Planar Object Recognition Using Local Descriptor Based On Histogram Of Intensity Patches.

Marek Jakab
Supervised by: Ing. Vanda Benešová, PhD.

Faculty of Informatics and Information Technologies.
Slovak University of Technology
Bratislava / Slovakia

Abstract

The purpose of our research is to develop an application of augmented reality on mobile device that will be educative and entertaining for their users - children. User will be asked for an input to take a picture from the book and the application will draw a supplementary information in the form of a 3D object on the screen. The key task of our application is the problem of image recognition on mobile platform using local descriptors. 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 geometry and therefore we have to take into account different scales, rotations and perform perspective transformations. 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, where the most occurring key-points are used with the aim to reduce the computation time.

Keywords: augmented reality, mobile, 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. Lot 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 [1] 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, because it uses the basic data structures defined in this 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 memory. We use the same approach 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. The results with score less than threshold = 5 will be selected as good matches and used to find a homography using the RANSAC (Random sample consensus) algorithm. Selected threshold with value of 5 gives good results for matching because the image could be still recognized when random noise or other disturbances occur in the camera image. For the lower threshold, the probability of successful matches will decrease.

2 Related work

There are several descriptors providing well matching probability. The most common are SIFT [11] [5] [6] (Scale-invariant feature transform), SURF [10] [5] (Speeded up robust features), BRIEF [8] (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) [3]. 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) [2] and also provide the scale invariance for SIFT descriptor. Despite the fact, computation time of DoG 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 [7] [6] is modified SIFT for tracking on mobile devices. Authors replaced DoG method for detecting key-points with FAST [4] 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. They observe only 10% worse matching in comparison to original SIFT according to experiments carried out and presented by authors.

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

3 Image training and matching

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, which after de-warping the image back to reference position occurs most frequently, 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.

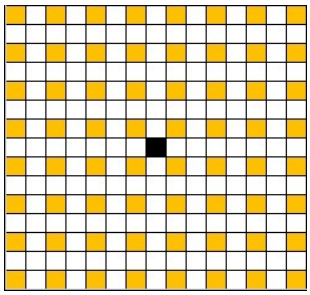

Figure 1. Sample grid around detected key-point. 8x8 highlighted pixels are used to create descriptor.

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.

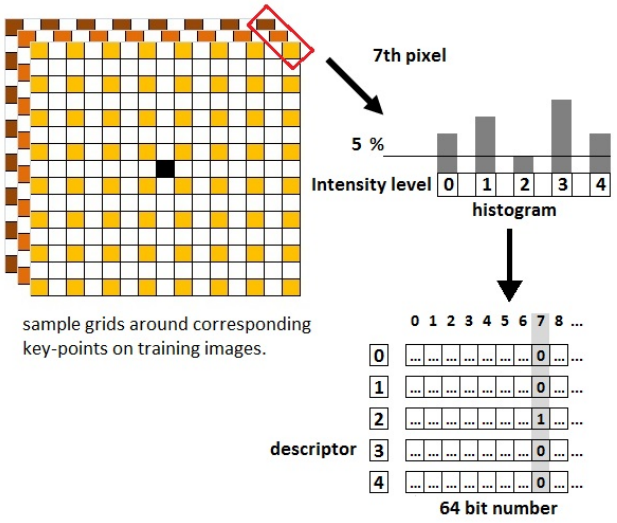

Figure 2. Process of forming the descriptor.

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.

3.3 Matching

To match two descriptors we need to count the dissimilarity score between them. 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 sum of set bits. We simply AND each of descriptors level containing 64 bit number and then OR all 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.

Supposing, we have created a descriptor R for a feature on an template image and we are looking for a descriptor on camera image C. Descriptors contain 5 levels, each level is 64 bit long. To sum the bits set to 1 and to get information of a good match, we need to make AND operation among each of the descriptor levels (first level of R descriptor AND first level of C descriptor and so with others). We get 5 numbers of 64 bit length which are then ORed to get one 64 bit number.

$$s = ((R_0 \& C_0) \| (R_1 \& C_1) \| (R_2 \& C_2) \| (R_3 \& C_3) \| (R_4 \& C_4)) \quad (1)$$

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.

$$dissimilarity\_score = sumOfOnesInBitfield(s) \quad (2)$$

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.

4 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 with step of 0,03 and 2 clockwise and anticlockwise rotations from reference image with step of 2,5°. Each of generated image were also 5 times perspective transformed from each of 4 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.

For testing the algorithm we took an image reference with the resolution 126 x 178 pixels and try to match it with the image from camera with resolution 320 x 240 pixels. Next graph shows the average computation time of the matched images using our implementation of HIPs for 1 viewpoint bin and the average time of computation for SIFT and SURF algorithm implemented in OpenCV library.

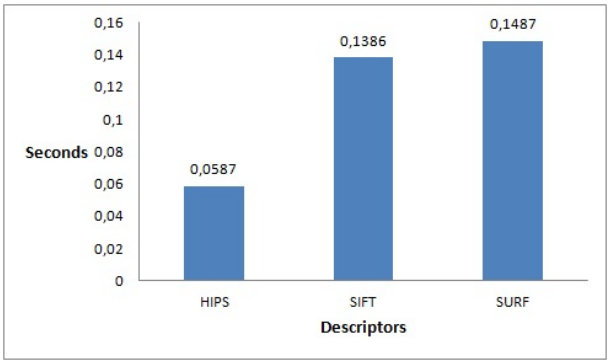

Figure 3. Elapsed computation time for matching the descriptors.

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.

Related to the Figure 3, we can see that HIPs is running more than twice faster than SIFT. However for possible matching for different bins we have to pass through the process for each viewpoint, therefore computation time

will rise. The presented algorithm is promising, but still needs further optimization in case of mobile platform.

We have done the same testing on an image with the resolution 251 x 356 pixels and with the resolution of 640 x 480 pixels. We made same approach of creating viewpoint bin and feature selection as in previous test with smaller image. In this test, we want to observe how good each of descriptors works with higher resolution images.

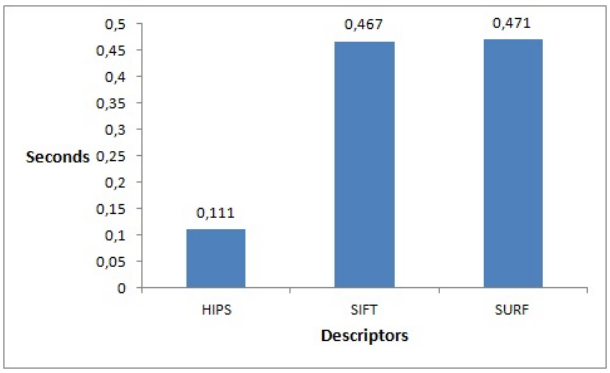

Figure 4. Elapsed time of computation for matching the descriptors with bigger resolution of an image and camera.

The result of our next test shows even bigger difference. Our implementation of HIPs took two times more computation time than the implementation in previous test. The difference between HIPs and SIFT / SURF is greater than in previous test, so it seems HIPs works better for larger images. However the computational time is still not suitable for real time applications.

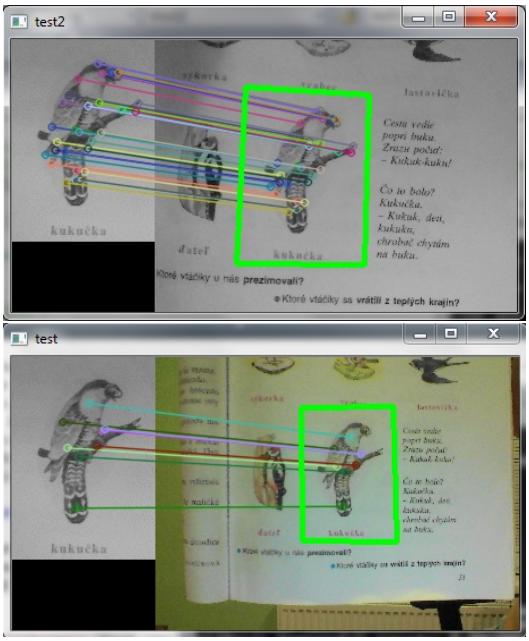

Figure 5. Above: Image matched by HIPs. Bottom: Image matched by SIFT

Next, we consider to improve the computation time 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 computation time needed for each image. In the next graph (Figure 6.) you can see the results in %.

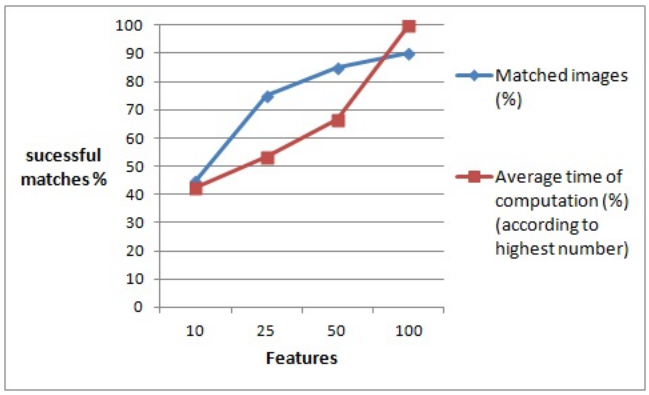

Figure 6. Average computation time required for successful matches.

We can see, that the duration of the matching part of the algorithm increases significantly by increasing 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. Calculation using 25 to 50 selected features is significantly faster and give still acceptable matching results.

Next graph (in the Figure 7.) 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 time of computation 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.

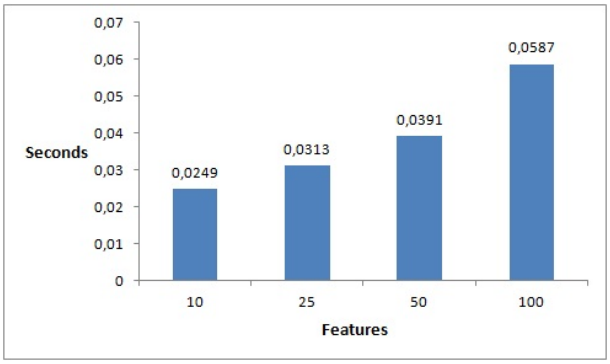

Figure 7. Computation time for our test in seconds.

5 Conclusion and future work

We implemented the HIPs algorithm and evaluated its speed in comparison to other well known descriptors. As presented, the results achieved by using the HIPs descriptor seems 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.

6 Acknowledgement

This work was supported by grant KEGA 068UK-4/2011 UAPI.

References

[1] Simon Taylor, Edward Rosten and Tom Drummond. Robust feature matching in 2.3 us. *In Proceedings of Computer Vision and Pattern Recognition(CVPR) conference*, 2009, s. 15-22.

[2] Michal Dobeš. Image processing and algorithms in C#. *1. edition. Praha. 2008.* ISBN 978-80-7300-233-6.

[3] Krystian Mikolajczyk and Cordelia Schmid. Scale & Affine Invariant Interest Point Detectors. *In International Journal of Computer Vision,* vol. 60, 2004, no. 1, pp. 63-86.

[4] Edward Rosten and Tom Drummond. Machine learning for high-speed corner detection. *In Proceedings of 9th European Conference on Computer Vision, Graz, Austria, May 7-13,* 2006, pp. 430-443

[5] Michal Kottman. Planar Object Detection using Local Feature Descriptors. *In: Association of Computing Machinery bulletin*, June 2011, vol. 3, no. 2, pp. 59-63.

[6] Daniel Wagner, Gerhard Reitmayr, Alessandro Mulloni, Tom Drummond and Dieter Schmalstieg. Pose tracking from natural features on mobile phones. *ISMAR '08 Proceedings of the 7th IEEE/ACM International Symposium on Mixed and Augmented Reality*, 2008, pp. 125-134.

[7] Daniel Wagner, Gerhard Reitmayr, Alessandro Mulloni, Tom Drummond and Dieter Schmalstieg. Real-Time detection and tracking for augmented reality on mobile phones. *In Visualization and Computer Graphics, vol. 16*, 2010, no. 3, pp. 355-368.

[8] Michael Calonder, Vincent Lepetit, Christoph Strecha and Pascal Fua. BRIEF: Binary Robust Independent Elementary Features. *In Proceedings of European Conference on Computer Vision*, Sep. 2010.

[9] Ethan Rublee, Vincent Rabaud, Kurt Konolige and Gary Bradski. ORB: an efficient alternative to SIFT or SURF. *In Proceedings of Computer Vision (ICCV), 2011 IEEE International Conference , Barcelona*, pp. 2564-2571.

[10] Herbert Bay, Tinne Tuytelaars and Luc Van Gool. SURF: speeded up robust features. *In Proceedings of 9th European Conference on Computer Vision, Graz, Austria*, May 7-13, 2006, pp. 404-417.

[11] Lowe, David G. Object recognition from local scale-invariant features. *In Proceedings of the Seventh IEEE International Conference on Computer Vision*, 1999, vol. 2, pp. 1150-1157.