ACCELERATING CDVS EXTRACTION ON MOBILE PLATFORM

Shen Zhang, Ronggang Wang, Qiusi Wang, Wenmin Wang

School of Electronic and Computer Engineering, Peking University Shenzhen Graduate School

ABSTRACT

The extraction of MPEG-7 Compact Descriptors for Visual Search (CDVS) on most popular mobile devices is slow, which is attributed to the complexity of the algorithm and the limited computing power of the mobile platform. A feasible and straightforward way to accelerate the extraction is to excavate the potential computing power of modern mobile microprocessor. In this paper, we implement a NEON SIMD based data level parallelism and Pthread based multi-thread parallelism scheme to accelerate the CDVS extracting process on a multi-core ARM processor. Experimental results show a speed-up of 3.5*x* on Gaussian convolution, and 2.3*x* on the whole extracting process with the proposed method.

Index Terms— CDVS, extraction, NEON SIMD, multi-thread

1. INTRODUCTION

Mobile devices equipped with a camera are becoming ubiquitous in recent years. Along with this trend is the emerging of large amounts of applications for mobile visual search. Generally speaking, a mobile visual search application sends query information to the server over the network and then receives the result along the inverse path. Conventionally, the query message contains either the whole image or all the features extracted from the image. Both of them are of big data size, thus leading to relatively slow response when the uploading network bandwidth is scarce.

The MPEG-7 Compact Descriptors for Visual Search (CDVS) is the very attempt to establish a standardized technology so as to enable efficient and interoperable design of visual search applications. Compared with the traditional way, a compact descriptor contains only a subset of the features extracted from an image and thus needs much fewer bits to represent the image.

To extract the MPEG-7 compact descriptor for an image, the local features of that image are extracted first and then a subset of the features are aggregated. The approach of feature selection is described in [1]. The local feature extracting process is similar to SIFT [2], which will be discussed in detail in section 3. Putting aside the algorithm details, the extracting process has a high computational complexity, and the time of the process on mobile is even more intolerable. For instance, on a Samsung Galaxy S3 mobile equipped with a four-core ARM microprocessor running at 1.4GHz, it takes more than 1 second to extract a compact descriptor for an image of 500×500 pixels.

Because CDVS is a new technology, the works to accelerate CDVS extraction on mobile are rare. However, several SIFT related optimization methods have been proposed. Blaine Rister et al. [3] implemented Gaussian convolution by utilizing the mobile GPU and openGL for Embedded System (openGL ES). Le Tran Su et al. [4] put forward a more radical accelerating method to replace the float Gaussian convolution with integer Gaussian convolution. We tried both methods above. However, neither of them provided satisfactory results. The former method performed mediocrely on account of the shrinking gap of the floating-point computing power between CPU and GPU. Moreover, the OpenGL ES API seemed more suitable for graphics acceleration than for general-purpose image processing. The latter one achieved an improvement on speed with the sacrifice of data accuracy, leading to bad performance of pairwise matching in our experiments.

In this paper, we propose a method to accelerate the MPEG-7 compact descriptor extracting process by applying a NEON SIMD based data level parallelism and Pthread based multi-thread parallelism scheme.

The NEON co-processor is a vector processor that supports Single Instruction Multiple Data (SIMD) operations. On ARM Cortex^{TM}-A9 architecture, the NEON co-processor contains 16 128-bit registers. One such a register can be used as a container for 16 1-byte data, 8 2-byte data, 4 4-byte data, or 2 8-byte data. By using this data level parallelism, those stages that consume large amounts of CPU time but have relatively simple calculation logic, can be accelerated greatly.

Apart from NEON, another important feature of modern microprocessor is the multi-core architecture, where multi-thread is a way to collaborate task scheduling among different cores. Several multi-thread models are available, such as the POSIX thread (Pthread) model and the OpenMP model. The former is efficient and flexible to use; the later provides a higher level of abstraction, but its performance is not as

Thanks to National Science Foundation of China 61370115, China 863 project of 2015AA015905, and Shenzhen Peacock Plan for funding.

good as the former. So we adopt Pthread in our method. Besides multi-thread models, the inherent parallelizability of the application and the partition scheme for tasks are critical in multi-thread parallelism. In section 3, the strategy to partition tasks is to be stated.

The rest of this paper is organized as follows. Section 2 introduces the pipeline of the MPEG-7 compact descriptor extracting process. Section 3 briefly analyzes the reasons for the most time-consuming stages and describes our proposed method to accelerate those stages. Section 4 presents the experimental results. Finally, this paper is concluded in section 5.

2. COMPACT DESCRIPTOR EXTRACTION

What constitute an MPEG-7 compact descriptor of an image are mainly two elements, one is a selected number of compressed local descriptors and the other is a single global descriptor representing the whole image. As illustrated in Fig. 1, the extracting process includes the following stages [5]:

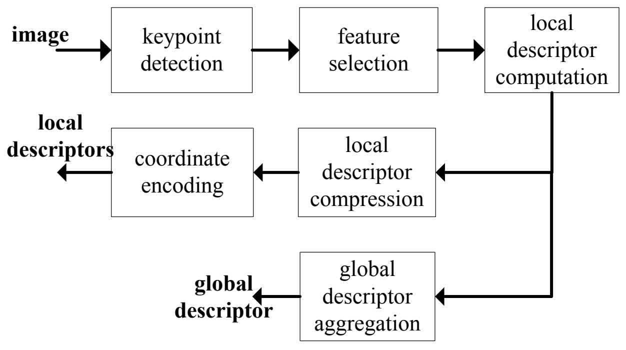


Fig. 1. MPEG-7 compact descriptor extraction pipeline

(a) Keypoint detection: Identifying the ALP (A Low-degree Polynomial) keypoints on a scale space constructed by a set of Laplacian of Gaussian filtered images.
(b) Feature selection: Selecting a subset of keypoints according to their characteristics, such as scale, peak value, orientation, distance from the center of the image, etc.
(c) Local descriptor computation: Computing the descriptors for all the selected features, including gradient calculation, orientation assignment and descriptor computation.
(d) Local descriptor compression: Compressing the selected local descriptors by means of transformation and quantization [6].
(e) Coordinate encoding: Compressing the coordinates of the selected features.
(f) Global descriptor aggregation: Aggregating all the descriptors of the selected features to form a global descriptor.

3. THE PROPOSED ACCELERATING METHOD

Identifying the bottleneck of an application and finding out the inherent parallelizability of those time-consuming stages are the primary tasks before accelerating CDVS extraction.

STAGE	TIME (MS)	PERCENTAGE
KEYPOINT DETECTION	449	34.5%
FEATURE SELECTION	1	0.1%
LOCAL DESCRIPTOR COMPUTATION	770	59.2%
LOCAL DESCRIPTOR COMPRESSION	5	0.4%
COORDINATE ENCODING	4	0.3%
GLOBAL DESCRIPTOR AGGREGATION	71	5.5%
TOTAL	1300	100.0%

Table 1. Time distribution of each stage

As shown in Table 1, the bottleneck of the extracting process lies in the keypoint detection stage and the local descriptor computation stage, whose total time-consumption takes up more than 90% of the whole extracting time.

In the keypoint detection stage, Gaussian convolution consumes 73.3% of the time (Fig. 4). Basically, the scale space of an image $L(x,y,\sigma)$, is computed by applying the convolution of a variable-scale Gaussian $G(x,y,\sigma)$, to each pixel $I(x,y)$:

$$\begin{cases} L(x,y,\sigma)=G(x,y,\sigma)*I(x,y) \\ G(x,y,\sigma)=\dfrac{1}{2\pi\sigma^2}e^{-(x^2+y^2)/2\sigma^2} \end{cases} . \quad (1)$$

In (1), * denotes the convolution operation, x and y are the horizontal and vertical coordinates of a pixel in an image, and σ denotes the size of a scale. Assuming the width and height of an image are equal to M and N respectively, the computational complexity of Gaussian convolution for one scale is $O(M*N*r^2)$, where r denotes the filter radius. Even if the 2-dimensional Gaussian convolution is separated into two independent 1-dimensional convolutions, the computational complexity is still $O(M*N*r)$.

The rest of the keypoint detection stage is the extrema identification by means of ALP approximation, which accounts for 26.7% of the total time.

The local descriptor computation stage involves more complex arithmetic operations, which can be divided into three sub-procedures: gradient calculation, orientation assignment and descriptor computation.

At a scale of a set of Gaussian filtered images $L(x,y)$, the gradient magnitude, $m(x,y)$, and the gradient orientation, $\theta(x,y)$, are computed by using pixel differences:

$$m(x,y)=\sqrt{(L(x+1,y)-L(x-1,y))^2+(L(x,y+1)-L(x,y-1))^2}$$
$$\theta(x,y)=\tan^{-1}((L(x,y+1)-L(x,y-1))/(L(x+1,y)-L(x-1,y)))$$

The orientation of a feature is assigned by picking out the dominant directions (the maximum value, and any other value beyond 80% of the maximum) from an orientation histogram that has 36 bins covering the 360-degree range of orientations within a region around that keypoint. Each sample being added to the histogram is weighted by its gradient magnitude, and by a Gaussian-weighted circular window with a σ that is 1.5 times that of the scale of the keypoint.

The feature descriptor is computed as a set of orientation histograms over 4×4 sub-regions around that keypoint, with each sub-region corresponding to one histogram. The histograms are relative to the keypoint, and each histogram contains 8 bins, with the value of each bin being the sum of the gradient magnitudes near that direction within the region. This process generates a feature vector with 4×4×8=128 elements [7].

Accelerating strategies for the two most time-consuming stages in this paper are proposed based on the above-mentioned analyses.

(1) Keypoint detection stage. Before applying NEON SIMD to Gaussian convolution, the border of an image is extended to eliminate the conditional statement required to handle edge pixels (the reduction of extra conditional statement can save 23% of the time cost in the initial implementation of Gaussian convolution).

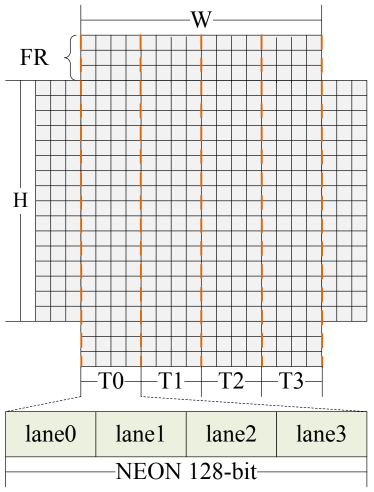


Fig. 2. The strategy used to accelerate Gaussian convolution. *W* and *H* denote the width and height of the image, *FR* is the abbreviation for Filter Radius, which equals to the filter length divided by 2, and *T0-T3* represent threads. In one image, a thread is responsible to process 4 columns, with each element in the columns mapping to a lane of a 128-bit NEON register. A thread pool implemented with Pthread is used to schedule the threads. Moreover, separated Gaussian filters are used in our experiments, and the results of a 1-dimensional convolution are stored in a transposed order.

The NEON instructions can perform four float operations at the same time. A column-major order Gaussian convolution is adopted so that each time the elements of four continuous columns in the same row can be fed to a NEON register with a single load instruction. The task allocated to a thread in this sub-procedure is to process four continuous columns. Because each core of the processor contains an independent NEON processing unit, theoretically speaking, a four-core microprocessor then is able to process 16 columns simultaneously.

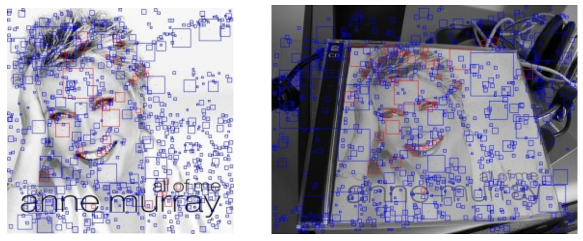
For the rest part of the keypoint detection stage, NEON or multi-thread is applied when possible.

(2) Local descriptor computation stage. The strategy we apply to gradient calculation is analogous to the Gaussian convolution except that NEON is inapplicable here because of frequent trigonometric calculations. So we merely assign the task of processing a single column for a thread and use a thread pool to schedule the tasks.

Unlike Gaussian convolution and gradient calculation, orientation assignment and descriptor computation are keypoint independent. So the reasonable task partition strategy for threads on these sub-procedures is to allocate a single keypoint for a thread. Because of the complex arithmetic logic, NEON SIMD becomes inapplicable once again. Hence, we only use Pthread to parallelize the tasks among keypoints.

4. EXPERIMENTAL RESULTS

We conduct the experiments on a Samsung Galaxy S3 mobile, and the size of the image being tested is 500×500, with 950 ALP keypoints. The results of feature selection as well as the performance after acceleration are shown in Fig. 3-5.



(a) (b)

Fig. 3. Feature selection results (ALP keypoints). (a) is the result of a 500×500 image; (b) is the result of a 640×480 image. The red squares in both images indicate the top 50 features with the highest probability for a successful pairwise matching, and the rest of the squares colored blue indicate the local features that are detected but discarded by the compact descriptor. In the two images, the larger the square, the bigger the scale of the keypoint.

As shown in Fig. 4, with the reference software of CDVS being used as a baseline, the NEON SIMD based parallelism achieves 2.1*x* speed-up in Gaussian convolution, the multi-thread parallelism achieves 1.5*x* and 2.0*x* when the thread number equals to 2 and 4 respectively, and the NEON and 4T combined version achieves 3.5*x* (which is the scheme we adopt at last). The GPU version (implemented with openGL ES) performs faster than the baseline, but much slower than our proposed scheme. In Fig. 5(a), the accelerating results are reached with 4 threads. The speed-ups of gradient calculation and orientation assignment are less than 2.0*x*, while the descriptor computation sub-procedure reaches 2.6*x* (the reason is that the descriptor involves much more compute intensive operations than the other two sub-procedures). Fig. 5(b) shows the speed-up of the whole extracting process with our proposed method achieves 2.3*x*.

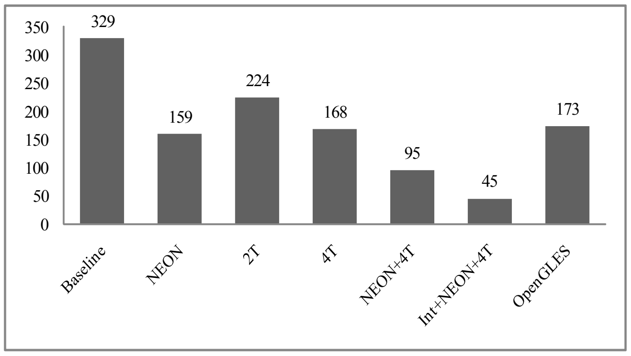
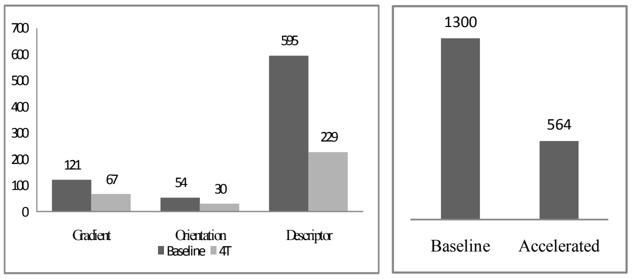


Fig. 4. Time expense of Gaussian convolution under various implementations (Unit: ms)



(a) (b)

Fig. 5. (a) Time expense of three sub-procedures of the local descriptor computation stage; (b) Total time expense of the whole extracting process (Unit: ms)

MODE	7	8	9	10	11
DESCRIPTOR LENGTH	0.5K	1K	2K	4K	8K
FEATURE NUMBER	55	140	190	260	450
MATHCHING RATIO (FLOAT)	0.70	0.84	0.90	0.93	0.94
MATHCHING RATIO (INT)	0.0	0.0	0.55	0.64	0.70

Table 2. Pair-wise matching results of the two images in Fig. 3 under different bit rates defined by CDVS. Different mode numbers indicate different bit rates, and different bit rates correspond to different feature numbers aggregated to the compact descriptor. In the table above, the results depending on integer Gaussian convolution perform much worse than on float Gaussian convolution.

We also explore to replace float Gaussian convolution with integer Gaussian convolution. Although the floating-point arithmetic provides high precision for image data, it also costs more CPU time. By using integer Gaussian convolution, the computing time is reduced greatly (in Fig. 4, "Int+NEON+4T"). However, as implemented in integer Gaussian convolution, keypoints with a small peak value (typically, smaller than 1) cannot be well detected. and the loss of those keypoints results in worse pair-wise matching performance. Table 2 provides a comparison between the two schemes.

5. CONCLUSION

In this paper, we have proposed a NEON SIMD based data level parallelism and Pthread based multi-thread parallelism scheme to accelerate the extraction of MPEG-7 Compact Descriptors for Visual Search (CDVS). Significant speed-up is achieved by applying a hybrid accelerating strategy in the keypoint detection stage and the local descriptor computation stage. In addition, the advantage and disadvantage of integer Gaussian convolution are clarified by the experiments. It can be expected that with technological progress of mobile hardware, the proposed scheme will provide further performance improvement in the future.

6. REFERENCES

[1] G. Francini, S. Lepsøy, and M. Balestri, "Selection of local features for visual search," Signal Processing: Image Communication, vol. 28, no. 4, pp. 311–322, 2013.

[2] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," International Journal of Computer Vision, vol. 60, no. 2, pp. 91–110, 2004.

[3] B. Rister, G. Wang, M. Wu, and J. R. Cavallaro, "A fast and efficient SIFT detector using the mobile GPU," in IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP), pp. 2674–2678, 2013.

[4] L. T. Su, P. J. Ghang, and J. S. Lee, "Integer Gaussian convolution with cache memory for real-time processing of the scale invariant feature transform algorithm," In International Forum on Strategic Technology, pp. 298–301, 2007.

[5] The CDVS Standardization Group, "Test model 10: compact descriptors for visual search," ISO/IEC JTC1/SC29/WG11, N14393, 2014.

[6] L.-Y. Duan, J. Lin, J. Chen, T. Huang, and W. Gao, "Compact descriptors for visual search," IEEE Multimedia Magazine, pp. 30-40, 2014.

[7] Q. Zhang, Y. Chen, and Y. Zhang, "SIFT implementation and optimization for multi-core systems," In Parallel and Distributed Processing, IEEE International Symposium on, pp. 1-8, 2008.