Real Time Implementation of HIPs Descriptor on Android Mobile Platform in an Augmented Reality Application

Marek Jakab*
Faculty of Informatics and Information Technologies
Slovak University of Technology in Bratislava
Ilkovičova 3, 842 16 Bratislava, Slovakia
marko.jakab@gmail.com

Abstract

The purpose of our research is to develop an application of augmented reality on mobile device, which is educative and entertaining for its users – children. A user is asked to capture images using a camera built in the mobile device and the application will provide a supplementary information into the image in the real time. The augmented information could be for example a text or graphics in the same image. Typically, the images are taken from children books. The key task of our application is the problem of image recognition on mobile platform using local descriptors. Our goal is to develop an application which is running in real time. Currently available descriptors included in OpenCV library are well designed, some of them are scale and rotation invariant, but most of them are time and memory consuming and hence not suitable for mobile platform. Therefore we decided to develop a fast binary descriptor based on the Histogram of Intensity PatcheS (HIPs) originally proposed by Simon Taylor et al. To train the descriptor, we need a set of images derived from a reference picture taken under varying viewing conditions and varying geometry. Our descriptor is based on a histogram of intensity of the selected pixels around the key-point. We use this descriptor in the combination with the FAST key-point detector and a sophisticated method of key-points selection is then used with the aim to reduce the computation time.

Categories and Subject Descriptors

I.4.7 [**Image Processing and Computer Vision**]: Feature Measurement—*Feature representation*; I.5.4 [**Pattern recognition**]: Applications—*Computer vision*

Keywords

Augmented Reality, Mobile Device, Descriptor, HIPs, Histogram, Image Recognition, Keypoint

1. Introduction

Problem of the visual object recognition using feature matching is one of the most challenging tasks in the computer vision. Nowadays, there are several ways for successfully match a template with an image. Most of these methods are quite robust, however they are still complex and most of them are not capable of performing matching in real time on large databases. In our paper, we describe a method for image matching, which could be promising for the real time applications even on mobile devices.

The authors Taylor et al. have presented [11] a simple patch feature with a binary mask representation, which enables very fast matching at runtime - Histogram of Intensity Patches. Our approach presented in this paper is based on this HIPs descriptor. In our algorithm we use methods of detecting local features and building a set of HIPs descriptors. This algorithm is compatible with other algorithms already included in OpenCV library. The basic idea how to decrease the computation time is to build the descriptor in a way that the training process includes many viewpoints corresponding to varying rotation, scale and affine transformation. Hence, rotation, scale and affine invariance could be achieved in the training phase, the matching runtime process directly use the descriptor and no additional computation time is necessary. This is the fact, which makes some other methods slower and not capable for running in real time.

In the training process, we build a binary descriptor of a patch around the detected feature key-point for all viewpoints. All descriptors are stored for a later use, so we do not need to go through the process of the training again. For the simulation of different image views, we use transformation provided by OpenCV library. However the training process takes several minutes to complete and use extra computing memory. We use the same approach with small differences on the acquired camera image, and then match the features by counting of the dissimilarity score, which is the result of bitwise operations between descriptors. We accept descriptor pairs as

*Bachelor degree study programme in field Informatics. Supervisor: Ing. Vanda Benešová, PhD., Institute of Applied Informatics, Faculty of Informatics and Information Technologies, STU in Bratislava.
Work described in this paper was presented at the 8th Student Research Conference in Informatics and Information Technologies IIT.SRC 2013. Source code available at http://vgg.fiit.stuba.sk

Jakab, M. Real Time Implementation of HIPs Descriptor on Android Mobile Platform in an Augmented Reality Application. Information Sciences and Technologies Bulletin of the ACM Slovakia, Special Section on Student Research in Informatics and Information Technologies, Vol. 5, No. 2 (2013) 27-31

a good matched pair only if their dissimilarity score is less than a given threshold. Additionally, we accept only those pairs of descriptors which have had the best similarity score in the previous selection using different view. Only these selected pairs are then used as an input in the next step - finding a homography matrix using the RANSAC (Random Sample Consensus) algorithm.

2. Related work

There are several descriptors providing well matching probability. The most common are SIFT [6] [5] [12] (Scale-invariant feature transform), SURF [1] [5] (Speeded up robust features), BRIEF[2] (Binary robust independent elementary features) or ORB [9] (Oriented BRIEF). In this part we describe how SIFT and SURF work, as we use them in comparison to HIPs in our tests.

SIFT descriptor use for key-point detection Difference of Gaussian (DoG) [7]. DoG is used on two neighbour images from image pyramid. These two images are blurred and then subtracted. Key-points are detected as we search for local maxima/minima in the DoG pyramid. DoG is a faster approximation of Laplacian of Gaussian (LoG) [3] and also provide the scale invariance for SIFT descriptor. Despite the fact, computation time of DoG algorithm is still high to use it in real time tracking. SIFT descriptor divides surrounding pixels into areas of size 4x4, and computes histogram of oriented gradients with 8 bins. Therefore the resulting vector is 4*4*8 = 128 dimensional.

SURF descriptor also divide the area around detected key-point into 4x4 smaller areas. For each area it computes Haar wavelets in X and Y direction and their absolute values. Next, these values are normalised and stored in 64 dimensional vectors. To provide scale invariance, SURF detector instead of making image pyramid scales the filter.

PhonySIFT [13] [12] is modified SIFT for tracking on mobile devices. They replaced DoG method for detecting key-points with FAST [8] detector. Scale invariance, which was provided by DoG, was replaced by storing descriptor from different scales. Area around the key-point to fill descriptor was changed from 4x4 to 3x3. As in original SIFT, they compute histogram of oriented gradients with 4 bins, so the result vector is 36 dimensional instead of 128. Authors observe only 10% worse matching in comparison to original SIFT.

The results of our work on HIPs descriptor were also presented at 17th Central European Seminar on Computer Graphics (CESCG) 2013.[4]

SIFT or SURF use floating point numbers for describing the area around the detected key-points. To optimize computational times, better approach is to use binary descriptors as HIPs, which we describe below.

3. Training process

To build a suitable descriptor to match selected image we need to pass the process of training. This process consists of detecting and describing the area around the feature key-point. To provide matching rotation and scale invariance, we build descriptors on more viewpoint bins of an image, which we want to detect by the algorithm. These viewpoint bins are simply created by warping of the reference image. For each bin, small rotations and transformations are performed with the aim of increased robustness. Created images need next to pass through key-point detector, then the binary descriptor of each key-point will be calculated.

3.1 Feature detecting

For each image viewpoint in a bin, local features key-points using FAST corner detector are detected. In the next step, the appearance of each feature in all images of the bin will be sorted and top detected 50 to 100 features are selected. The used parameters of warping have to be stored since they are necessary to find out a position of the feature in the reference image.

3.2 Patch extracting and building the descriptor

After we have detected the top 50 to 100 feature key-points in the current viewpoint bin, the descriptor could be calculated. We form a sample grid of 8 x 8 pixels around each of most detected corners key-point on each image in viewpoint. Pixels in the position given by the sample grid will take a part in process of filling the descriptor. 8 x 8 pixels, i.e. 64 pixels will form the descriptor, which will be enough to determine good or bad matches using dissimilarity score.

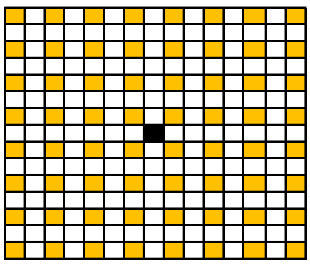

Figure 1. Sample grid around detected key-point. 8x8 highlighted pixels are used to create descriptor.

To provide the matching more robust to light variations, the selected values are normalised and then quantised into 5 levels of intensities. Intensities of pixels in this grid are used to form the histogram. Histogram is created in a way, it represents frequency of intensity appearance at selected position in all grids around corresponding key-point detected on training images. The feature descriptor building process is as follows: we fill "1" at selected position of the selected intensity level, if the intensity appearance in the histogram for this position is less than 5%. If selected intensity appears more frequently than 5%, we put "0" at the corresponding position. This boundary of 5% is determined by authors of HIPs to be best for filling the descriptor.

The finished feature descriptor will need to take 5 bits for each one of 64 pixels. Therefore we need 40 bytes to store the descriptor values and another 4 bytes memory to store the feature position. For example, if we decide to form the descriptor out of 100 features in a single bin, we need 4400 bytes of memory. To detect an image in various position, we need to calculate several hundreds of bins with different transformations.

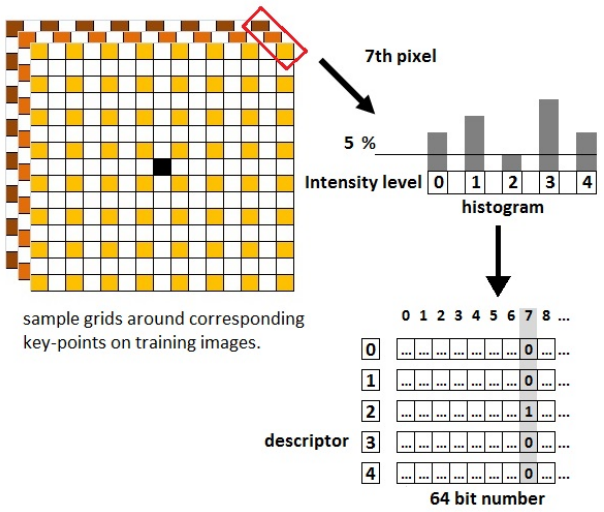

Figure 2. Process of forming the descriptor.

4. Matching

After we have created the descriptors for the detected features, we do the same procedure on a captured image. Because we built the binary descriptor in a way that we filled it with 1 if pixel rarely felt into the bin, we can match descriptors by a simple way of using bitwise operations and bit count. We simply AND each of descriptors level and then OR the results. This operation require 5 AND and 4 OR operations. Then we need to count number of set bits in our result, which provides us information about dissimilarity of descriptors.

$$s = ((R_0 \& C_0) \| (R_1 \& C_1) \| (R_2 \& C_2) \| (R_3 \& C_3) \| (R_4 \& C_4))$$

Where number R_i means i-th intensity level of the descriptor made in the surroundings of a feature from the reference template image and C_i i-th intensity level of the descriptor from the camera image.

$$dissimilarity\_score = countOfOnesInBitfield(s)$$

To declare descriptors as a good match they need to have this dissimilarity score less than some threshold, typically 5. After we select all good matches, we use them to find homography using RANSAC found in OpenCV library. Next we draw matched features and show successful match on screen.

5. Tests and results

Our testing algorithm is made in C/C++ programming language using OpenCV library and runs on laptop with i7 3632 QM 2,20 GHz processor and 8GB of DDR5 1600MHz memory. We created 1 viewpoint bin consisted of 3 different scales and 2 clockwise and anticlockwise rotations from reference image. Each of generated image were also 5 times perspective transformed from each sides by small amounts. In sum, we got 315 transformed images to form the descriptor. This option is no optimized yet and will be investigated in our future research.

First, we took an image reference with the resolution 251 x 356 pixels and try to match it with the image from camera with resolution 640 x 480 pixels. Next chart shows the average time of computation of the matched images using our implementation of HIPs for 1 viewpoint bin and the average computation time for SIFT and SURF algorithm implemented in OpenCV library. HIPS descriptor was created from top 100 key-points detected on the reference image in a single bin, containing 315 images of small rotations, scales and perspective transformations.

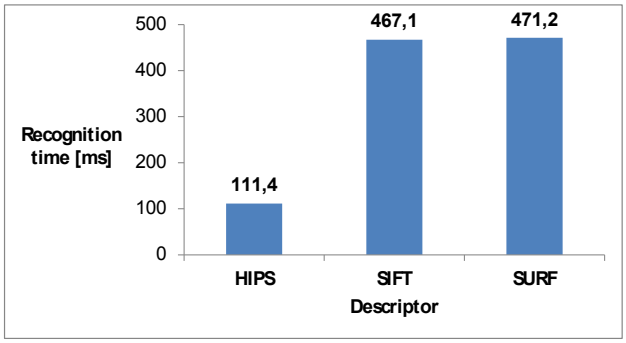

Figure 3. Elapsed time of computation for matching the descriptors.

Related to the Figure 3, we can see faster performance of HIPs descriptor in comparison to SIFT and SURF. However for possible matching for more bins and therefore increase scale and rotation variance, we have to match descriptors trough more viewpoints, therefore time of computation will rise. The presented algorithm is promising, but still needs further optimisation in case of mobile platform. However, current implementation gives us the possibility to select viewpoints, we consider to be more promising to detect.

Next, we considered to improve the time of computation by reducing the number of detected features used for the descriptor forming and matching. Taking less features and therefore creating less descriptors could improve computation time, but also can reduce the probability of a successful match. We have acquired 20 random images, which will takes part in our next test. Then we have set the number of features, which will be formed into descriptors, to 10, 25, 50 and 100. Next we have try to match a reference image with the 20 images taken before. We have evaluated the number of successful matches and also we have measured the time of computation needed for each image. In the next chart (Figure 4.) you can see the results in %.

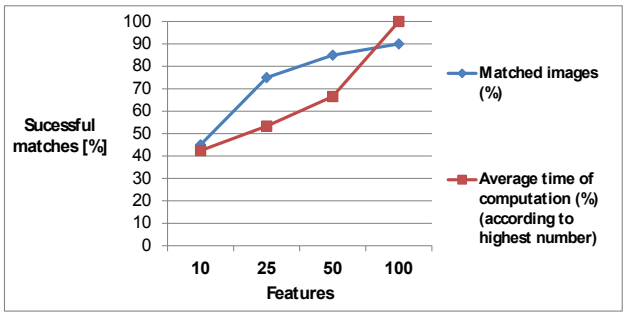

Figure 4. Average computation time required for successful matches.

We can see that the time of computation needed for the matching of one image increases significantly by increas-

ing the number of selected features to form the descriptor. Otherwise, if only 25 features are selected, we can see only a small difference in successful matching ratio comparing with 100 selected features. Therefore we can decide that for our purposes with current rotations, scales and perspective transformations, there is a good trade-off to form the descriptor by using 25 to 50 selected features.

Next chart (Figure 5.) show average time in seconds needed for the matching in our test. The difference significantly grows with more features selected. We can choose to make descriptors from less features, but this test contains only one viewpoint bin and therefore the computational time seems to be still high. To decrease the matching time, there is an opportunity of forming created descriptors into a binary tree or making indexes in which we can search faster than in our current tests, where the search through the descriptors is linear.

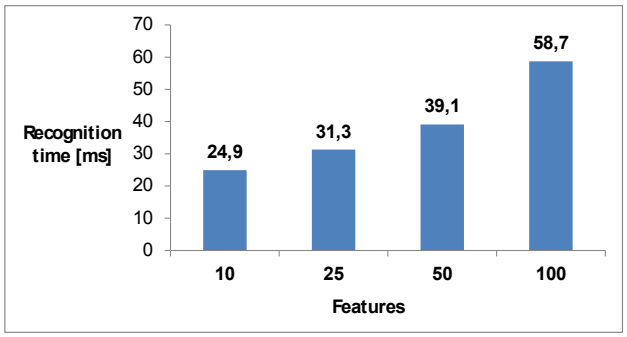

Figure 5. Computation time for our test in seconds.

5.1 Test on mobile devices

We have successfully implemented the mentioned HIPs descriptor for the Android platform application. Crucial point in the development and in the implementation of the descriptor was a real time capability on mobile devices. We have tested and evaluated the time profiling of the implementation on different mobile devices with different hardware specifications corresponding to different categories. The list of devices used in next test and their hardware specifications are provided in next table (Figure 6.).

Mobile Device	CPU	RAM
HTC Desire Bravo	1 GHz ARMv7 "Snapdragon"	ROM 512MB RAM 576MB
HTC Wildfire S	600 MHz ARM1136	ROM 512 MB RAM 512 MB
Samsung Galaxy SIII	Exynos 4 Quad 1,4 GHz (4x Cortex A9)	RAM 1 GB

Figure 6. List of mobile devices with their specification.

HIPs descriptor was build of 40 most frequently detected keypoints on an image and matched with descriptors created on an camera image with resolution 640 to 480 pixels. In our test, we observe the recognition speed of our algorithm and the presented values are averaged time values calculated from 15 successful matches. The received results are shown in next figure (Figure 7.).

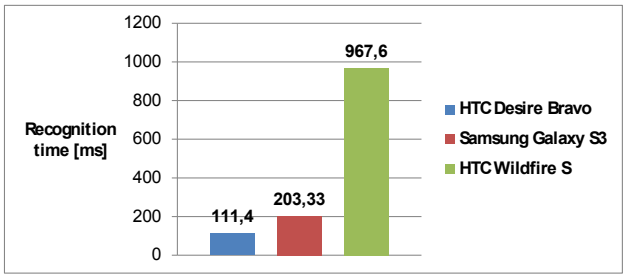

Figure 7. Matching time of HIPs on mobile devices.

The presented matching time of HIPs descriptor show us that the real time matching using our algorithm and using current devices on the market is possible. However, our test is not fully scale and rotation invariant, therefore there is still option for improvement. The result speed depend on the design of the CPU, which could be the reason HTC Desire Bravo gives us better performance result in matching.

6. Conclusion and future work

As presented, the results achieved by using the HIPs descriptor seem to be promising, however there are still possible improvements in the algorithm and also in the implementation. These improvements should make the algorithm faster and suitable for real time matching in larger databases. Data are stored in memory, which refer to all different image warps and therefore the memory requirement is higher. In our implementation the memory requirement is still acceptable and manageable by mobile devices (around 20 to 40 megabytes for training phase), but also here is a need of an optimization. The pros of this method is that we do not need to save any of image transformation during the evaluating process and it can be done just once in the training phase. The next possible improving, which could be done in our future work is an optimization of the algorithm by trying various transforming conditions on each bin. Our algorithm has run-time complexity of $O(n*m)$ for matching now, where n is the number of descriptors detected on reference image and m number of descriptors created from image from camera. Our goal is to make the presented algorithm faster in the run-time and then integrate this method as a part of our augmented reality application on a mobile device.

Acknowledgements. KEGA 068UK-4/2011 UAPI

References

[1] H. Bay, T. Tuytelaars, and L. V. Gool. Surf: speeded up robust features. In *Proceedings of the 9th European Conference on Computer Vision*, pages 404–417, 2006.

[2] M. Calonder, V. Lepetit, C. Strecha, and P. Fua. Brief: Binary robust independent elementary features. In *Proceedings of the 11th European Conference on Computer Vision*, pages 778–792, September 2010.

[3] M. Dobeš. *Image processing and algorithms in C#*. BEN, Praha, 2008.

[4] M. Jakab. Planar object recognition using local descriptor based on histogram of intensity patches. In *Proceedings of the 17th Central European Seminar on Computer Graphics*, pages 139–143, 2013.

[5] M. Kottman. Planar object detection using local feature descriptors. *Association of Computing Machinery bulletin*, 3(2):59–63, June 2011.

[6] D. G. Lowe. Object recognition from local scale-invariant features. In *Proceedings of the Seventh IEEE International Conference on Computer Vision*, volume 2, pages 1150–1157, 1999.

[7] K. Mikolajczyk and C. Schmid. Scale & affine invariant interest point detectors. *International Journal of Computer Vision*, 60(1):63–86, 2004.

[8] E. Rosten and T. Drummond. Machine learning for high-speed corner detection. In *Proceedings of 9th European Conference on Computer Vision, Graz, Austria*, pages 430–443, May 2006.

[9] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski. Orb: an efficient alternative to sift or surf. In *IEEE International Conference on Computer Vision (ICCV)*, pages 2564–2571, 2011.

[10] O. Sidla, M. Kottmann, and W. Benesova. Real-time pose invariant logo and pattern detection. In *Proceedings of SPIE*, January 2011.

[11] S. Taylor, E. Rosten, and T. Drummond. Robust feature matching in 2.3 us. In *Proceedings of Computer Vision and Pattern Recognition conference*, pages 15–22, June 2009.

[12] D. Wagner, G. Reitmayr, A. Mulloni, T. Drummond, and D. Schmalstieg. Pose tracking from natural features on mobile phones. In *ISMAR '08 Proceedings of the 7th IEEE/ACM International Symposium on Mixed and Augmented Reality*, pages 125–134, 2008.

[13] D. Wagner, G. Reitmayr, A. Mulloni, T. Drummond, and D. Schmalstieg. Real-time detection and tracking for augmented reality on mobile phones. *Visualization and Computer Graphics*, 16(3):355–368, May-June 2010.