

# Real-time SIFT-based Object Recognition System

Zhao Wang, Han Xiao, Wenhao He, Feng Wen and Kui Yuan

*Institution of Automation  
Chinese Academy of Sciences  
95 Zhongguancun East Road, Beijing, China  
[zhao.wang@ia.ac.cn](mailto:zhao.wang@ia.ac.cn)*

**Abstract –** In this paper a real-time object recognition system is realized, based on the Scale Invariant Feature Transform (SIFT) algorithm. The system mainly contains a display, a camera and an image acquisition and processing board developed by our research team. An FPGA chip and a DSP chip are embedded in the card as the major calculation units, which make real-time computation possible. The whole recognition algorithm is divided into three parts: the detection of SIFT keypoints, the extraction of SIFT descriptors and the final object recognition. In order to achieve real-time detection of SIFT keypoints through hardware computation on FPGA, the original SIFT algorithm is adapted to accommodate the parallel computation and pipelined structure of hardware. Using a mode of DSP invoking a customized FPGA module, a 72-dimensional keypoint descriptor is proposed to save memory space and to cut down the computing cost in keypoints matching. The recognition proceeds by matching individual features to a database of features from known objects using a fast approximate nearest-neighbor search algorithm changed based on the k-d tree and the BBF algorithm. In addition, three matching strategies are adopted to discard the false matches so as to improve the accuracy of recognition. The object recognition functionality is mainly achieved in the DSP. A model database is built and used to test the accuracy and effectiveness of the system.

**Index Terms –** *object recognition, SIFT keypoints, embedded system, k-d tree, BBF algorithm*

## I. INTRODUCTION

Real-time object recognition is a fundamental problem for service mobile robots achieving scene understanding. Normally, general-purpose computer system is used to process the vision information. But the computation ability of the CPU is not enough to handle the algorithms of high complexity such as object recognition algorithms, and the low processing speed is often difficult to meet the needs of real-time service robot. As a customizable logic circuit, Field Programmable Gate Array (FPGA) can accelerate the algorithm by the parallel computation and pipelined structure. But for some complicated algorithms, the flexibility of FPGA is insufficient. As a microprocessor with a kind of special structure, Digital Signal Processor (DSP) can make up for the deficiency of FPGA by its flexible addressing mode. Besides, DSP also has powerful calculation ability. Therefore, using an FPGA chip and a DSP chip as the main computing elements our research team developed an intelligent image acquisition and processing board to realize the real-time computation for complex algorithms.

Traditional object recognition methods are mainly based on the global features of the object, such as moment features

and geometric features, which may not be applied to situations like image rotation and background clutter. But object recognition techniques based on the local features can effectively overcome these difficulties. Over the past decade, some local invariant features [1~3] due to their invariance in image scale, rotation, viewpoint changes and illumination changes have received an increasing attention in image registration, object recognition and other machine vision applications. Among of these, Scale Invariant Feature Transform (SIFT) [3] is one of the most representative methods. However, the high computation complexity often makes it infeasible for real-time applications. Thus, some researchers have explored to implement SIFT with GPU [10], FPGA [7, 8] or ASIC [9] to accelerate the computation. In this paper, the SIFT algorithm is implemented on the embedded image processing board developed by our research team to achieve a real-time object recognition functionality with a low power consumption.

The SIFT-based object recognition algorithm can be divided into three parts: the detection of SIFT keypoints, the extraction of SIFT descriptors and the final object recognition. The first stage contains some easy but computation intensive operations, such as Gaussian filtering and the detection of scale-space extrema, which make it very suitable to be implemented on FPGA. The second stage contains some complex manipulations, which are difficult to implement with pure parallel computation. Therefore, a semi-parallel computation scheme with DSP invoking a customize FPGA module are designed. This scheme implements a great deal of the computation in a pipelined architecture in FPGA while leaves the rest of the computation to DSP. For the third stage, massive memory operation is needed to perform keypoints matching and object recognition, so this stage is completely achieved with DSP.

This paper is organized as follows. After briefly discussing the whole hardware system in the next section, the implementation scheme and detail descriptions of the SIFT-based object recognition algorithm in the intelligent image acquisition and processing board will be described in the Sections III. In Section IV, a new feature database is built from some 3D models. In the following section, the redesigned object recognition system is tested under various environmental conditions with this database. Section VI concludes the paper.

## II. THE EMBEDDED IMAGE PROCESSING SYSTEM

The hardware system (see Fig. 1 (I)) contains a display, a camera and an image acquisition and processing board (see

Fig. 1 (II)). The object image is taken by the camera and is input into the intelligent image card, where operations like down-sampling, Gaussian filtering, detecting SIFT keypoints, extracting and matching of SIFT descriptors and object recognition are performed. The recognition results are shown on the display. If there is any object that could be matched with an object in the database, the two objects will be shown on the display at the same time (see Fig. 1 (III)). If not, the display only shows the current image taken by the camera.

The intelligent image acquisition and processing board (see Fig. 1 (II)) is the core of the whole system, which is designed by WenHao He. The FPGA used on the card is Altera Cyclone III EP3C40F484, and the DSP on the card is TI's TMS320DM642. In addition, there are two  $512\text{K} \times 16$ bits SRAM, two  $4\text{M} \times 32$ bits SDRAM, one  $32\text{M} \times 16$ bits DDR and a  $4\text{M} \times 8$ bits Flash on the card.

The card also contains two analog video input ports and one analog video output port. Analog video signal of PAL format is converted to BT.656 digital signal by the A/D converter TVP5150. Then the digital signal is transferred into the FPGA. The pins of the FPGA are connected with the two SRAMs, where the image data and results could be saved. Besides, the pins of DSP and FPGA are connected, which makes DSP convenient to access the memory resources in the FPGA and the SRAM on the board. The logic circuit in the FPGA can also receive and process the data sent by DSP.

The video processed by the present system is PAL format. And the size of standard PAL format digital image is  $720 \times 576$ . The interlaced scanning requires that the odd field is sent first, then the even field. In order to detect keypoints through a pipelined architecture in FPGA, our system only processes the odd field (with horizontal down-sampling), so the actual size of the image is  $360 \times 288$ .

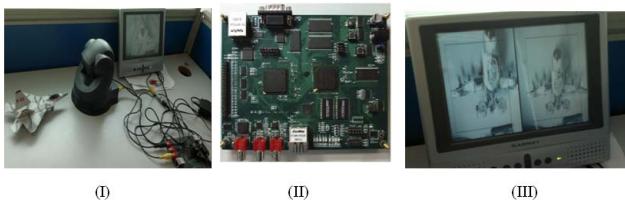


Fig. 1 The embedded system: (I) The experimental system; (II) The intelligent image acquisition and processing board; (III) The results of matching.

### III. SIFT-BASED OBJECT RECOGNITION ALGORITHM

After introducing the hardware system, this part will describe the detailed realization of object recognition algorithm on the embedded system. The overall implementation scheme of the algorithm is shown in Fig. 2. As already mentioned, the SIFT-based object recognition algorithm can be divided into three stages. The first stage identifies keypoints over all scales and image locations, which is completely achieved on FPGA. The second stage generates a descriptor for each keypoint based on a local image patch in its neighborhood, which is realized by calling the FPGA module in the DSP. And the final stage searches through the

database to find a possible match for each descriptor and performs the final object recognition, which is implemented on the DSP.

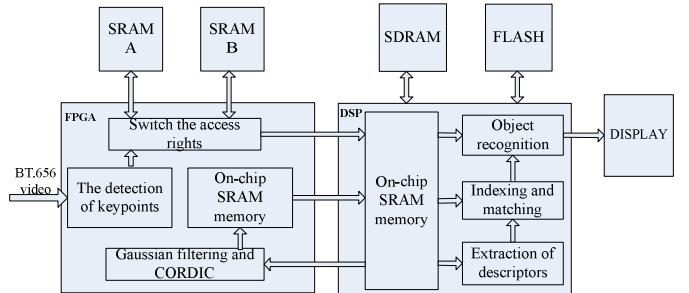


Fig. 2 The overall implementation scheme

#### A. The detection of SIFT keypoints

According to [3], the major procedures of detecting SIFT keypoints are as follows:

- 1) Perform cascaded Gaussian filtering on the gray image to build (one octave of) the Gaussian pyramid.
- 2) Subtract between adjacent layers of the pyramid to produce the difference-of-Gaussian images.
- 3) Extract the scale-space extrema from the difference-of-Gaussian images to get potential interest points.
- 4) Check the stabilities of the candidate interest points (with Hessian Matrix and so on) to get the final keypoints.
- 5) Resize the image and repeat the above procedures to build more octaves of the pyramid and detect keypoints on more scales.

We adapted Lowe's algorithm to implement it in a pipelined architecture that fully exploits the parallel computing capability of FPGA. Refer to [4, 8] for details.

Fig. 3 shows the keypoints detected from a scene by our system, wherein the rectangle indicates the region of interest set by DSP. To demonstrate that our adapted algorithm implemented in hardware has similar performance with its software counterpart, we compare the results from FPGA with the results obtained in MATLAB. Table I shows the experimental results of 12 images. The coincidence rates indicate that most of the keypoints detected by FPGA coincide with those detected by MATLAB.

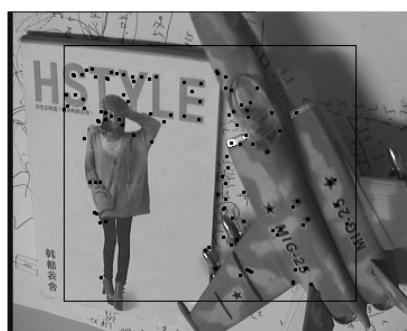


Fig. 3 Keypoints detected by our system

TABLE I  
COMPARISON BETWEEN FPGA AND MATLAB RESULTS

	Scale	FPGA	MATLAB
Total number of keypoints from 12 images	1	761	604
	2	418	436
	3	329	346
Coincidence rate between keypoints detected by FPGA and MATLAB (%)	1	75.7	95.4
	2	98.3	94.3
	3	86.3	82.1

### B. The Extraction of SIFT descriptors

Extracting SIFT descriptors, as described in [3], is implemented by identifying the dominant orientations for each keypoint and building a descriptor based upon the image gradients in its local neighborhood. The standard keypoint descriptor used by SIFT is a 128-element vector, which is very distinctive but rather complex and time consuming in computation. In this work, we design a 72-dimensional new descriptor to replace the 128-dimension SIFT descriptor. The aim is not only to accelerate the calculation and save the memory, but also to realize a new matching criterion. The procedures of generating the 72-dimensional descriptor with the cooperation of FPGA and DSP can be summarized as follows.

1) Whenever the DSP receives an interrupt signal from the FPGA, it reads the locations of all keypoints (detected by FPGA) and the  $360 \times 288$  raw image into its internal memory.

2) The DSP selects a keypoint and extracts a  $26 \times 26$  pixel matrix from around the keypoint in the raw image with bilinear pixel interpolation according to the keypoint scale.

3) The DSP feeds the  $26 \times 26$  pixel matrix into FPGA, and the pixel stream is processed by a pipeline in the FPGA, which convolves the  $26 \times 26$  image patch with a  $9 \times 9$  Gaussian filter and computes gradients on the resultant  $18 \times 18$  matrix, generating a  $16 \times 16$  amplitude matrix and a  $16 \times 16$  orientation matrix which are fed back to the DSP.

4) The DSP weights the  $16 \times 16$  amplitude matrix with a cone-shaped function (instead of a Gaussian window) to weaken the edges and to emphasize the center, and then computes the amplitude-weighted orientation histogram and derives the main orientation from it.

5) The DSP rotates the  $16 \times 16$  amplitude matrix and orientation matrix according to the main orientation, and then divides the rotated  $16 \times 16$  matrix into 9 overlapping subregions (Fig. 4). From each of the subregion, an  $8 \times 8$  cone-shaped function is applied on the amplitude matrix and an 8-bin amplitude-weighted orientation histogram is generated.

6) The DSP combines the 8-bin histograms from the 9 subregions, resulting in a 72-dimensional vector. Then the 72-dimensional vector is normalized with the infinity-norm which is more efficient in computation than Euclidean norm. After normalization, the value of each element of the descriptor is an integer in  $0 \sim 255$ .

Similar to the calculation of norm, the distance between two descriptors generated with the above method is defined by the infinity-distance. For two descriptors  $Des_1 = (a_1, a_2, \dots, a_{72})$  and  $Des_2 = (b_1, b_2, \dots, b_{72})$ , the infinity-distance is:

$$\|Des_1 - Des_2\| = \max \{ |a_1 - b_1|, |a_2 - b_2|, \dots, |a_{72} - b_{72}| \}$$

Since the dimension of descriptors is reduced from 128 to 72 and infinity-norm is used instead of Euclidean norm, the memory and time required in computation are decreased a lot. With DSP invoking a pipeline inside FPGA, the average time for generating one descriptor is 1.25 ms, which is much faster compared with the 11.7 ms reported in paper [8]. Meanwhile, the distinctiveness of this 72-dimensional descriptor is powerful enough for object recognition, which will be demonstrated with experiments in Section V.

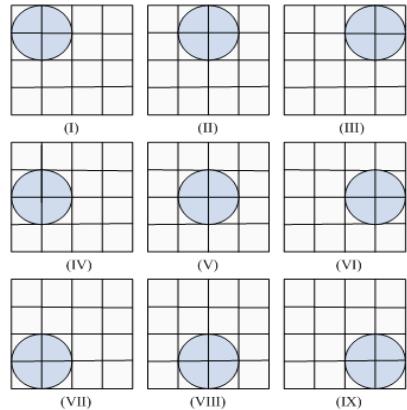


Fig. 4 Nine subregions in a patch used to construct descriptor

### C. Object recognition

For object matching and recognition, SIFT features are first extracted from a set of reference objects to build a database. The detailed information is introduced in Section IV. A new object is matched by individually comparing each feature from the new object to this previous database and finding candidate matching features based on infinity-distance of their feature vectors. If at least 5 features of the new object are correctly matched with the features belong to the same object in the database after discarding false matches, then these two objects pair is termed a match.

As described in [3], the SIFT descriptors are highly distinctive, which allows a single feature to find its correct match with good probability in a large database of features. So the best candidate match for each feature is found by identifying its nearest neighbor in the database of features from training objects. Here the nearest neighbor is defined as the feature with minimum infinity-distance for the invariant descriptor vector as we described in the previous section.

Exhaustive search can get the correct nearest neighbor, but it is very time-consuming. So Lowe introduced the k-d tree [5] to accelerate this stage at first. However, the k-d tree provides no speedup over exhaustive search for more than 10 dimensional spaces. Therefore, Best-Bin-First (BBF) is used by Lowe [6], which is approximate in the sense that it returns the closest neighbor with high probability. However, although Lowe's BBF algorithm is available in software, it also contains some time-consuming and complex operations that are not beneficial to implement with hardware, such as the

Hough transform and the heap-based priority queue. So a modified fast approximate nearest-neighbor algorithm is proposed here. Following are the major stages of recognition proposed by the paper:

1) Extract features from the training objects to build a feature database. Besides, add the locations and dominant orientations of these features to the database.

2) Build a chain using the feature descriptors from the database. Every node of the chain contains three information: the order number of the keypoint in the object, the order number of the object in the database and the order number of the next node connecting with the current node in the chain.

3) Change the chain to a k-d tree. The detailed stages will be discussed below in greater detail, especially the different part with the standard k-d tree algorithm.

Until now, database has been built, which seems to be similar with the Lowe's method. However, the detailed stages of building the k-d tree are different. Firstly, the calculation of variance needs  $72 \times n$  multiplications, which is time-consuming for DSP. So an approximate calculation is used:

$$V_j = 1/N * \sum_{i=1}^N |d_{ij} - \mu_j|,$$

where  $\mu_j$  is the mean value. Secondly, split the data space with mean value instead of medium value used in the standard k-d tree algorithm, this is because searching the medium value is time-consuming and mean value has obtained after variance is calculated. In addition, in a sense this will make similar vectors gathered more compact. In that point this is beneficial to searching the nearest neighbor. Although this change will make no feature vector saved in the nodes of k-d tree, and only the leaf nodes store feature vectors, which will lead to the increase of the depth of the tree and further lead to the increase of the required memory but yet backtracking procedure could be deleted, which will save time greatly.

After building the k-d tree, next step is indexing the tree to search the nearest neighbors for features of a new object. Comparing with the value of maximum dimension saved in every node, finally find a leaf node for the current feature vector. Then identify the nearest neighbor by comparing the distance with 50 leaf nodes (about 7000 nodes in the tree) around the leaf node. The leaf node corresponding to the minimum distance is called the candidate nearest-neighbor. Here the distance still means infinity-distance.

However, many features from a new object will not have any correct match in the training database because they arise from background clutter or are not detected in the training objects. Therefore, it would be useful to have some strategies to discard features that do not have good match to the database. In this paper, three strategies are adopted. Firstly, a global threshold on distance to the closest feature is used. Secondly, a more effective measure is obtained by comparing the distance of the closest neighbor to that of the second-closest neighbor. This method performs well because correct matches need to have the closest neighbor significantly closer than the closest incorrect match to achieve reliable matching [3]. Finally, a new method called point pair method proposed

by the paper is used. This method is based on a fact that the coordinates of every keypoint in a point pair will be changed during object recognition, but the mutual position does not change. Fig. 5 shows a point pair in the same object, keypoint A and keypoint B.  $R_A$  and  $R_B$  are their dominant orientations.  $\theta_A$  is the angle between  $R_A$  and  $\overrightarrow{AB}$ ;  $\theta_B$  is the angle between  $R_B$  and  $\overrightarrow{BA}$ . If  $\hat{\theta}_A$ ,  $\hat{\theta}_B$  represent the angles of the matching point pair of A and B, then these two point pairs have the same location relationship means  $\hat{\theta}_A = \theta_A$  and  $\hat{\theta}_B = \theta_B$ . In the experiments, as long as the difference of them is less than a threshold  $30^\circ$ , these two point pairs will be considered as the correct matches.

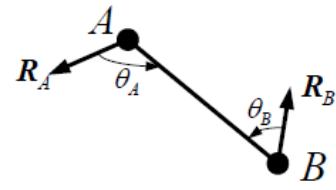


Fig. 5 A point pair

#### IV. A MODEL DATABASE

In order to test the real-time system, a feature database detected from airplanes and car models are built. The size of this database is about 7000, coming from 71 different models. Fig. 6 shows the features in the database and objects that the features belong to. Although the complexity of the database seems lower than some famous data set, such as Caltech-101 used in pure software, the objects used in this system are 3D models. Bigger change of rotation to these 3D models will completely change the content. So we try to use the same view to capture the object under different experiments to make sure the most content captured from the same object under different conditions is coincident. The average number of features detected from every object is about 100. The maximum feature number that every object can detect is 127, which is limited by the memory of our system. And the maximum object number that the memory can store is 128, which can be increased by upgrading system.

Because the on-chip memory of DSP used in our system does not contain FLASH and EEPROM, the data and program stored in the memory will all lost after cutting down the power. So a FLASH memory is expanded in the external of the DSP chip. The capacity of the FLASH memory is  $4M \times 8$ bits. The feature database is saved in the FLASH. A program writing the database into the FLASH memory and a program reading the database to the DSP memory from FLASH memory are programmed, respectively.



Fig. 6 The feature database

## V. EXPERIMENTS AND RESULTS

As discussed above, we use the infinity-distance between two feature vectors to determine whether the two vectors belong to the same keypoint in different objects. The threshold set is a trade-off between the number of false matches and the total number of matches. Table II shows the results of matching an image (see Fig.7) using different threshold parameters, where the results of different distance ratio were made on condition that the global threshold was set to 175, and “Yes” in the third part means using the point pair method, “No” means the opposite. The premise of this part was that global threshold was set to 175 and the distance ratio was set to 0.85. It is clearly proved that three strategies are effective in improving the accuracy of matching. By the influence of the camera resolution and the simplicity of the model objects, the number of keypoints detected from some objects is very small. Therefore, a little larger threshold is set to make sure enough points are left for matching in later recognition experiments.

TABLE II  
RESULTS USING DIFFERENT THRESHOLD

Threshold		Total number	Error number	Error rate of matches
Global	150	55	5	9.1%
Threshold	175	64	8	12.5%
Distance	200	71	12	16.9%
Distance	0.8	50	1	2%
Ratio	0.85	55	3	5.4%
Ratio	0.9	60	5	8.3%
Point	Yes	46	0	0
Pair	No	55	3	5.4%

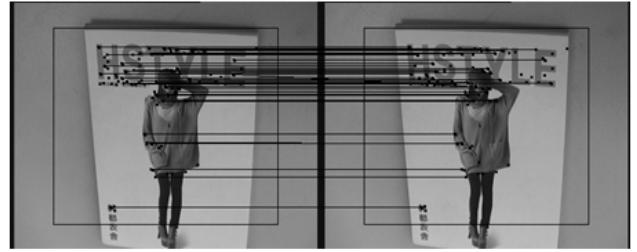


Fig. 7 The matching results to an image: global threshold is 175, the distance ratio is 0.85 and the point pair method is used.

In addition, we ran many experiments to test the performance of the proposed system, which consist of examining each descriptor’s robustness to effects caused by the changes in illumination, viewpoints and scales, and the addition of background clutter. Besides, an experiment without any change was made as the comparison criteria. The correct recognition rate is used to quantify our results as the evaluation metric. The new database is used in these experiments.

Table III shows the average results of some models by testing 10 times under every condition. And every result is made by the following stages. Firstly, we reject all matches in which the distance with the nearest-neighbor is greater than 300 and the distance ratio of closest to second-closest neighbors is greater than 0.9. Then remove the points that do not meet the point pair matching condition with its adjacent three keypoints. And if fewer than 5 points remain after discarding outliers, the match is rejected. The first column of the table is the order number of models in the database. The second refers to the total keypoints number of the models. And the middle five columns refer to the correct recognition rate under the five conditions. Because the first 43 models in the database is already very small, the changing in scales has nearly no influence on the images captured by our camera. So the results of this part have been omitted. The last three columns are the average results of many experiments. The error rate means the rate of the current object being mistaken as other object, while the false rejection rate refers to the rate of finding no match for the current object.

Although the results of several models like the 15 are not so satisfactory, the table still clearly proves that the method can give acceptable result in most scenes. The reasons leading to bad results contain that some models themselves are very dark and simple that few keypoints can be detected, and some models are very similar that they may be mistaken. Nevertheless, it is necessary to point out that the range of rotation in depth for reliable recognition is only about 30 degrees for 3D objects. So 3D object recognition is best performed by integrating features from multiple views, which will be one of the contents in my future work. In addition, the SIFT descriptors to illumination change is partly invariant. It is because that the number of keypoints detected from the objects will be very little when the light is very dark, which will greatly influence the quantity of matching.

TABLE III  
RESULTS UNDER SOME CONDITIONS

Model number		0	15	25	35	45	55	65
Sum of keypoints		127	87	71	92	127	86	78
Correct rate under some conditions (%)	Illumination	30	20	50	50	100	90	70
	Scales	*	*	*	*	100	100	80
	Rotation ( $<30^\circ$ )	50	40	90	100	100	60	30
	clutter	60	30	90	100	100	80	100
	No changes	100	100	100	100	100	100	100
Average results of all tests (%)	Correct rate	60	40	100	100	100	86	76
	Error rate	10	0	0	0	0	0	0
	False rejection rate	30	60	0	0	0	14	24

Finally, some results of correct matching are shown in the Fig. 8, which are displayed by MATLAB after the related data is transferred into the host computer. From these figures we can see that not only most models can be correctly recognized but also most of the keypoints in these models can find their correct nearest neighbor, even the background is very messy and some models are very similar.

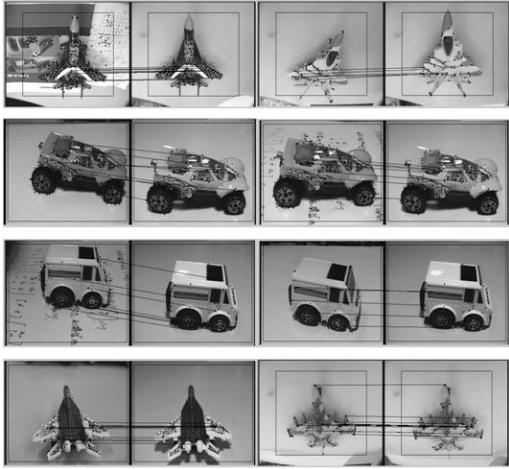


Fig. 8 Some matching results under some conditions: the left image is the current object and the right image is the image stored in the database.

The total time to recognize an object is about 200ms on the 600MHz DSP, which is less than 0.3s on a 2GHz Pentium 4 processor got by Lowe in [3]. The total time of capturing an image and detecting keypoints is about 20ms, the extraction of descriptors is about 110ms, and the time of recognition is 60ms. The stage of extracting descriptors occupies over 55% of the whole execution time, although the dimension of each descriptor has fallen to 72 from 128, further improvement to this stage is still need.

## VI. CONCLUSIONS

This paper designs a real-time SIFT-based object recognition system and implements SIFT algorithm on a low-power embedded image processing board with an FPGA and a DSP as the major calculation units. Experiments show that the

time of recognizing an object from a database of 7000 features is about 200ms on the 600MHz DSP, faster than Lowe's 0.3s on a 2GHz Pentium 4 processor. At the same time, the good performance of SIFT is kept.

The paper splits the algorithm into three stages and mainly describes the differences from Lowe's standard algorithm and the implementation details on embedded system. Firstly, SIFT keypoints are detected from objects, which is completely implemented on the FPGA. Then, 72-dimensional instead of Lowe's 128-dimensional SIFT descriptors are extracted with the DSP invoking a customized FPGA module. Finally, a fast approximate nearest-neighbor search algorithm is used to recognize objects under various conditions from a database built from 3D models.

In the future, the system will be used on service robots, and depth information will be added to further improve the performance of recognition.

## ACKNOWLEDGMENT

The work is supported by National Natural Science Foundation of China (No.61273360, No.61203328), and Key Project of S&T Plan of Beijing Municipal Commission of Education (No.KZ201210005001). Besides, the authors would like to thank the authors of references and the anonymous reviewers for improving the quality of this paper.

## REFERENCES

- [1] H. Bay, A. Ess, T. Tuytelaars, and L.V. Gool, "SURF: Speeded Up Robust Features," *In Computer Vision and Image Understanding*, vol. 110, PP. 346-359, 2008.
- [2] K. Mikolajczyk, C. Schmid, "Scale and affine invariant interest point detectors," *International Journal of computer vision*, Vol. 60, no. 1, pp. 63-86, 2004.
- [3] D.G. Lowe, "Distinctive Image Features from Scale-Invariant Keypoints," *International Journal of Computer Vision*, vol. 60, no. 2, pp. 91-110, 2004.
- [4] Han Xiao, Kui Yuan, Wenhao He, "An improvement on the Keypoint Detection Method of SIFT for Hardware Computation," *The 8<sup>th</sup> World Congress on Intelligent Control and Automation*, Taipei, Taiwan, pp. 63-68, 2011.
- [5] M. Muja, D. G. Lowe, "Fast approximate nearest neighbors with automatic algorithm configuration," *In Int. Conf. on Comp. Vision Theory and Applications*, 2009.
- [6] Beis, Jeff, David G. Lowe, "Shape indexing using approximate nearest-neighbour search in high-dimensional spaces," *Conference on Computer Vision and Pattern Recognition*, Puerto Rico, pp. 1000-1006, 1997.
- [7] V. Bonato, E. Marques, G. A. Constantinides, "A Parallel Hardware Architecture for Scale and Rotation Invariant Feature Detection," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 18, no. 12, 2008.
- [8] Han Xiao, Kui Yuan, Wenhao He, et al., "SIFT keypoints detection based on FPGA," *High Technology Letters (Chinese)*, vol. 22, pp. 429-435, 2012.
- [9] K. Joo-Young , K. Minsu, L. Seungjin, et al., "A 201.4 GOPS 496 mW Real-Time Multi-Object Recognition Processor With Bio-Inspired Neural Perception Engine," *IEEE Journals & Magazines*, vol. 45, No. 1, pp. 32-45, 2010.
- [10] K. Junchul, P. Eunsoo, C. Xuenan, et al., "A fast feature extraction in object recognition using parallel processing on CPU and GPU," *IEEE International Conference on Systems*, pp. 3842-3847, 2009.