J Real-Time Image Proc (2019) 16:1025–1036
DOI 10.1007/s11554-016-0594-y

ORIGINAL RESEARCH PAPER

Realization of CUDA-based real-time registration and target localization for high-resolution video images

Xiyang Zhi[1] · Junhua Yan[2,3] · Yiqing Hang[2] · Shunfei Wang[2]

Received: 27 September 2015 / Accepted: 15 April 2016 / Published online: 4 May 2016

Abstract High-resolution video images contain huge amount of data so that the real-time capability of image registration and target localization algorithm is difficult to be achieved when operated on central processing units (CPU). In this paper, improved ORB (Oriented FAST and Rotated BRIEF, FAST, which means "Features from Accelerated Segment Test", is a corner detection method used for feature points extraction. BRIEF means "Binary Robust Independent Elementary Features", and it's a binary bit string used to describe features) based real-time image registration and target localization algorithm for high-resolution video images is proposed. We focus on the parallelization of three of the most time-consuming parts: improved ORB feature extraction, feature matching based on Hamming distance for matching rough points, and Random Sample Consensus algorithm for precise matching and achieving transformation model parameters. Realizing Compute Unified Device Architecture (CUDA)-based real-time image registration and target localization parallel algorithm for high-resolution video images is also emphasized on. The experimental results show that when the registration and localization effect is similar, image registration and target localization algorithm for high-resolution video images achieved by CUDA is roughly 20 times faster than by CPU implementation, meeting the requirement of real-time processing.

Keywords Image registration · Target localization · High resolution · Video images · CUDA

✉ Junhua Yan
yjh9758@126.com

1 Research Center for Space Optical Engineering, Harbin Institute of Technology, Harbin 150001, Heilongjiang, China

2 College of Astronautics, Nanjing University of Aeronautics and Astronautics, Nanjing 210016, China

3 Science and Technology on Electro-optic Control Laboratory, Luoyang 471009, China

1 Introduction

Accurate target localization by effective image registration algorithm, which plays a significant role in target recognition and machine vision navigation, has received extensive attention in the field of computer vision.

Constructing and matching descriptors in SIFT-based [1] (SIFT means "Scale-Invariant Feature Transform", and it's an algorithm to detect and describe local features) or SURF-based [2] (SURF means "Speeded Up Robust Features", and it's a local feature detector and descriptor partly inspired by SIFT) methods are very complicated and memory consuming; therefore, it is very difficult to apply them for real-time image registration and target localization for high-resolution video images by CPU [3]. In recent years, local binary features have received wide attention and been researched in depth due to the simple structure, low memory requirements and fast feature extraction and matching. Research led by Xu et al. [4] showed that it was possible to realize image stabilization using image registration algorithm based on ORB with average cost of about 125 ms in 640×480 video images, 2–3 times faster than based on SIFT. Research led by Li et al. [5] used image registration algorithm based on ORB to realize target detection with average cost of about 30 ms in 352×288 video images, about six times faster than based on SURF.

The use of CUDA [6] parallel architecture to achieve speedup for image processing algorithm for high-resolution video images can reduce computing time and realize real-time image registration and target localization [7, 8]. Research led by Tian et al. [9] implemented SIFT feature extraction and matching algorithm using CUDA and achieved 30–50 times speedup compared with CPU implementation. Research led by Heymann et al. [10] accelerated SIFT feature extraction algorithm by graphics processing unit (GPU) with the speed of 20 frames/s in 640 × 480 video images. However, the real time of registration for high-resolution video images has not yet been fully achieved. Therefore, this paper focuses on the research of image registration and target localization parallel algorithm based on binary features, and on further improvement of CUDA parallel algorithm [11] through optimizing bandwidth and data access mode [12], and finally on the realization of real-time image registration and target localization for high-resolution video images.

2 Research of image registration and target localization parallel algorithm

High-resolution digital video image contains such a huge amount of data that when binary feature-based image registration and target localization algorithm operated on CPU is applied, enormous time is cost and real-time implementation is hard to be realized. This paper focuses on 3 of the most time-consuming part of the algorithm: binary feature extraction, feature matching based on Hamming distance and RANSAC [13] algorithm, and on performing them through CUDA for real-time image registration and target localization for high-resolution video images.

2.1 Research of image registration and target localization parallel algorithm based on binary feature

Binary feature-based image registration and target localization algorithm is mainly divided into four parts: (1) Binary feature extraction; (2) Feature matching based on Hamming distance; (3) RANSAC algorithm for matching pure points and achieving transformation model parameters; (4) Coordinate transform to locate the target. The whole process of the algorithm is illustrated in Fig. 1.

ORB [14] feature is generated by FAST [15] key point detection and BRIEF [16] feature description, providing a good real-time performance. In addition, the ORB features are invariant to image translation and rotation, and partially invariant to change in illumination and viewpoint. According to the disadvantage that the ORB features are not invariant to image scaling, this paper presents improved ORB feature extraction algorithm.

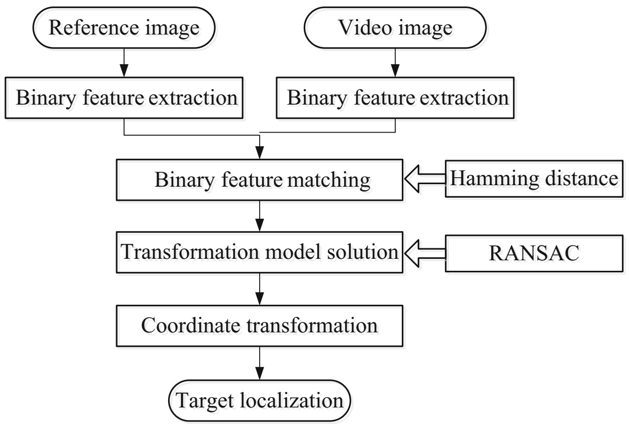
Reference image
Video image
Binary feature extraction
Binary feature extraction
Binary feature matching
Hamming distance
Transformation model solution
RANSAC
Coordinate transformation
Target localization

Fig. 1 Flowchart of binary feature-based image registration and target localization algorithm

2.2 Research of improved ORB feature extraction parallel algorithm

This paper describes an approach to extracting improved ORB features. The image Gaussian Pyramid is firstly constructed to identify key points and feature vectors in each image layer, ensuring that the improved ORB features are invariant to scale. Since all key points and features extracted in each image layer are mapped to the original image, making the location of features too dense and repetitive, repeated points deletion and non-maxima suppression algorithms are further proposed to make sure feature points are evenly distributed and the effect of image registration is improved. Following are the major stages of computation used to generate the improved ORB features: (1) construction of image Gaussian Pyramid. (2) FAST key point detection in each image layer. (3) Coordinate normalization in each image layer of different sizes. (4) Repeated points deletion. Each key point is compared to its corresponding key points in the scale above and below and the key point which has the largest response is preserved. (5) Non-maxima suppression. Each key point is compared to its eight neighbors in the current image and eight neighbors in the scale above and below. It is kept only if its response is larger than all of these neighbors and otherwise it will be deleted. (6) Sort the key points according to FAST and Harris response [17], and pick the top *N* key points where *N* is the number set in advance. (7) Assign the orientation to each key point, describe the rBRIEF feature, and complete the improved ORB feature extraction. The overall steps of the improved ORB feature extraction algorithm are illustrated in Fig. 2.

Corresponding parallel algorithm is further researched, and the parallelization design of the improved ORB feature extraction algorithm based on CUDA is as follows:

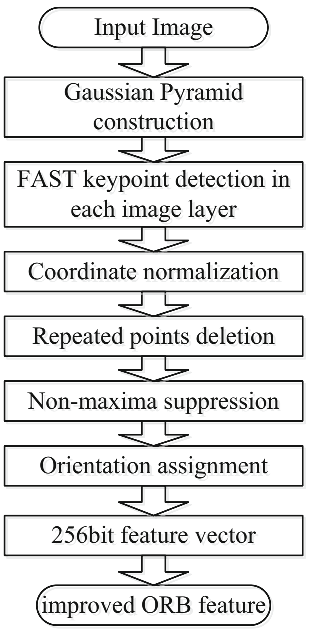

Fig. 2 Flowchart of Improved ORB feature extraction algorithm

1. There is no data communication between image layers of Gaussian Pyramid in the process of FAST key point detection. Therefore, FAST key points are detected parallel in each image layer.
2. FAST key point detection is only associated with the neighbor data in the current image of each pixel and the process is the same, so it can be computed parallel and massively.
3. Repeated point deletion and non-maxima suppression algorithms are only related to the neighbor data in the current image and the neighbors in the scale above and below. Input three layers of image Gaussian Pyramid at the same time and calculate parallel in the same way.
4. For each key point location, the orientation is calculated parallel and based on local image information independently. Then, 256 bit ORB feature is generated.

2.3 Research of improved ORB feature matching parallel algorithm

Improved ORB feature matching algorithm is achieved in two steps as follows.

2.3.1 Feature matching based on Hamming distance for matching rough points

K_1 and K_2 are two improved ORB feature vectors: $K_1 = x_0 x_1 \cdots x_{255}$, $K_2 = y_0 y_1 \cdots y_{255}$. The Hamming distance between K_1 and K_2 is defined as $D(K_1, K_2)$:

$$D(K_1, K_2) = \sum_{i=0}^{255} x_i \oplus y_i \tag{1}$$

where $\oplus$ is the XOR operation, x_i or $y_i (i = 0, 1, \ldots, 255)$ can take the binary values of either 1 or 0. The smaller the value of the Hamming distance $D(K_1, K_2)$ is, the higher the similarity degree between two feature vectors is. Otherwise the similarity degree is lower.

Define *Thr* as the threshold of the Hamming distance. If $D(K_1, K_2)$ is larger than *Thr*, then remove the matching key points corresponding to the two feature vectors. Calculate the Hamming distance of each improved ORB feature vector between the reference image and the image to be registered to get all the rough matching points.

2.3.2 RANSAC algorithm for matching pure points and achieving transformation model parameters

In this paper, the affine transformation is used to describe the changes between images. The point (x, y) in the reference image can be transformed to the point (x_1, y_1) in the image to be registered after the affine transformation.

$$[x_1, y_1, 1]^T = H[x, y, 1]^T \tag{2}$$

where $H = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ 0 & 0 & 1 \end{bmatrix}$ is the transform matrix and a_{ij} represents the parameter in the affine transform matrix.

Let P be the set of the coordinate data of all rough matching points. Then, the parameters of the affine transform matrix can be computed by sampling three matching points from P. The other matching points from P are used to verify the calculated affine transform model. The number of matching points that conforming to this model is added up and the best model which has the most number of matches is determined.

Corresponding parallel algorithm is further researched, and the parallelization design of the improved ORB feature matching algorithm based on CUDA is as follows:

1. The calculation of Hamming distance between two feature vectors is completely independent and needs no data communication, therefore the Hamming distance can be computed parallel and independently.
2. The XOR operation between the 256-dimensional features vectors is only related to the bit data of each dimension, therefore the calculation of it can be independent and parallel.
3. In the process of RANSAC algorithm, the sampling from the coordinate data set P is random and independent. Therefore, each calculation of random sampling is achieved in parallel.

4. When calculating the affine transform model based on sampling data, the calculation process is only related with each sampling data, and each calculation process is independent, so the transform model based on RANSAC can be computed parallel and independently.
5. There is no data communication when checking the correctness of each transform model according to its own parameters and all rough matching points. Therefore, the process of transforming model testing based on RANSAC can be parallel and independent.

3 Real-time implementation of image registration and target localization based on CUDA

In this section, detailed illustration of the implementation of our algorithm based on CUDA is shown. Firstly, the overall framework of our algorithm is shown in Fig. 3. Then, detailed description of parallel optimization and implementation of improved ORB feature extraction is given in Sect. 3.2. Finally, Sect. 3.3 presents implementation of improved ORB feature matching which includes Hamming distance and RANSAC algorithm based on CUDA.

3.1 Parallel optimization and implementation of improved ORB-based image registration and target localization by CUDA

The tasks of CPU and GPU are reasonably divided according to the amount of calculated data, the parallelizability of the algorithm, transmission delay and other factors. The improved ORB feature extraction and matching by CUDA are parallel implemented. In addition, the improved ORB feature matching contains Hamming distance for matching rough points and RANSAC algorithm for precise matching and achieving transformation model parameters. The implementation process of the CUDA-based image registration and target localization is shown in Fig. 3.

3.2 Parallel optimization and implementation of improved ORB feature extraction based on CUDA

Real-time implementation of CUDA-based improved ORB feature extraction is divided into three parts: parallel construction of image Gaussian Pyramid; parallel detection of improved FAST key point; parallel extraction of improved ORB feature vector. The main steps are as follows.

3.2.1 Parallel construction of image Gaussian Pyramid

Determine the size of each image layer. Map the coordinate of each image for thread index where each thread corresponds to an image pixel. Then calculate the Gaussian Pyramid of image in parallel by down sampling and bilinear interpolation.

3.2.2 Parallel detection of improved FAST key point

1. FAST key point detection in each image layer

Map the coordinate of each image layer for thread index, where each block is assigned 32 × 8 threads and the number of blocks is:

$$N_{block\_1} = \frac{W - 6 + dimBlock.x - 1}{dimBlock.x} \times \frac{H - 6 + dimBlock.y - 1}{dimBlock.y} \tag{3}$$

where N_{block_1} is the number of blocks in this step, W and H are width and height of the image. The purpose of this distribution method is to prevent access violation. Each thread corresponds to one pixel, accesses its neighboring information, and parallel detects for the FAST key point. If the condition of FAST key point is met, put the coordinates of the pixel in global memory. At the same time, use the *atomicAdd* function to add up the number of key points in global memory.

Distribute the global memory with the same size of the image layer, which is used to store response of the corresponding key point, and then set it to zero. If the pixel is determined as FAST key point, the response is stored in the corresponding position of the allocated memory.

2. Coordinate normalization in each image layer of different sizes

Distribute the global memory with the same size of the original image as a response map to store the updated response of the key point. To use the execution units effectively, each block is assigned 256 threads and the number of assigned blocks is:

$$N_{block\_2} = \frac{N_{keypoints} + dimBlock.x - 1}{dimBlock.x} \tag{4}$$

where N_{block_2} is the number of blocks in the step of coordinate normalization and $N_{keypoints}$ is the number of key points. In each thread, the normalized coordinates of each key point are calculated and updated on the corresponding location in the response map.

3. Repeated points deletion and non-maxima suppression

Pass the coordinates, response and neighbor response of the key point in current image layer and corresponding

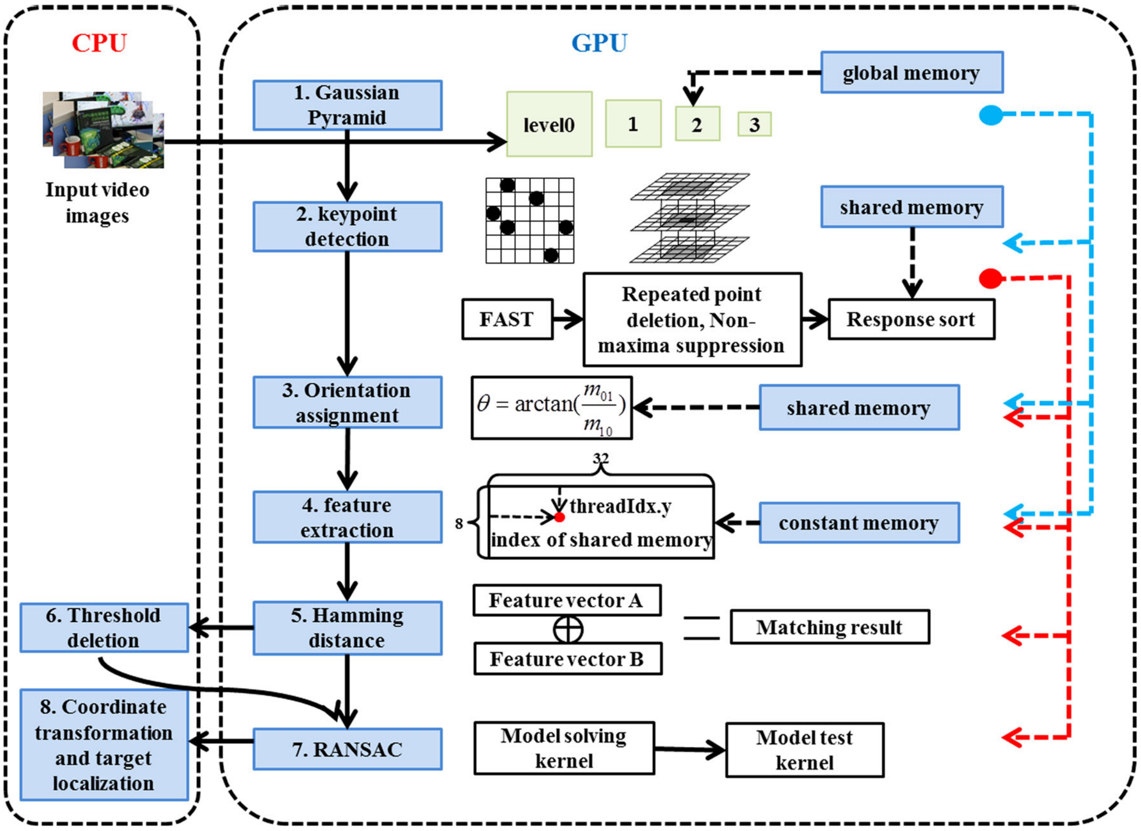

Fig. 3 Flowchart of CUDA-based image registration and target localization algorithm

neighbor response in the scale above and below as arguments to the kernel function of repeated points deletion and non-maxima suppression. Each block is assigned 256 threads and the number of blocks is:

$$N_{block\_3} = \frac{N'_{keypoints} + dimBlock.x - 1}{dimBlock.x} \tag{5}$$

where N_{block_3} is the number of blocks in this step and $N'_{keypoints}$ is the number of key points in current image layer. Each thread corresponds to one key point and reads 27 responses around the key point in three image layers. If the response of the key point is the biggest, retain this key point and remove two corresponding key points in the scale above and below, otherwise remove this key point. Use the *atomicAdd* function to add up the number of key point after repeated points deletion and non-maxima suppression.

4. Sort the key points according to FAST and Harris response

Put the coordinates and response of the key point after repeated points deletion and non-maxima suppression into the kernel function to sort. Implement parallel sort by calling the *sort_by_key* algorithm in Thrust library. Get $(2 \times N)$ key points which have strong responses and N is the number set in advance.

Calculate the Harris responses of these key points and also implement parallel sort by calling the *sort_by_key* algorithm in Thrust library. Then pick the top N key points which have strong responses as the results of improved FAST key point detection.

3.2.3 Parallel extraction of improved ORB feature vector

Firstly, compute the orientation of the key point in parallel. Each block is assigned 32×8 threads and the number of blocks is:

$$N_{block\_4} = \frac{N_{all\_keypoints} + dimBlock.y - 1}{dimBlock.y} \tag{6}$$

where N_{block_4} is the number of blocks in the step of computing the orientation of the key point and $N_{all\_keypoints}$ is the number of all key points. Calculate two neighbor moments *m_01*, *m_10* and store the results in shared memory to improve the efficiency of repeated data access. The thread with the 0 index in each block calculates the orientation of the key point according to the definition of intensity centroid with the values of *m_01* and *m_10* and stores the results in global memory.

Configure the parameters and start the kernel function of feature extraction. Each block is assigned 32 × 8 threads and the number of blocks is:

$$N_{block\_5} = \frac{L + dimBlock.x - 1}{dimBlock.x} \times \frac{N_{all\_keypoints} + dimBlock.y - 1}{dimBlock.y} \quad (7)$$

where N_{block_5} is the number of blocks in this step and L is the length of feature vector.

In this paper, a 256-element improved ORB feature vector is used. The amount of data on each dimension is a bit, so that each feature vector has the length of 32 uchar data. Use thread index to get the serial number of each key point and the ordinal position in the global memory of each feature vector. Generate the feature vector according to the orientation of the key point and the sampling mode in the image block around in each thread. Finally, the results of improved ORB feature vectors are stored in the global memory. The manner of thread index of improved ORB feature vector extraction is shown in Fig. 4.

3.3 Parallel optimization and implementation of improved ORB feature matching based on CUDA

3.3.1 Parallel optimization and CUDA implementation of improved ORB feature matching

In this subsection, CUDA function in the OpenCV Library is used to complete the parallel feature matching algorithm based on Hamming distance. The main steps are as follows:

1. Create a *BruteForceMatcher_GPU_base* object. Call the *matchSingle* function to calculate Hamming distance algorithm between each feature vector in two images. Obtain the maximum Hamming distance value and the corresponding index of each feature vector in the image to be registered from the reference image.
2. Transmit the Hamming distance value and the data matrix of corresponding index from global memory back to CPU host memory.
3. In CPU host, mismatches are removed according to the threshold to get the rough matching points, the coordinates and corresponding index values of the matching points are saved.
4. Load the data of coordinates and corresponding index of the rough matching points from CPU memory into GPU memory, providing data for RANSAC algorithm for parallel calculation of transformation model parameters.

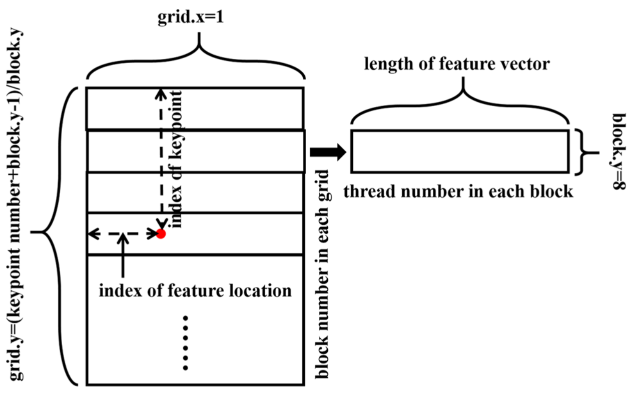
grid.x=1
grid.y=(keypoint number+block.y-1)/block.y
index of keypoint
index of feature location
block number in each grid
length of feature vector
thread number in each block
block.y=8

Fig. 4 Diagram of thread index of Improved ORB feature vector extraction

3.3.2 Parallel optimization and CUDA implementation of RANSAC algorithm

Firstly, generate random numbers in CPU memory according to the index array. Secondly, parallelly implement calculation and test of RANSAC model using CUDA. Finally, select the best transformation model parameters which are in a good agreement in CPU memory. The main steps are as follows:

1. Generate *M* sets of random numbers according to the number of iterations of *M* and each set contains three different random numbers in CPU memory. Load the random numbers, which are invariants in solution process, from CPU memory into constant memory in GPU. Use the caching mechanism of constant memory to improve data access speed.
2. Configure the parameters and start the kernel function of model calculation. The thread with the 0 index in each block gets three sample numbers determined by three random numbers and reads the corresponding coordinates of rough matching points from the constant memory. Then each thread calculates the model parameters with the coordinate data and stores the results in shared memory.
3. Configure the parameters and start the kernel function of model test. Each thread reads the corresponding sample in constant memory and check the match degree with the model in this block. Use the *atomicAdd* function to add up the numbers of samples which are consistent with the model.
4. Transmit the numbers of samples which are consistent with the model in each block from GPU memory to CPU memory. Then select the best model which has the largest number and achieve the pure matching points according to the threshold.

4 Experimental results

In this paper, real-time image registration and target localization algorithm for high-resolution video images based on improved ORB by CUDA is realized. The experiments verified that the parallel algorithm achieved on CUDA is 20 times faster than the serial algorithm achieved on CPU when the effects of image registration and target localization are similar, meeting the requirement of real-time processing. Experimental environment: the computer used was a desktop with six 2 GHz Intel Xeon E5-2620 processors for the CPU, GPU equipped with a NVIDIA Tesla K20C of 0.71 GHz enclosing 1024 threads each block, 3.5 of calculation capacity. The computer was also equipped with the VS2010 configured with OpenCV 2.4.6. In addition, we use CUDA 5.5 in the experiments. The experiment of improved ORB feature extraction was conducted to verify the effect of repeated points deletion and non-maxima suppression. There are three of the most time-consuming parts in the image registration and target localization algorithm: improved ORB feature extraction, feature matching based on Hamming distance and RANSAC algorithm for precise matching and achieving transformation model parameters. The experiments of these three parts were carried out, respectively, to get the computation time of CUDA and CPU when the experimental effects were similar. The total time of image registration and target localization algorithm achieved on CUDA and CPU were measured, respectively, when the experimental effects were similar.

4.1 Improved ORB feature extraction experiment

Two sets of video images of different pixels were selected to be experimented, respectively, of ORB feature extraction and improved ORB feature extraction: (a) 1280 × 720 pixels; (b) 1920 × 1080 pixels. The contrast effects of feature extraction are shown in Figs. 5 and 6.

The numbers of same key points and mutual neighbor key points in the same images after ORB feature extraction and improved ORB feature extraction were compared as shown in Table 1.

The experimental results show that improved ORB feature extraction algorithm can effectively remove the huge amount of repetitive points and excessively dense neighbor points caused by feature extraction in multiple layers of image Gaussian Pyramid. With improved ORB feature extraction algorithm, the key points extracted can be evenly distributed and the accuracy of image registration and target localization is improved. In experimental image (b), there are 263 same key points and 544 mutual neighbor key points out of 1024 key points extracted when using ORB feature extraction, whereas 93 same key points and 254 mutual neighbor key points out of 1024 key points extracted when using improved ORB feature extraction.

4.2 Image registration and target localization experiment

Two sets of video images of different pixels were selected to be experimented, respectively, of image registration and target localization: (a) 1280 × 720 pixels; (b) 1920 × 1080 pixels. There was a certain degree of scaling, rotation and viewpoint change between the reference image and the image to be registered. The contrast effects of image registration and target localization achieved on CPU and CUDA, respectively, are shown in Figs. 7, 8, 9 and 10.

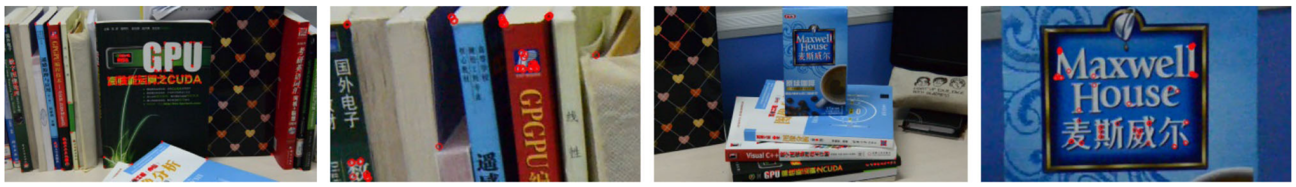
(Ⅰ) 1280×720 original image (Ⅱ) 1280×720 local image (Ⅲ) 1920×1080 original image (Ⅳ) 1920×1080 local image

Fig. 5 Effects of ORB feature extraction

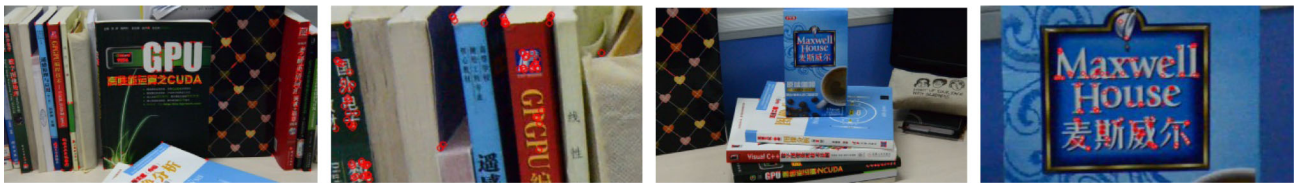
(Ⅰ) 1280×720 original image (Ⅱ) 1280×720 local image (Ⅲ) 1920×1080 original image (Ⅳ) 1920×1080 local image

Fig. 6 Effects of Improved ORB feature extraction

Table 1 Contrast of effect of ORB feature extraction and Improved ORB feature extraction

Image number	Total number of extracted key points	ORB feature extraction		Improved ORB feature extraction	
		Number of same key points	Number of mutual neighbor key points	Number of same key points	Number of mutual neighbor key points
(a)	1024	280	565	108	234
(b)	1024	263	554	93	254

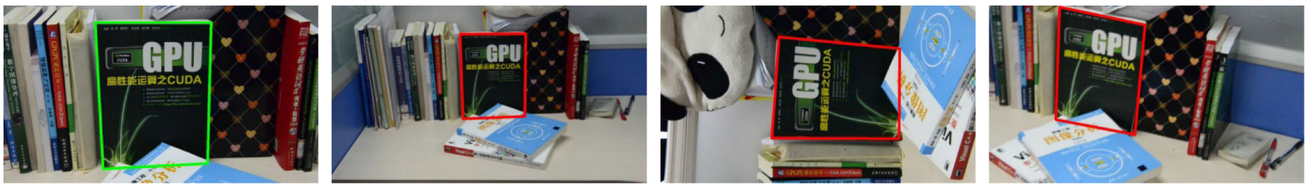
(Ⅰ) reference image (Ⅱ) scaling change (Ⅲ) rotation change (Ⅳ) viewpoint change

Fig. 7 Effects of CPU-achieved image registration and target localization of 1280 × 720 high-resolution video images

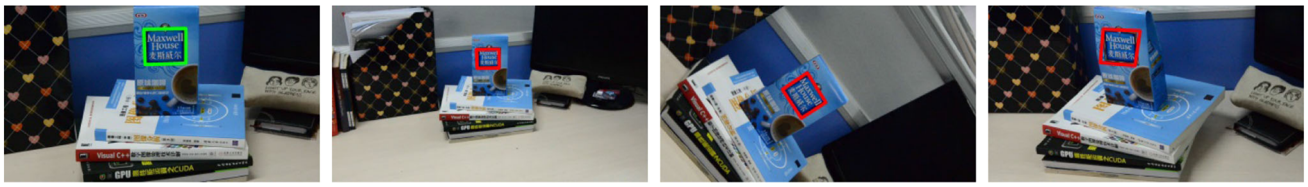
(Ⅰ) reference image (Ⅱ) scaling change (Ⅲ) rotation change (Ⅳ) viewpoint change

Fig. 8 Effects of CPU-achieved image registration and target localization of 1920 × 1080 high-resolution video images

The experimental results show that, in regard of the certain degree of scaling, rotation and viewpoint change of high-resolution video images, improved ORB image registration and target localization algorithm achieved, respectively, on CPU and CUDA can both realize accurate registration and target localization.

In this paper, root mean-square error (RMSE) is used as the evaluation index for the image registration and target localization effect. Each RMSE values between reference image and the image to be registered after image registration and target localization achieved, respectively, on CPU and CUDA in the six sets images were calculated and compared. The numerical comparison is shown in Table 2.

The experimental results show that, in the six sets images, the RMSE values between reference image and the image to be registered after image registration and target localization achieved, respectively, on CPU and CUDA are both minute, and the effect of image registration and target localization based on improved ORB feature achieved by CPU and by CUDA implementation is similar.

4.3 Accelerating experiments based on CUDA

To verify the acceleration of CUDA parallel algorithm, 3 of the most time-consuming parts of image registration and target localization algorithm were experimented, respectively, and when the effect was similar, the computation time difference of algorithm implementation between CPU and CUDA was compared. When effect was similar, the different total computation time of image registration and target localization cost by CPU and by CUDA implementation was compared. Two sets of video images were selected to be experimented: (a) 1280 × 720 pixels, 43 s, 25 frame/s, 1093 frames; (b) 1920 × 1080 pixels, 31 s, 25 frame/s, 796 frames. Some of the representative images in the two sets of experimental video were shown in Figs. 11 and 12.

1. Improved ORB feature extraction

Contrast of performance of improved ORB feature extraction achieved on CPU and CUDA is shown in Table 3.

The experimental results show that CUDA-achieved improved ORB feature extraction is one order of

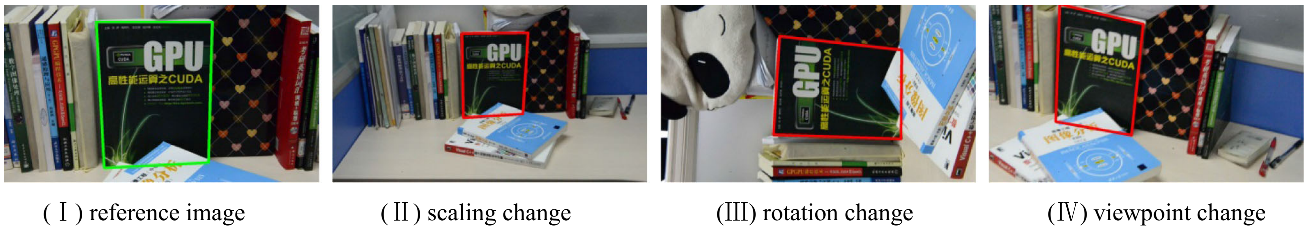

Fig. 9 Effects of CUDA-achieved image registration and target localization of 1280 × 720 high-resolution video images

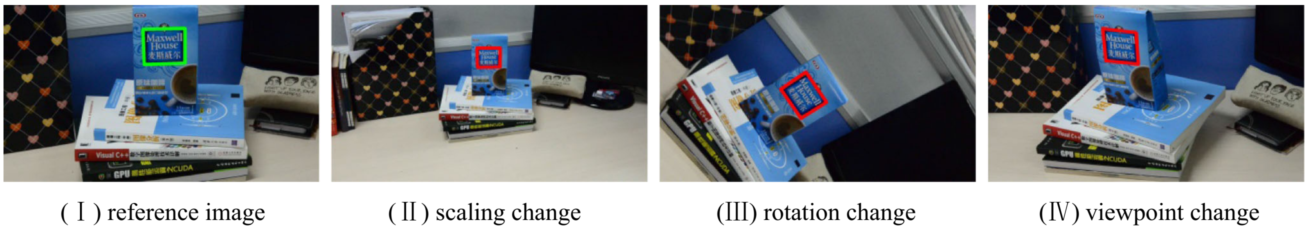

Fig. 10 Effects of CUDA-achieved image registration and target localization of 1920 × 1080 high-resolution video images

Table 2 Contrast of effect of CPU/CUDA-achieved image registration and target localization based on Improved ORB feature

Image change	Image number	RMSE of CPU-achieved registration and target localization	RMSE of CUDA-achieved registration and target localization
Scaling change	(a)	0.0216	0.0205
	(b)	0.0103	0.0088
Rotation change	(a)	0.0030	0.0059
	(b)	0.0019	0.0010
Viewpoint change	(a)	0.0126	0.0085
	(b)	0.0031	0.0035

magnitude faster than its CPU implementation when extracting the same number of key points and feature vectors. As in the video (b), when extracting 512 key points and feature vectors, the average computation time of CPU is 346.6410 ms/frame and that of CUDA implementation is 34.1722 ms/frame and the accelerating rate of CPU to CUDA is up to 10.1439. The accelerating rate will still increase as the image pixels and the number of key points extracted increase.

2. Feature matching based on Hamming distance

Contrast of performances of feature matching based on Hamming distance achieved on CPU and CUDA is shown in Table 4.

The experimental results show that when a same number of key points are input, the feature matching based on Hamming distance implemented on CPU and CUDA can get similar number of average rough matching points. As in the video (a), when 1024 key points were input, CPU and CUDA implementation achieve 746 and 728 rough matching points, respectively. This shows that the effect of the feature matching based on Hamming distance implemented on CPU and CUDA is similar. Moreover, the average computation time of CPU is 26.0010 ms/frame and that of CUDA implementation is 0.8422 ms/frame and the accelerating rate of CPU to CUDA is up to 30.8724. The algorithm using CUDA can achieve one order of magnitude speedup comparing with CPU implementation and the accelerating rate will still increase as the image pixels and the number of matching points increase.

3. RANSAC algorithm for matching pure points and achieving transformation model parameters

Contrast of performance of RANSAC algorithm for matching pure points and achieving transformation model parameters achieved on CPU and CUDA is shown in Table 5.

The experimental results show that when a similar number of rough matching points are input, the RANSAC algorithm for precise matching and achieving transformation model parameters implemented on CPU and CUDA can get similar number of average pure matching points.

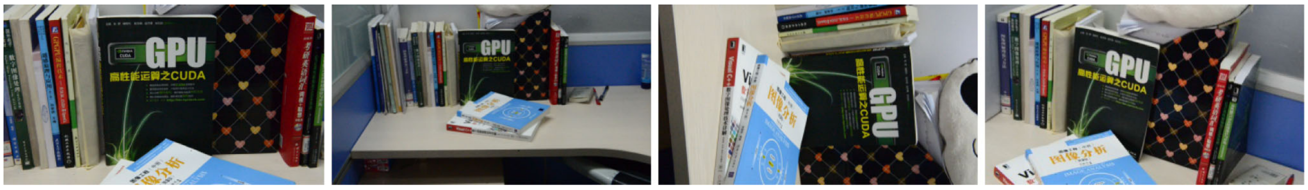
(Ⅰ) 30th frame (Ⅱ) 267th frame (Ⅲ) 723th frame (Ⅳ) 926th frame

Fig. 11 Part of images of 1280 × 720 high-resolution video (a)

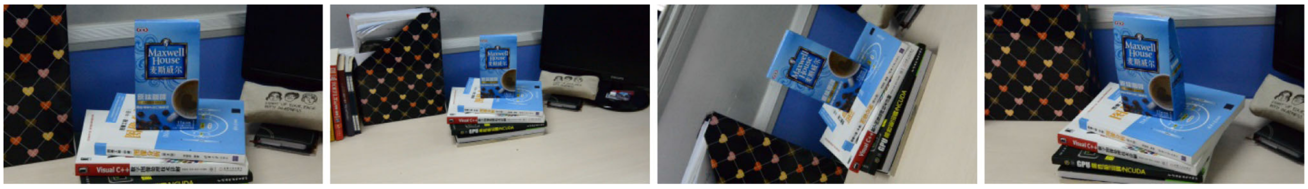
(Ⅰ) 35th frame (Ⅱ) 229th frame (Ⅲ) 448th frame (Ⅳ) 753th frame

Fig. 12 Part of images of 1920 × 1080 high-resolution video (b)

Table 3 Contrast of performance of CPU/CUDA-achieved Improved ORB feature extraction

Video number	Key points in each frame	Average time of CPU-based improved ORB feature extraction (ms/frame)	Average time of CUDA-based improved ORB feature extraction (ms/frame)	Speedup ratio
(a)	512	206.1230	33.2210	6.2046
	1024	221.8500	30.0104	7.3924
(b)	512	346.6410	34.1722	10.1439
	1024	355.1730	31.2002	11.3837

Table 4 Contrast of performance of CPU/CUDA-achieved feature matching based on Hamming distance

Video number	Key points in each frame	CPU/CUDA-achieved average rough matching points in each frame	Average time of CPU-based feature matching based on Hamming distance (ms/frame)	Average time of CUDA-based feature matching based on Hamming distance (ms/frame)	Speedup ratio
(a)	512	234/246	9.8120	0.3927	24.9860
	1024	746/728	26.0010	0.8422	30.8724
(b)	512	232/218	10.9410	0.3632	30.1239
	1024	752/744	29.4470	0.8472	34.7580

As in the video (a), when 1024 key points were input, CPU and CUDA implementation achieved 259 and 256 pure matching points, respectively. This shows that the effect of the RANSAC algorithm for precise matching and achieving transformation model parameters implemented on CPU and CUDA is similar. Moreover, the average computation time of CPU is 231.9470 ms/frame and that of the CUDA implementation is 2.0659 ms/frame and the accelerating rate of CPU to CUDA is up to 112.2741. The algorithm using CUDA can achieve two orders of magnitude speedup compare with CPU implementation and the accelerating rate will still increase as the image pixels and the number of matching points increase.

4. Real-time image registration and target localization algorithm for high-resolution video images based on improved ORB

The total computation time and speedup ratio of improved ORB-based image registration and target

Table 5 Contrast of performance of CPU/CUDA-achieved RANSAC algorithm for matching pure points and achieving transformation model parameters

Video number	Key points in each frame	CPU/CUDA-achieved average pure matching points in each frame	Average time of CPU-based RANSAC algorithm (ms/frame)	Average time of CUDA-based RANSAC algorithm (ms/frame)	Speedup ratio
(a)	512	138/122	143.9110	1.7221	83.5672
	1024	259/256	231.9470	2.0659	112.2741
(b)	512	140/128	323.7640	1.6870	191.4641
	1024	262/249	401.8650	2.0462	196.3958

Table 6 Total time and speedup ratio of CPU/CUDA-achieved image registration and target localization for high-resolution video images

Video number	Key points in each frame	Average time of CPU-based image registration and target localization (ms/frame)	Average time of CUDA-based image registration and target localization (ms/frame)	Speedup ratio
(a)	512	360.3200	35.1634	10.2470
	1024	479.9620	32.7017	14.6770
(b)	512	681.2054	36.8411	18.4904
	1024	759.3550	33.6587	22.5604

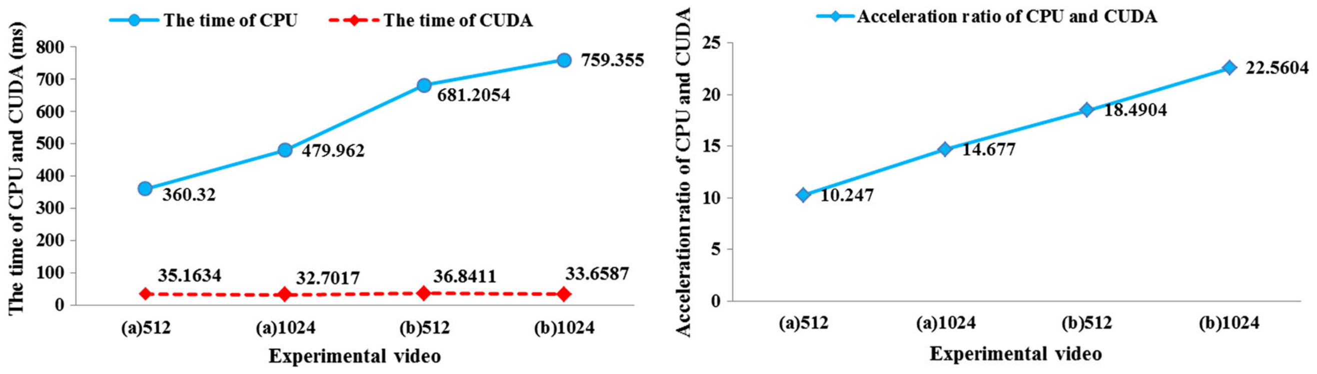

Fig. 13 Total time and speedup ratio of CPU/CUDA-achieved image registration and target localization for high-resolution video images

localization for high-resolution video images achieved on CPU and CUDA are shown in Table 6 and Fig. 13.

It can be seen in Table 6 and Fig. 13 that, when implemented by CPU, the computation time of improved ORB-based image registration and target localization for high-resolution video images is relatively long and increases significantly as image pixels and the number of key points extracted increase. However, when implemented by CUDA, the computation time is shorter and the increase of which is relatively modest, meeting the requirement of real-time processing. The experimental results show that when effect is similar, the improved ORB-based image registration and target localization algorithm for high-resolution video images achieved by CUDA is roughly 20 times faster than CPU implementation and the speedup ratio will still increase with the increase of image pixels and the number of key points extracted. Since improved ORB feature extraction is the most time-consuming part in image registration and target localization algorithm, which takes about 50 % of the CPU time and above 90 % of the CUDA time, the total speedup ratio is close to the speedup ratio of improved ORB feature extraction algorithm.

5 Conclusion

In this paper, improved ORB-based real-time image registration and target localization for high-resolution video images is realized and the difference between the performances CPU and CUDA implementation is compared. The experimental results show that when effect is similar, the algorithm of improved ORB-based image registration and target localization for high-resolution video images achieved by CUDA is 20 times faster than CPU implementation and the speedup ratio will still increase with the increase of image pixels.

Acknowledgments This work was supported by the National Natural Science Foundation of China (61471194), Science and Technology on Electro-optic Control Laboratory and Aeronautical Science Foundation of China (20135152049), China Aerospace Science and Technology Corporation (CASC) Aerospace Science and Technology Innovation Foundation Project, the Fundamental Research Funds for the Central Universities, Nanjing University of Aeronautics And Astronautics Graduate School Innovation Base (Laboratory) Open Foundation Program (kfjj20151505).

References

1. Lowe, D.G.: Distinctive image features from scale-invariant key points. Int. J. Comput. Vis. **60**(2), 91–110 (2004)
2. Bay, H., Tuytelaars, T., Gool, L.V.: Surf: speeded up robust features. Comput. Vis. Image Underst. **110**(3), 404–417 (2006)
3. Sinha, S.N., Frahm, J.M., Pollefeys, M., Genc, Y.: Feature tracking and matching in video using programmable graphics hardware. Mach. Vis. Appl. **22**(1), 207–217 (2011)
4. Xu, J., Chang, H., Yang, S., Wang, M.: Fast feature-based video stabilization without accumulative global motion estimation. Consum. Electron. IEEE Trans. **58**(3), 993–999 (2012)
5. Li, X.: Rapid moving object detection algorithm based on ORB features. J. Electron. Meas. Instrum. **27**(5), 455–460 (2014)
6. Zhang, S., Zhu, Y.L.: GPU High Performance Computing Of CUDA, pp. 58–68. China Water & Power, Beijing (2009)
7. Huang, Y., Liu, J., Tu, M., Li, S., Deng, J.: Research on CUDA-Based SIFT Registration of SAR Image. In: 2011 Fourth International Symposium on Parallel Architectures, Algorithms and Programming, IEEE Computer Society, pp. 100–104 (2011)
8. Yan, J., Hang, Y., Xu, J., Chu, L.: Quick realization of CUDA-based registration of high-resolution digital video images. Chin. J. Sci. Instrum. **35**(2), 380–386 (2014)
9. Tian, W., Fan, X.U., Wang, H.Y., Zhou, B.: Fast scale invariant feature transform algorithm based on CUDA. Comput. Eng. **36**(8), 219–221 (2010)
10. Heymann, S., Müller, K., Smolic, A., Fröhlich, B., Wiegand, T.: Sift implementation and optimization for general-purpose gpu. In: Wscg, pp. 317–322 (2007)
11. Herout, A., Jošth, R., Juránek, R., Havel, J., Hradiš, M., Zemčík, P.: Real-time object detection on CUDA. J. Real-Time Image Proc. **6**(3), 159–170 (2011)
12. Xiao, H., Zhang, Z.X.: Parallel image matching algorithm based on GPGPU. Acta Geodaetica Cartogr. Sin. **39**(1), 46–50 (2010)
13. Fischler, M.A., Bolles, R.C.: Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. Commun. ACM **24**(6), 381–395 (1981)
14. Rublee, E., Rabaud, V., Konolige, K., Bradski, G.: ORB: an efficient alternative to sift or surf. In: Proceedings of the 13th IEEE International Conference on Computer Vision, pp. 2564–2571 (2011)
15. Rosten, E., Drummond, T.: Machine Learning for High-Speed Corner Detection. In: Computer Vision-ECCV 2006, Springer Berlin Heidelberg, pp. 430–443 (2006)
16. Calonder, M., Lepetit, V., Strecha, C., Fua, P.: Brief: binary robust independent elementary features. Comput. Vision-ECCV 2010 **6314**, 778–792 (2010)
17. ‘Corner detection’, http://en.wikipedia.org/wiki/Corner_detection/. Accessed 9 Mar 2016

Xiyang Zhi is an assistant professor in Research Center for Space Optics Engineering of Harbin Institute of Technology. He received the Ph. D. degree in department of Optical Engineering, Harbin Institute of Technology. His current research interests include statistical signal processing, target detection and tracking and image quality assessment technology. Email: zhixiyang@hit.edu.cn.

Junhua Yan was born in 1972. She received her B. Sc. degree, M. Sc. degree and Ph. D. degree all from Nanjing University of Aeronautics and Astronautics in 1993, 2001 and 2004, respectively. Now she is an associate professor in Nanjing University of Aeronautics and Astronautics, a visiting researcher in Science and Technology on Electro-optic Control Laboratory. Her main research interests include multi-source information fusion, target detection, tracking and recognition. E-mail: yjh9758@126.com.

Yiqing Hang received B. Sc from Nanjing University of Aeronautics and Astronautics in 2012. Now she is currently a M. Sc. candidate of Nanjing University of Aeronautics and Astronautics. Her main research direction is target detection and localization. E-mail: hyq58qitian@126.com.

Shunfei Wang was born in 1992. He received B. Sc from Nanjing University of Aeronautics and Astronautics in 2014. Now he is currently a M. Sc. candidate of Nanjing University of Aeronautics and Astronautics. His main research direction is object detection and tracking. E-mail: wangshunfei2010@126.com.