2014 International Joint Conference on Neural Networks (IJCNN)
July 6-11, 2014, Beijing, China

Content-Based Image Retrieval by Dictionary of Local Feature Descriptors

Patryk Najgebauer, Tomasz Nowak, Jakub Romanowski, Marcin Gabryel,
Marcin Korytkowski and Rafał Scherer, *Member, IEEE*

***Abstract*— This paper describes a novel method of image keypoint descriptor indexing and comparison used to speed up the process of content-based image retrieval as the main advantage of the dictionary-based representation is faster comparison of image descriptors sets in contrast to the standard list representation. The proposed method of descriptor representation allows to avoid initial learning process, and can be adjusted taking into consideration new examples.**

The presented method sorts and groups components of descriptors in the process of the dictionary creation. The ordered structure of the descriptors dictionary is well suited for quick comparison of images by comparing their dictionaries of descriptors or by comparing individual descriptors with the dictionary. This allows to skip a large part of operations during descriptors comparison between two images.

In contrast to the standard dictionary, our method takes into account the standard deviation between the image descriptors. This is due to the fact that almost all descriptors generated for the points indicating the same areas of the image have different descriptors. Estimation of the similarity is based on the determined value of the standard deviation between descriptors.

We assume that proposed method can speed up the process of descriptor comparison. It can be used with many solutions which require high-speed operations on the image e.g. robotics, or in software which computes panoramic photography from scrap images and in many others.

I. Introduction

Image comparison and retrieval by their content is a complex process. Algorithms for content-based image retrieval have to be immune to change of scale, to image rotation and other transformations. Developers of image comparison methods try to imitate human perception process. The main problem is the difference between the perception of images by humans and computers. Human perception is focused on remembering the semantic description of the image with avoiding image details. Humans remember events, actions and objects represented by the images, but often they are not able to precise reproduce the images from memory. It is possible by human knowledge and skills learned from birth. Computer interpretation of images relies on a high number of details and computer algorithms are not able to semantically describe images because they lack imagination. On the other hand, thanks to their high accuracy, in many cases specialized algorithms such as facial and fingerprint recognition are better than humans as we need specialized knowledge and a long time to achieve the same goal [1][3][4][9][10][13].

In general-purpose applications, algorithms based on image keypoints are often used for comparing images [7][14]. Unfortunately, these methods generate large amount of data, which must then be compared between other sets of data from other images. This requires large computational burden during keypoint generation and their comparison process. Keypoints are areas of the image of specific details that are in contrast with the rest of the image. Thanks to this concept most of the picture can be automatically omitted and we can focus only on the essential details. Keypoints also create some relationships with each other by distance, orientation and size which then can determine translation between compared similar images, as well as rotation and displacement of objects included in the image. Some disadvantage is the fact that often some of the points between similar images do not overlap thus it cannot be assumed that the relationship between keypoints is invariant.

One of the most popular algorithms used to generate image keypoints is the SIFT algorithm (Scale-Invariant Feature Transform) [11]. The newer version of SIFT is the SURF (Speeded Up Robust Features) algorithm [2][8]. SURF is less accurate, but faster than SIFT. The method proposed in this paper is based on keypoints that were generated with the help of the SURF algorithm. The main features of each generated keypoint are: position, orientation, size, and descriptor. Algorithms for determining keypoints apply a mask defining the local extremes of the image. Such an action is typical for the algorithms of blob detection [15]. The image is analyzed many times, and in each subsequent step, the size of the mask is increased, creating the so-called, pyramid. This allows to identify keypoints regardless of their scale. The important advantage of the algorithm is a substantial acceleration with respect to the previous solutions, due to several improvements. The previous solutions used Gaussian mask which requires the addition of all the pixels repeatedly with a specified coefficient. The SURF algorithm has been improved by using the so-called integral image that allows a quick way to determine the sum of the pixels in a given area. This structure is represented by the sum of pixels in any rectangular area of the input image $I(1)$. This also has a negative impact on the accuracy of the designated descriptors, since adopted simplified masks that causes a

Patryk Najgebauer, Tomasz Nowak, Jakub Romanowski, Marcin Gabryel, Marcin Korytkowski and Rafał Scherer are with the Institute of Computational Intelligence, Czestochowa University of Technology, POLAND (email: {patryk.najgebauer, tomasz.nowak, jakub.romanowski, marcin.gabryel, marcin.korytkowski, rafal.scherer}@iisi.pcz.pl).

The work presented in this paper was supported by a grant from Switzerland through the Swiss Contribution to the enlarged European Union.

978-1-4799-1484-5/14/$31.00 ©2014 IEEE

minor deviation in rotation and position of keypoint.

$$I_{\sum}(x,y) = \sum_{i=0}^{i \le x} \sum_{j=0}^{j \le y} I(x,y) \tag{1}$$

where I is a processed image, $I_{\sum}(x,y)$ - the sum of all pixels in the image. Calculation of the sum of the pixels in the selected area of the image (integral image) is described by (2)

$$\sum = A - B - C + D \tag{2}$$

where A, B, C, D values are the sum of pixels in the selected point. Finally, the algorithm for each keypoint generates its descriptor and its orientation. Determined orientation of the keypoint allows to generate similar descriptors of points regardless of the global orientation of the entire image. Keypoints descriptor is array of values that represent the changes of local gradient around is keypoint.

$$V_{sub} = \left[\sum dx, \sum dy, \sum |dx|, \sum |dy|\right] \tag{3}$$

The primary descriptor generated by an SURF algorithm is made with 64 values. Each successive group of 4 values of descriptor creates a matrix of size 4 x 4 which extends over the keypoint (Fig. 1). Each group is the sum of Haar wavelets in x and y axis, this allows to efficiently describe the local gradient (3). Objects recognition based on keypoints

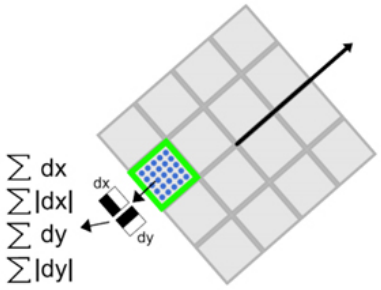

Fig. 1. Descriptor structure, 4x4 matrix of 4 value from sub region [2]

and their descriptors is done by matching all keypoints for two different images. Keypoints are usually applied to search for exactly the same object in another picture, e.g. while tracking an object on a video sequence. The algorithms which work on the key points allow to search similarity in another image even if it is rotated or partially hidden [6]. This paper proposes a new method for image keypoint descriptors indexing and comparison and is organized as follows. Section 2 presents a description of the problem and Section 3 the proposed method. Subsection 3.1 presents the process of dictionary creation from a descriptor set. Subsection 3.2 presents the base method of descriptor comparison (similar descriptors extraction from a dictionary) with the dictionary. Subsection 3.3 presents an additional method for dictionary comparison between two dictionaries (similar to set intersection). Section 4 presents experimental results of the method and the last section concludes the paper.

II. Problem Description

To properly compare similar descriptors we need to keep the appropriate deviation tolerance between them. For example, Table I presents the distribution of variation between similar values of descriptor component (Fig. 2) of the total standard deviation of 0.4743. For this example it can be assumed that standard deviation does not exceed 0.5 for similar descriptors. Increasing value of tolerance also results in possibility to create groups of similar descriptors. Keypoints and their orientations presented in (Fig. 2) human

TABLE I
Differences between V_{sub} of two similar keypoints descriptor

V_{sub}	**1**	**2**	**3**	**4**
1	0.0000	0.0059	0.0031	−0.0047
2	−0.0098	0.0144	0.0349	0.0159
3	−0.0495	−0.0214	−0.0159	0.0079
4	−0.0770	−0.0062	−0.0120	−0.0173

consider to be identical, but if we take into account the value of their descriptors, they are completely different. Each

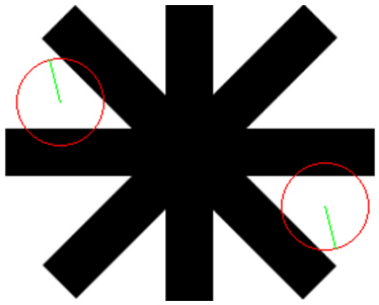

Fig. 2. Image example with similar SURF keypoints with 0.47 value of difference between descriptor components

of local feature detection methods (e.g. SURF or SIFT) generates large amount of keypoints per image [12]. Most disadvantages of such methods is the process of descriptor sets comparison. During our experiments we tried to speed up this process. The first and simplest method is generation of hashes for the keypoint descriptors. Hash code is a simple value that reduce large array of descriptor values to a short form. In case of hash codes we noticed several problems, mostly resulted from:

- Descriptors that values are located on the border between different hash codes.
- Randomly pick deviations of single values of descriptors.
- Deviations resulting from micro changes of keypoint position, orientation and scale.
- Distortion created from image noise.

After generating the keypoints of the compared images, they were divided into two sets. Their number depends on the size of the photos and the amount of details. Often for photos

of size equal to 1280 x 800 pixels, the amount of generated keypoints exceed 1000 points. The simplest and often used method of comparing the keypoints between images is to compare each other, but with such a large number of points that can require many long-running calculations. For example, two sets consisting of a thousand keypoints require one million comparisons. To reduce the number of required comparisons of descriptors we can sort them. Most of related works do not mention descriptors matching or only use brute force. Most of works are also more focused on keypoints spatial relationships. Certain part of the related works presents methods that reduce the number of comparisons but these approaches need initial learning process and not allow for latter extension of the structure. Most of them are based on k-means [16][17]. In our work we tray to avoid an initial learning process. The solution to the above problem will be discussed further in this work.

III. Method Description

To speed up the process of descriptor comparison, the presented method performs sorting and grouping of descriptor components. Each of the SURF descriptors contains 64 components which are floating-point values. Processed descriptor components create a structure similar to a dictionary [5]. Similarly to the vocabulary dictionary, in our method each descriptor is equivalent to a simple word. In the proposed method each word is of similar length and contains 64 elements of descriptors that is equivalent to a vocabulary single character. The proposed method by grouping of descriptors components allows to reduce the total number of descriptor comparison between images. It is achieved by a reduction of a total number of comparisons in each step of the comparison of descriptor components. Descriptors are omitted if their difference is greater than the assumed threshold. The descriptor dictionary is created in a similar way to the B-tree where each node has a reference to the next nodes that corresponds to descriptor components.

A. Dictionary Creation

The presented method processes all descriptors and creates a dictionary from their components. Descriptors can be inserted immediately after detection. Fig. 3 presents an example of the descriptor list that contains only the first five values and will be used to create a sample dictionary from Fig. 5. From each component of the descriptor, a dictionary character (node) is created. Character insertion into dictionary is started from the first character created from the first descriptor component. At first character are compared with each child character of dictionary root until value deviation is decreases. After that, if their deviation exceed the given threshold then the character with all of descendants characters is added into root child list in position where comparing was ended. In other case, if the deviation does not exceed the threshold, the method goes to the best matched child and starts comparing their child with the next character of inserting descriptors. Fig. 4 presents the process of dictionary descriptors (words) adding. As we can see, the descriptor number two (marked by red color) has similar a tree character with the next added descriptor number tree (marked by green color). In that case we group them by adding a sub-word of added descriptors (green) as a child of the last similar character element (red).

descriptor number	descriptor component number 1	2	3	4	5
1	-0.10	0.20	0.12	-0.06	0.22
2	0.05	0.18	0.22	-0.06	-0.07
3	0.06	0.15	0.25	0.18	-0.01
4	0.05	0.18	0.26	0.19	0.22
5	0.04	0.47	-0.10	-0.01	0.02
6	0.03	0.47	-0.06	0.22	0.18
7	0.27	0.18	-0.01	0.02	-0.01
8	0.24	0.52	0.22	0.18	0.22
9	0.46	0.18	-0.01	-0.06	-0.07
10	0.46	0.15	0.02	-0.01	0.22

Fig. 3. An example of list of first five descriptors components used to dictionary creation.

Threshold value used in the approach of dictionary creation is smaller (divided by 64) than the threshold used in main dictionary comparison process, we also do not compare entire descriptors (dictionary words) but only single components (dictionary character). This is caused by the fact that some sub-words which where grouped and assigned to other words even if their deviation was nearest to exceed. In that case these words were significantly different to their original and later were wrong matched.

An example of dictionary creation process is presented in Fig. 5. As we can see, to the first dictionary character of descriptor number 2 are directly assigned only descriptor number 5, but in entire dictionary this character represents descriptors numbered from 2 to 6. Thanks to component grouping we reduce the number of descriptor components that have a similar value.

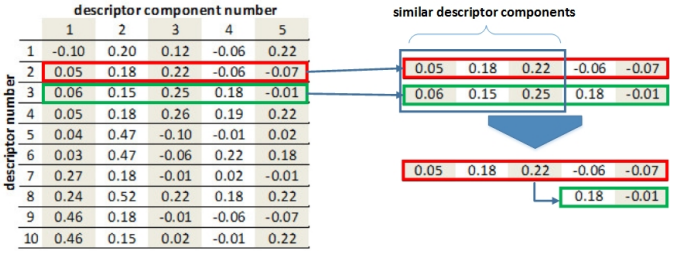

Fig. 4. An example of dictionary word comparison and grouping in the process of dictionary creation.

This approach also speeds up the process of descriptors index building, especially in the case of using a large number of descriptors. Other data of keypoints such as position, size, orientation is stored in the last character.

To better understand the structure of the descriptors dictionary we can present it as a tree. Descriptors dictionary represented as a tree has the same height dependent on the length of the descriptor. In our case, height is 64 that is equal to length of descriptor that are generated by the SURF

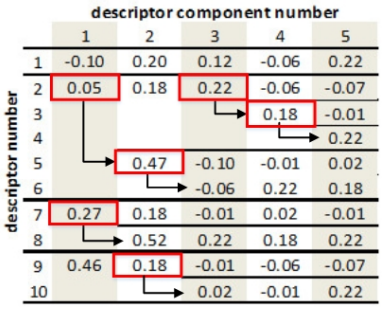

Fig. 5. An example of dictionary fragment consisting of ten descriptors

algorithm. Each path from the root to a leaf represents a single descriptor or their group, if all the values overlaps. The leaves store data of the related keypoints.

B. Descriptors Comparison With The Dictionary

Comparison of descriptors can be performed in two ways, first by comparing single descriptors with a dictionary and second by comparing two dictionaries.

The first method works in a similar way to finding the appropriate descriptor in the dictionary. This method works in a similar way to the method that adds a new descriptor. Components of the compared descriptor are compared with the dictionary beginning from the first component. Only components of the same level are compared. If similar components to the compared descriptor are found in a dictionary the method proceeds to the second stage and compares a sub list of dictionary components with the second component of descriptors. In each step, the value of total differences between dictionary path and the compared descriptor is increased. If this value exceeds the value of threshold, the path is omitted and goes to the next similar component of the dictionary. With each step the amount of total possible comparison combination is reduced. If the method reaches the last component, it returns the data of matched descriptors and tries to find another one. Fig. 6 shows an example of

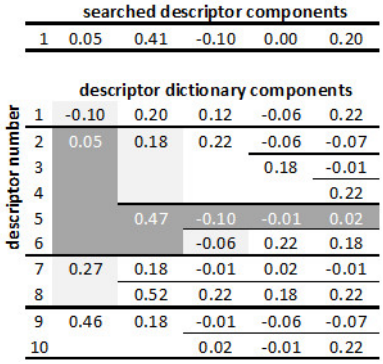

Fig. 6. Example of searching for a single descriptor in the dictionary

the descriptor comparison with the dictionary. The example presents only a fragment of the descriptor structure. The deviation in the example search between the descriptor, and the most similar path of dictionary is about 0.25. In the presented example we assumed that the threshold value is 0.5. If the deviation exceeds the threshold the searched path is skipped. We marked by light gray background color the components that were compared but they differences exceeds the thresholds. By dark grey background color we marked the components that matched the searched keypoint descriptor. We can also notice that comparison is omitted in five descriptors in the first dark gray components group as they have similar components.

C. Comparing Two Descriptor Dictionaries

In the case when dictionaries are created for each image or groups of images it is possible to compare whole dictionaries. That approach speeds up the whole operation even for large sets of keypoints. As it was mentioned earlier, dictionaries organize and group the descriptors, what accelerates their multiple utilization. Ordered structure of dictionaries allows for fast determination of the common parts between dictionaries instead of compare descriptor list with dictionary. Example of comparison of two dictionaries is presented

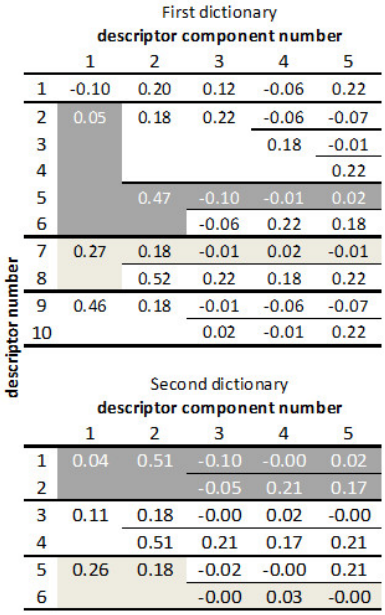

Fig. 7. Example of two dictionaries comparison. Similar components between dictionaries are marked by dark gray background color

in Fig. 7 where dark gray background denotes common parts of first group of similar descriptors and light gray the second group. In this approach we compare sorted data. The main advantage of the first method is that components are compared only once. We do not need to process them later as it is in the case of multi comparison between a dictionary and descriptors list.

IV. Experimental Results

Experiments have been carried out on test images presented in Fig. 8. During the experiments we adopt the value of the threshold deviation of descriptor equal to 0.5. To better demonstrate the effect of the algorithm, images have been

TABLE II

RESULTS OF DICTIONARY CREATION FROM FIG. 8

Image	Fig. 8a	Fig. 8b	Fig. 8c	Fig. 8d	Fig. 8e	Fig. 8f
No. of key points	147	158	170	65	1538	1265
No. of dictionary words	123	125	137	63	1496	1256
No. of component groups	20	28	22	13	77	71

specially prepared. They were chosen to be similar to varying degrees, thus we could obtain positive search results.

Table II presents experimental results of the algorithm to build the dictionary. The table shows the differences between the number of descriptors generated by the SURF algorithm, and the amount of the corresponding words in the dictionary of descriptors. We can conclude from first two rows of the table that only 11% of the generated keypoints descriptors are fully grouped in dictionary. This shows that about 11% of the descriptors coincides with the other, that translates to unnecessary processing of identical descriptors during images comparison.

Table II tells us also about grouping of first components of descriptors during dictionary creation. The last row contains the number of first elements in the dictionary, which group the first components of descriptors. As we see the first item of the first dictionary (Fig. 8a) groups about 6 descriptors and grows along with their quantity (for Fig. 8e there are 19 elements). This is related to the limited number of combinations of groups depending on the threshold of the maximum deviation between the descriptors.

Table III presents a summary of the results of the combination of comparisons for the objects shown in Fig. 8. The result of each comparison are three values. The first value is the number of matched (identical) descriptors between two images. The next value is an approximate amount of fully compared descriptors (number of compared components divided by descriptor length), because presented method does not compare the whole descriptors, but only their components.

This approach presents in better way the relationship between the number of comparisons made with the help of the dictionary, and the standard, all-to-all comparison. Number of comparisons without proposed improvements is presented as the third value. We can conclude from Table III that the number of comparison operations is substantially reduced in our method, moreover the number also decreases with increasing number of descriptors.

V. CONCLUSIONS

The paper concerned very important problem of modern computer science that is checking the similarity of two or even millions of images by their content. The most popular

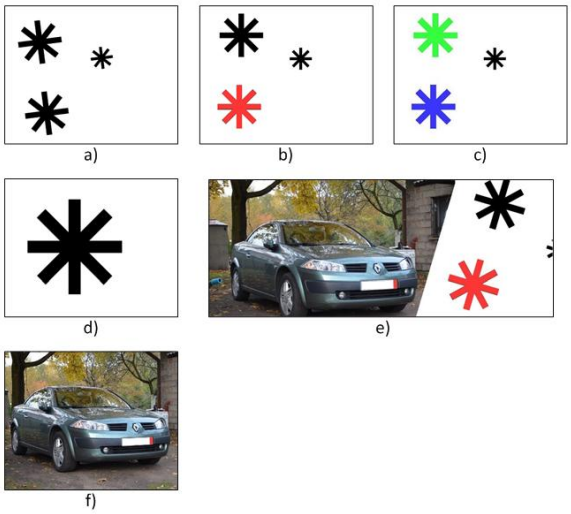

Fig. 8. Images used in experimental results

TABLE III

EXPERIMENTAL DICTIONARY COMPARISON RESULTS BETWEEN IMAGES (FIG. 8), VALUES: FINAL SIMILAR DESCRIPTORS, NUMBER OF COMPARED DESCRIPTOR BY DICTIONARY, NUMBER OF COMPARED DESCRIPTOR BY LIST (ALL TO ALL) METHOD

Image	Fig. 8a	Fig. 8b	Fig. 8c	Fig. 8d	Fig. 8e
Fig. 8b	55 323 19434				
	-	-	-	-	
Fig. 8c	27 327 20910	87 435 21250			
	-	-	-	-	
Fig. 8d	7 126 7995	7 162 8125	4 149 8905		
	-	-		-	
Fig. 8e	19 2373 18974	36 3004 192250	19 2558 210706	3 1519 96894	
	-				-
Fig. 8f	0 1913 155595	0 2401 158125	0 2038 173305	0 1210 79695	210 8404 1892440

method nowadays is generation of image keypoints and comparing them in brute force of all-to-all manner, what is extremely time-consuming. In the paper we presented a novel method of fast comparing images based on a dictionary structure. As we can see on the basis of experiments, create a dictionary file descriptors allows to significantly reduce the number of required operations during image comparison. For example, comparing pictures Fig. 8f and Fig. 8e showed 205 common descriptors. To designate the common descriptors it was enough to perform 8404 full descriptor comparisons, which represents only 0.4% of the operations that are needed in case of comparing not sorted descriptors. Our solution can be very useful for comparing a large group of pictures in order to determine their similarities. Dictionary does not require a large amount of memory, for sample image in

Fig. 8f, 1.256 descriptors were generated, which took about 4 KB of memory.

The presented method can be applied as a standalone content-based image retrieval system that can speed up grouping images by similarity or searching image with a similar content. The proposed method can be also used as a temporary database that process e.g. video material to search for similar scenes.

REFERENCES

[1] C.B. Akgül, D.L. Rubin, S. Napel, C.F. Beaulieu, H. Greenspan and B. Acar, "Content-based image retrieval in radiology: current status and future directions," *J.Digit. Imaging*, vol. 2, pp. 208–222, 2011.

[2] H. Bay, A. Ess, T. Tuytelaars and L.V. Gool, "Speeded-Up Robust Features (SURF)," *Comput. Vis. Image Underst*, vol. 110, pp. 346–359, Jun. 2008.

[3] M. Bazarganigilani, "Optimized Image Feature Selection Using Pairwise Classifiers," *Journal of Artificial Intelligence and Soft Computing Research*, vol. 1, no. 2, pp. 147–154, 2011.

[4] Y. Chang, Y. Wang, C. Chen and K. Ricanek, "Improved Image-Based Automatic Gender Classification by Feature Selection," *Journal of Artificial Intelligence and Soft Computing Research*, vol. 1, no. 3, pp. 241–253, 2011.

[5] S. Edelkamp and S. Schrödl, "Dictionary Data Structures, Heuristic Search," *Elsevier*, pp. 89–159, 2012.

[6] M.A. Fischler and R.C. Bolles, "Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography," *SRI International. Artificial Intelligence Center*, pp. 381–395, 1981.

[7] J.J. Foo and R. Sinha, "Pruning SIFT for Scalable Near-duplicate Image Matching," *Australian Computer Society*, vol. 63, pp. 63–71, 2007.

[8] D. Gossow, P. Decker and D. Paulus, "An evaluation of open source SURF implementations," *Lecture Notes in Computer Science*, vol. 6556, pp 169–179, 2011.

[9] P. Górecki, P. Artiemjew, P. Drozda, K. Sopyla, "Categorization of Similar Objects Using Bag of Visual Words and Support Vector Machines," *Proceedings of 4th International Conference on Agents and Artificial Intelligence*, pp. 231–236, 2012.

[10] D.R. Kisku, A. Rattani, E. Grosso and M. Tistarelli, "Face Identification by SIFT-based Complete Graph Topology," *IEEE Workshop*, pp. 63–68, Jun. 2010.

[11] D. Lowe, "Distinctive Image Features from Scale-Invariant Keypoints," *Int. J. Comput. Vision*, vol. 60, no. 2, pp. 91–110, 2004.

[12] M.G. Pena, "A Comparative Study of Three Image Matching Algorithms: Sift, Surf, and Fast," *BiblioBazaar*, 2011.

[13] F.B. Tek, A.G. Dempster and I. Kale, "Malaria Parasite Detection in Peripheral Blood Images," *British Machine Vision Conference*, 2006.

[14] P. Turcot and D. Lowe, "Better matching with fewer features: The selection of useful features in large database recognition problems, *Computer Vision Workshops (ICCV Workshops)*, pp. 2109-2116, 2009.

[15] L. Wang and H. Ju, "A Robust Blob Detection and Delineation Method," *Proceedings of the 2008 International Workshop on Education Technology and Training & 2008 International Workshop on Geoscience and Remote Sensing*, vol. 1, pp. 827–830, Dec. 2008.

[16] D. Schmitt and N. McCoy, "Object Classification and Localization Using SURF Descriptors", Citeseer, 2011.

[17] M. Muja, D. G. Lowe, "Fast Approximate Nearest Neighbors with Automatic Algorithm Configuration.", In VISAPP (1), pp. 331-340, Feb. 2009.