to multiply two arrays. For example, the _mem8_f2 instruction can simultaneously access two consecutive float-type data of a certain starting address, and the _dmpysp instruction can be used to complete the multiplication of two sets of float-type data. It can also be seen from the modification process that the vectorization process of SIMD is often also related to loop optimization. This example is also a typical example of loop expansion.

This paper also uses loop expansion optimization to improve instruction-level parallelism, and uses branch elimination and branch extrapolation to enhance the SIMD vectorization effect.

EXPERIMENTAL RESULTS AND ANALYSIS

Test Plan

The relevant matching program of this subject is compiled in the integrated development environment of CCS5.5.2, and then run on the FT-M6678 platform. In the correlation matching program, the correlation coefficient results are placed in the specified memory location. Through the function options in the ccs integrated development environment, view the correlation coefficient results, and verify the correctness of the program after the transplantation through the correlation coefficient results. Pass and x86 platform results to complete the correctness test after transplantation.

Add timing functions at the beginning and end of the main(...) function of the relevant matching program, and find the time difference to get the running time of
the program. After each optimization method is completed, the correctness and program execution time are tested separately.

| TABLE I. FT SYSTEM ENVIRONMENT. |
|---------------------------------|
| **CPU model** | TMS320C6678 |
| CPU frequency | 1.0GHz |
| Development environment | CCS5.5.2, DSPLIB |
| Translator | TI C6000 |

**Performance Test after Program Optimization**

In this paper, by adding timing functions at the beginning and end of the main(...) function of the relevant matching program, the running time can be obtained by finding the time difference. Under the premise of satisfying the correctness of the results, the comparison of the optimization results is shown in the following table, where the acceleration ratio is the running time of the program before optimization divided by the time after optimization at this stage, and the second optimization stage uses a certain based on the optimization of the previous stage. Optimization method for optimization.

![Speedup ratio](image)

Figure 5. Optimization effect diagram.

It can be seen from the test results that after the image segmentation optimization, the program image segmentation optimization speedup ratio is 1.43, the vectorization optimization speedup ratio is 1.93, the control flow optimization speed ratio is 1.99, and the cycle optimization speedup ratio is 2.03. After all optimization, the overall effect is 2.03, and the acceleration effect is more obvious.

Text heads organize the topics on a relational, hierarchical basis. For example, the paper title is the primary text head because all subsequent material relates and elaborates on this one topic. If there are two or more sub-topics, the next level head
(uppercase Roman numerals) should be used and, conversely, if there are not at least two sub-topics, then no subheads should be introduced. Styles named “Heading 1”, “Heading 2”, “Heading 3”, and “Heading 4” are prescribed. In the process of calculating the entire image, the data is large. When the image data exceeds the ache cache size of FT-M6678, the locality of memory access is poor, and the cache hit rate will be relatively low. Through image block optimization, redundant calculation is reduced, improve memory access locality and cache hit rate, achieve a speedup of 1.43 times, the optimization effect is obvious. The use of branch elimination and branch externalization methods to eliminate redundant control flow, to avoid hindering the exploration of SIMD vectorization, and use compilation The vector inline instructions provided by the environment manually rewrite the core operation code. After testing, the optimization effect is 1.93, which has a certain optimization effect compared with the optimization effect of 2.07 in the previous paragraph. In the instruction-level parallel process, when the number of instructions is insufficient, the number of instructions in the loop is too small to fill the eight functional components of the FT-M6678 processor, resulting in the inability to fully utilize the instruction level provided by the processor. The advantage of parallelism. Faced with this problem, this paper proposes to use loop expansion optimization. After loop expansion, the number of instructions in the loop body is sufficient. The compiler can schedule between these instructions and choose a relatively good combination of parallel instruction emission, so it can improve the performance of the program. Through loop optimization, the final acceleration effect is 2.03.

Fast and stable correlation matching algorithm has always been a research hotspot in the field of image matching. Correlation matching based on correlation coefficients is one of the most important algorithms in the field of image matching. At present, a variety of classic image processing classic algorithms including correlation correlation matching are not efficient for M6678 DSP architecture. In order to solve this problem, promote the application of domestic DSP chips in the field of image processing and artificial intelligence. In this paper, the correlation matching algorithm based on the correlation coefficient is transplanted to the M6678 platform, combining the algorithm characteristics and the architecture characteristics of the target platform, performance optimization in terms of parallelism and locality is performed. The experimental results show that the optimized program performance is more obvious, and makes full use of the platform's unique computing resources, which has reference significance for the transplantation and optimization of other image processing algorithms on the platform.

CONCLUSION

Fast and stable correlation matching algorithm has always been a research hotspot in the field of image matching. Correlation matching based on correlation coefficients is one of the most important algorithms in the field of image matching. At present, a variety of classic image processing classic algorithms including correlation correlation matching are not efficient for M6678 DSP architecture. In order to solve this problem, promote the application of domestic DSP chips in the
field of image processing and artificial intelligence. In this paper, the correlation matching algorithm based on the correlation coefficient is transplanted to the M6678 platform, combining the algorithm characteristics and the architecture characteristics of the target platform, performance optimization in terms of parallelism and locality is performed. The experimental results show that the optimized program performance is more obvious, and makes full use of the platform's unique computing resources, which has reference significance for the transplantation and optimization of other image processing algorithms on the platform.

REFERENCES

1. Yang, Aolei; Jiang, Weiwei; Chen, Ling. An Adaptive Edge Detection Algorithm Based on Improved Canny[J]. Communications in Computer and Information Science, 2017, Vol. 761: 566-575.
2. Lifflander, J (Lifflander, Jonathan), 1; Krishnamoorthy, S (Krishnamoorthy, Sriram), 2. Cache Locality Optimization for Recursive Programs[J]. ACM SIGPLAN Notices, 2017, Vol. 52(6): 1-16.
3. Efficiency Optimization Methods of the Signal Level Real-Time Simulation Software[J]. Hans Journal of Data Mining, 2016, Vol. 9-14.
4. Chen, Xinhai; Xie, Peizhen; Chi, Lihua; Liu, Jie; Gong, Chunye. An efficient SIMD compression format for sparse matrix-vector multiplication[J]. Concurrency and Computation Practice & Experience, 2018, Vol. 30(23).
5. Gilseok HONG; Seonghyeon KANG; Chang soo KIM. Efficient Parallel Join Processing Exploiting SIMD in Multi-Thread Environments* [J]. IEICE Transactions on Information & Systems, 2018, Vol. 30(23).
6. Yin Songfeng, Wang Yicheng, Cao Liangcai, Jin Guofan, Ling Yongshun. Fast correlation matching based on fast Fourier transform and integral graph[J]. Acta Photonica Sinica, 2010, 39(12): 2246-2250. Kim, D.; Jung, Y.; Ju, Y.; Song, J.. Fast Implementation of LSH with SIMD(Article)[J]. IEEE Access, 2019, Vol. 7: 107016-107024.
7. Tetsushi Kodama; Tomoaki Ishiyama. Acceleration of the tree method with SIMD instruction set[J]. Astrophysics, 2019, Vol. 2.
8. Zhou, H (Zhou, Hao), 1,2; Xue, JL (Xue, Jingling), 1. A Compiler Approach for Exploiting Partial SIMD Parallelism[J]. ACM Transactions on Architecture and Code Optimization, 2016, Vol. 13(1).
9. N. P. Smart (1); F. Vercauteren (2). Fully homomorphic SIMD operations[J]. Designs, Codes and Cryptography, 2014, Vol. 71(1): 57-81.
10. Osorio, Roberto R.; Rodriguez, Gabriel. Truncated SIMD Multiplier Architecture for Approximate Computing in Low-Power Programmable Processors[J]. IEEE Access, 2019, Vol. 7: 56353-56366.
11. Watanabe, H.; Nakagawa, K. M.. SIMD vectorization for the Lennard-Jones potential with AVX2 and AVX-512 instructions (Article)[J]. Computer Physics Communications, 2019, Vol. 237: 1-7.