A comparative study on mobile visual recognition

Elisavet Chatzilari[1,2], Georgios Liaros[1,3], Spiros Nikolopoulos[1], and Yiannis Kompatsiaris[1]

[1] Information Technologies Institute, Centre for Research and Technology Hellas, Thessaloniki, Greece,
[2] Centre for Vision, Speech and Signal Processing University of Surrey Guildford, UK
[3] Dept. of Informatics, Ionian University, 49100, Kerkyra, Greece
{ehatzi,geoliaros,nikolopo,ikom}@iti.gr

Abstract. In this work we perform an extensive comparative study of approaches for mobile visual recognition by simultaneously evaluating the performance and the computational cost of state-of-the-art key-point detection, feature extraction and encoding algorithms. Every step is independently tested so that its contribution to the final computational cost can be measured. The widely used OpenCV library is utilized for the implementation of the algorithms, while the evaluation is performed on the PASCAL VOC 2007 dataset, a challenging real world dataset crawled from the web. Our study identifies the algorithmic configurations that manage to optimally balance performance and computational cost, and provide a viable solution for real time mobile visual recognition.

Keywords: image classification, feature extraction, mobile visual recognition, OpenCV

1 Introduction

In the 90s, the second generation (2G) cellular technology limited the functionalities of mobile phones to the very basics, i.e. making calls and sending text messages (SMS). Rapidly, mobile networks and devices began to evolve to higher speed networks (GPRS and WAP) and smaller devices with new functionalities (MMS and emails). In the last decade, the third generation (3G) was launched which in turn gave its place to 4G in 2009. In parallel with the developments in network capabilities, mobile devices evolved to smartphones, typically equipped with more processing power, rich sensors like GPS receivers and quality cameras. This fact opened up a whole new range of possibilities and radically new services like mobile visual recognition, a field of intense research for the last years. Mobile visual recognition refers to the process of taking a photo with the mobile phone camera and, after processing its visual content, presenting relevant information back to the user. A typical example is when a photo of a landmark is processed to return textual information (e.g. a wikipedia article) or related images (e.g. different views of the same landmark).

The basic motivation for using mobile image recognition is the ability of visual content to transfer rich semantic content that is either too complicated or too ambiguous to be expressed with words. Indeed, if the user is not sure how to describe something with words it may be easier to search with a picture. Moreover, it is also possible to use image recognition services with embedded optical character recognition (OCR) capabilities for translating foreign billboards and road signs. Finally, the ability of mobile image recognition to turn the world around us into semantic links through a mobile phone camera, e.g. in the form of augmented reality metatags pointing to news, websites, or special offers (e.g. Layar, Wikitude) is what makes this service way more attractive than text or voice-based search that are more demanding from the perspective of user input.

Although research in visual recognition has made great progress during the last decade achieving relatively high performance figures [11],[24], the situation is different when aiming for practical applications where real-time processing is a critical factor. According to [4], 40% of the users will abandon a mobile application if they have to wait more than three seconds, and if that time rises to 10 seconds, 60% will not use it for a second time. For a visual recognition application, this formulates the typical trade off between the accuracy of the utilized algorithms and their computational cost, which is even more critical in the mobile environment where the resources are limited. In this context, the majority of existing mobile visual recognition applications employ a client/server architecture (Fig. 1). Cell phones act as clients that capture the object of interest and send queries to the server where the actual processing part takes place (i.e. analyze the image, identify its content, retrieve relevant information from the data pool and send it back to the client). However, with the fast paced evolution of mobile phones and the increased processing power, it becomes more and more interesting to investigate whether the entire processing load can be allocated solely on the smartphone, and in this way avoid the shortcomings of the client/server architecture like bandwidth limitations, data transfer latencies and service dependence on the existence of WiFi or 3G network availability. Moreover, another appealing property of this architecture is its potential to scale to an unlimited number of users since the algorithms run entirely on the phone and information is stored locally, removing the possibility of overloading the centralized server by numerous requests.

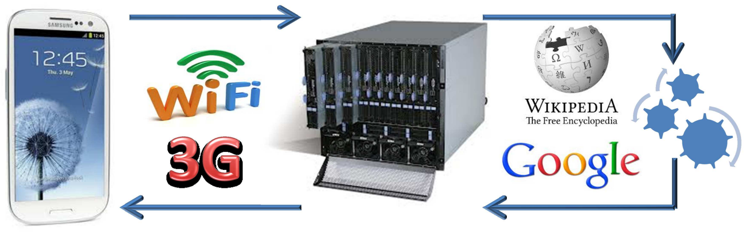


Fig. 1. Client/server architecture

In this work, we attempt to balance the computational cost and the performance of a visual recognition system so as to select a configuration that is able to run in an acceptable time frame and, at the same time, can provide satisfactory results for the specific application. We only consider the case where the entire process runs on the smartphone, and additionally look for a solution that can be executed close to real time. The goal is to identify a configuration that performs at less than one second per image in order to be appealing to the end user. In addition, considering the space and memory limitations of a mobile device environment, the typical content-based image retrieval scheme (i.e. find relevant images with the query image captured by the phone) would not be able to scale to large databases. For this reason, the objective of the proposed system falls under a classification scheme (i.e. analyze a captured image and provide keywords that describe its content), that only requires the storage of a machine learning model in the local database for each concept. Possible applications of such a scheme include the automatic annotation of mobile photo collections for keyword-based retrieval, personalized photo sharing and object recognition and annotation in real time for relevant information provision (e.g. detecting a bus and combining location information, the routes timetables can be presented).

We select one of the most popular image recognition pipelines that consists mainly of two parts, the image representation and the machine learning-classification component. For the classification component, being a field of high research interest for many decades now, many advances have been achieved, resulting in very effective and efficient algorithms [32]. It usually consists of the training step, which is the computationally expensive one, and the testing step. In a mobile recognition setting, the training step is done offline and only the testing step is performed online. Considering that the testing step can be rather fast even for very expensive algorithms, we have decided to employ Support Vector Machines (SVMs) [8], which is a state-of-the-art algorithm that can be very fast at the testing step for linear and additive kernels, while demonstrating exceptional generalization ability. On the other hand, the image representation algorithms will need to be executed online for every captured image and thus need to be carefully selected in order to achieve maximum performance at minimum cost. Towards this direction, we compare different algorithmic configurations consisting of key-point detection, feature extraction and feature encoding, so as to identify the ones that combine speed and performance. Our contribution is on thoroughly studying the existing state-of-the-art algorithms for visual recognition in a mobile environment and providing the necessary information for deciding how to optimally balance the aforementioned trade off based on the requirements of the application at hand.

The rest of this manuscript is organized as follows. In Section 2, related works are presented. In Section 3, an overview of the evaluated system architecture is described, while the two components of the system, image representation and classification, are analytically discussed in the next two Sections 4 and 5, respectively. The experimental results are shown in Section 6, while Section 7 concludes our work and discusses our plans for future work.

2 Related Work

Visual recognition has been a field of intense research interest for the past decades. This has increased even more with the recent changes in software, hardware and network technologies, i.e. digital cameras, Web 2.0 applications, smartphones, computers with high processing power, etc. This technological evolution has resulted into huge amount of multimedia content and has brought up the need for automatic visual recognition. The Query-by-example is the first appearance of image retrieval attempts that are based on visual content. ALIPR (Automatic Photo Tagging and Visual Image Search) [17] is one of the first attempts by researchers to incorporate visual content in search engines. In a similar direction, the authors of [25] present a framework for efficient large scale object retrieval that can potentially be applied on web-scale databases of billions images. Nowadays, Google search engine has employed an image retrieval service where the user can drag an image to the search box and it searches for similar images and sites with respect to its visual content in the Web.

On a mobile environment visual-based image search is even more useful and practical as it is way easier to capture an image to trigger the search than to type keywords. In general, in the visual recognition area the leap from academic research to commercial products has been much shorter and many companies have already incorporated visual recognition in mobile applications. For example, the service offered by Kooaba (www.kooaba.com) receives a snapped image as query and displays related information, further links and available files, applied to wine lists, printed catalogues, etc. oMoby (www.omoby.com) offers a shopping service that helps users find information about products by snapping a photo, such as links to retailers offering product information, reviews, prices, and more. Point & Find (pointandfind.nokia.com) is a service offered by Nokia that uses visual search technology to let users find more information about the surrounding objects, places, etc., in real time. Google Goggles (www.google.com/mobile/goggles/) is a mobile application that lets users search the web using pictures taken from their mobile phones.

On a research level, a few approaches have been presented that attempt to exploit the benefits of visual recognition in a mobile environment. The authors of [29] propose an augmented reality system, that recognizes landscapes within a database which is cached locally in the mobile phone, and presents related links to the user. They utilize a combined visual and location (GPS) based retrieval scheme and the whole system runs exclusively on the phone. In [18], a mobile application that uses visual recognition in order to find relevant documents in a locally hosted database is proposed. The authors of [33] propose lighter mobile versions of two popular feature descriptors so that the adopted natural scene tracking system can perform at frame rates of up to 20Hz.

In this work we attempt to examine the visual recognition pipeline in a mobile environment by simultaneously evaluating the image representation algorithms with respect to performance and computational cost. Our work can be considered more closely related to [13], which evaluates a retrieval scheme and focuses on the various architectures that can be employed in a mobile visual search framework. On the other hand, we select to implement the architecture which dictates that all the processing takes place on the phone and focus our experiments on comparing the various algorithmic configurations in an effort to balance performance and computational cost.

3 Framework Overview

A visual recognition framework can be considered to consist of two parts, the image representation and the classification part. In the last decade, the most popular approach that has been employed for image representation is referred to as bag of visual words (BOW) [9]. A typical pipeline for this approach is shown in Figure 2. For each image, points of interest are detected and for every key-point a set of features are extracted so as to be encoded into a single vector using the BOW approach. In order to train the vocabulary of visual words, key-point detection and feature extraction is applied on a large database of images and the features are clustered into visual words. Each component is analyzed in more detail in Section 4.

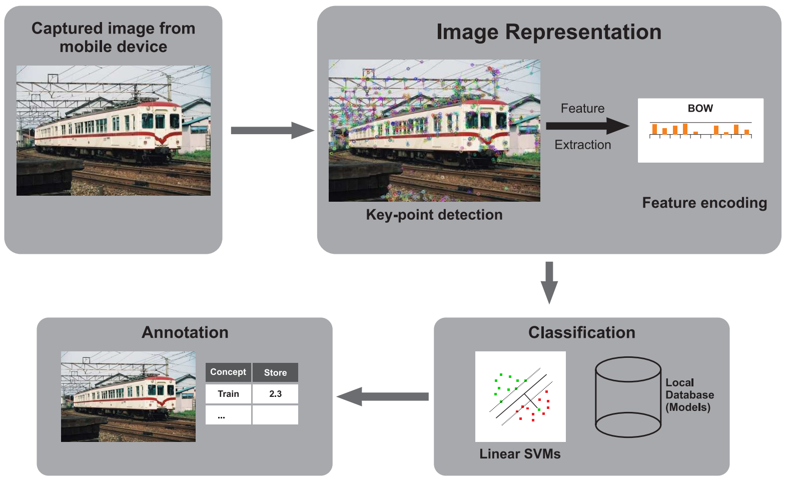


Fig. 2. Framework overview

Finally, after representing each image with a single feature vector, a model that learns the correspondence between image labels and features needs to be trained. A very popular algorithm that learns to map features to concepts is the SVMs [8], which aims at splitting the input feature space so that the different classes are separated. For every concept, the images are either labelled as positive or negative if they depict or not the concept respectively. The algorithm attempts to split the feature space by a hyperplane that maximizes the margin between the positive and the negative side. A new image is then classified as positive or negative depending on the placement of its feature vector according to the hyperplane.

Table 1. Description of key-point detection algorithms (*** stars represent the fastest and * star the slowest).

Algorithm	Speed	Translation invariance	Description	Ref.
Dense	***	✗	Key-points are selected on a grid using a step of a fixed number of pixels.	-
FAST	***	✗	Key-points are selected based on the intensities of the pixels around the examined pixel	[26]
GFTT	**	✓	Pixels are ranked based on a quality measure that is extracted using eigen analysis	[28]
HARRIS	**	✗	Detects high intensity variations of sliding windows around the pixel and defines a corner based on a score	[14]
MSER	*	✓	Detects connected regions with little change across several intensity thresholdings of the image	[21]
SIFT	**	✓	Locations at minima and maxima of a Difference of of Gaussian function are selected as key-points	[20]
STAR	*	✓	Detects the extrema of the Gaussian operator's Laplacian across scale	[1]
SURF	**	✓	Uses integral images and box filtering techniques	[3]
ORB	**	✗	Augments the FAST detector with a pyramid scheme and the Harris corner measure	[27]

4 Image Representation

4.1 Key-point Detection

Key-point detection refers to the detection of well defined positions in an image that can robustly characterize its conent. They are usually points of high interest that are used to represent important aspects of the image. Visual recognition requires repeatable key-points under several transformations such as rotation, scale changes, translation and lighting variations, whilst maintaining the detection time to the minimum. Mikolaijczyk et al. [22] review several key-point detectors on a benchmark dataset. The key-point detection algorithms that were used for our experiments can be seen at the first column of Table 1. In the second column, the relative speed of each detector is indicated by stars (more/less stars means the detector is faster/slower). In the third column, the translation invariant detectors are checked and the non invariant are crossed.

4.2 Feature Extraction

The feature extraction component attempts to describe the surrounding environment of each key-point so that it captures characteristic and indicative information of its visual content. Each key-point that was previously detected is represented as a multidimensional feature vector (descriptor). Several state-of-the-art descriptors are evaluated in [30]. In this work the evaluated descriptors are shown in Table 2, along with information about the extraction time and whether they are rotation and/or scale invariant.

Table 2. Description of feature extraction algorithms (*** stars represent the fastest and * star the slowest).

Algorithm	Speed	Size	Rotation invariance	Scale invariance	Description	Ref.
BRIEF	***	32	✗	✗	Bit string description of an image patch, constructed by a set of binary tests	[6]
ORB	***	32	✓	✗	Steered version of BRIEF according to the orientation of key-points	[27]
SURF	**	128	✓	✓	Uses box filtering techniques and integral images	[3]
SIFT	*	128	✓	✓	A histogram of orientations is computed in a 16x16 area around the key-point	[19]

4.3 Feature Encoding

After applying the key-point detection and the feature description algorithms, each image is represented by a set of multidimensional vectors. In order to transform these vectors into a single vector representation, a feature encoding algorithm is applied. The most popular method, borrowed from text retrieval, is the bag of visual words. A vocabulary of visual words is constructed from a large independent training set of images by clustering the extracted descriptors in n clusters/visual words, using an algorithm such as k-means. Afterwards, the feature encoding algorithm is applied. An extensive empirical study on encoding methods can be found at [7]. We have implemented and tested three different algorithms for encoding:

Hard assignment: The local feature descriptors of the image are matched with the visual words of the vocabulary. A histogram of the visual descriptors is populated by adding ones to the corresponding bins.

Soft (kernel codebook) assignment [31]: In this case instead of assigning a descriptor to a single corresponding visual word we assign it to k bins in a soft manner. More specifically, for every descriptor we add a quantity q to the bins of the k top nearest visual words. This quantity q is the Gaussian kernel (Radial Basis Function) distance of the descriptor and the visual word.

Vector of Locally Aggregated Descriptors (VLAD) [15]: In this case, first each descriptor is assigned to its closest visual word. Then for each visual word a vector is calculated by accumulating all the differences of the assigned descriptors with the visual word. Finally, these vectors are concatenated into a single vector representation.

In all the aforementioned cases, for each descriptor the nearest visual words must be computed. The baseline method, brute force, searches for the nearest neighbours amongst all possibilities in a greedy fashion. However, as this can be computationally expensive, approximate indexing algorithms that are significantly faster are examined. One of the most popular libraries for approximate neighbour search is the Fast Approximate Nearest Neighbor Search library (FLANN) [23], which is optimized for use in high dimensional spaces.

5 Machine Learning and Classification

The second part of a typical visual recognition system is to train a classification model for every visual concept that will learn to assign unseen images to concepts. In order to train a classification model for a specific concept we need a set of positive and a set of negative images. Each image is described by a feature vector which is extracted by the representation framework described in the previous section. Then, an SVM model is trained on these data in order to learn the properties that define the examined concept. The models are trained using the one versus all (OVA) technique, i.e. all positive examples of the specific concept versus all negative examples. The model is practically the hyperplane that separates the space of the positive and negative samples. In the case of a linear kernel, it can be represented by a vector $\mathbf{w}$, i.e. the model parameters, and a bias scalar b. For a concept c and a test image I, which corresponds to a feature vector $\mathbf{f}_I$, a confidence score is extracted by computing its distance to the hyperplane of the model for the concept c:

$$confidence(I, c) = \mathbf{w}_c * \mathbf{f}_I + b_c \tag{1}$$

The distance of a vector from the hyperplane indicates our confidence that the examined image depicts the concept of interest. High positive values of this score increase our confidence that this image belongs to the positive class while high negative values provide strong confidence that the image does not depict the concept. The fact that each model can be represented by a single vector and a scalar and that the testing process is essentially a vector multiplication, renders SVMs the best solution for a real time classification framework. This allows for storing the information about all the models in the phone memory, while testing is computationally very efficient, making it possible for the image classification algorithm to run entirely on a mobile phone.

6 Experimental Evaluation

Our intention in the experimental evaluation is to perform an extensive survey of the state-of-the-art algorithms used in a typical pipeline for visual recognition. In section 6.1 the dataset and the implementation details are explained. The strategy adopted for our evaluation experiments is to examine how the performance of each configuration is connected to the required extraction time, aiming to simultaneously improve the recognition performance and computational cost of the most prominent configurations and finally identifying the one that combines these two aspects best (Sec. 6.2).

6.1 Dataset & Implementation Details

For performing our experiments we have selected the benchmarking dataset of the PASCAL VOC 2007 competition [10]. Ground truth for both the training and the testing set of the 2007 dataset were released, while the datasets for the next years' competition lack ground truth annotations for the testing set, which is the reason the dataset from 2007 was selected instead of the more recent ones. It consists of 9.963 images collected from flickr, half of which are used for training and half for evaluating the visual recognition setups. The dataset is annotated with twenty concepts in a multi label manner (person, bird, cat, cow, dog, horse, sheep, aeroplane, bicycle, boat, bus, car, motorbike, train, bottle, chair, dining table, potted plant, sofa, tv/monitor). The split that was provided with the dataset was used, i.e. divided into two equal subsets, one for training and one for testing so that equal numbers of positive examples for each concept exist in the two subsets. This dataset was used for the performance evaluation of all different settings while the computational time estimation was performed on an image with resolution 1024 x 768 was captured from the mobile phone.

OpenCV [5], one of the most popular libraries for computer vision that also provides an interface for android cell phones was used with default parameters for the detection and description algorithms. More specifically, OpenCV was used to implement all possible combinations between the key-point detection and feature extraction algorithms described in Section 4.1 and 4.2 resulting in 36 different configurations. The feature encoding algorithms described in 4.3 were subsequently implemented using the indexing algorithms provided by FLANN. K-means was used to cluster a set of descriptors into 2000 visual words for hard and soft assignment. For the VLAD encoding method, the number of the centers was chosen so that the final feature vector was 2000-dimensional (i.e. 16 centers were extracted for the 128-dimensional SIFT and SURF, and 64 for the 32-dimensional BRIEF and ORB). For the classification part the LIBLINEAR library was selected [12]. The computational costs were computed using a Samsung Galaxy S3 device, which features a quad-core processor clocked at 1,4Ghz and 1GB RAM. In order to evaluate the various pipelines we incorporate the widely used performance measure of mean Average Precision (mAP). Average precision takes into account both precision and recall, and measures the rank quality of the retrieved images. It is calculated using Eq. 2, where $P(k)$ is the precision at rank k and $rel(k)$ is an indicator function equaling 1 if the item at rank k is a relevant document and zero otherwise. mAP is the mean of the average precision over all concepts.

$$AP = \frac{\sum_{k=1}^{n} Pr(k) * rel(k)}{\text{\# of relevant images}}, \tag{2}$$

where $rel(k)$ is an indicator function equaling 1 if the item at rank k is a relevant document and zero otherwise.

6.2 Classification Performance Versus Computational Cost

In this section the classification performance with respect to the total computation cost of each configuration is reviewed. Initially, in Table 3, the cost for each pair of key-point detection - feature description using hard binning and brute force nearest neighbour search is shown side-by-side with the recognition performance. In order to continue experimenting only with the most prominent configurations in terms of balancing the performance-cost trade-off we apply the following selection strategy. For every feature extraction approach we choose the best performing key-point detection algorithm and dismiss the ones that are more computationally expensive. Similarly, for every feature extraction approach, we choose the fastest algorithm and dismiss the ones that perform worse. If we denote as $Time(k,f)$ the computational cost and $Perf(k,f)$ the performance of a configuration composed by key-point detection method k and feature extraction method f we choose to dismiss the settings (k,f) for which:

$$\begin{aligned} & \forall f_j = \{BRIEF, ORB, SIFT, SURF\} \\ & Time(k_i, f_j) > Time(\arg\max_{k_i} Perf(k_i, f_j)) \\ or \quad & Perf(k_i, f_j) < Perf(\arg\max_{k_i} Time(k_i, f_j)) \end{aligned}$$

In this way we manage to discard cases that are computationally expensive without adding anything to performance. In the case of GFTT and HARRIS we can see in Table 3 that they need approximately the same time to compute while GFTT performs better in all cases. For this reason when choosing the fastest algorithm, we choose the GFTT even in the cases that it is not strictly the fastest (e.g. the configurations GFTT+SURF and HARRIS+SURF where although HARRIS+SURF is faster by 10 ms we select GFTT as faster). The pairs selected after applying the aforementioned selection strategy are written in bold in Table 3. Additionally, all settings are visualized in a 2D plot, in Figure 3, where the x axis represents the computational cost in logarithmic scale and the y axis the complement of the performance measure in mAP. The dismissed settings are depicted as red squares and the selected as black stars. It is obvious that the selection strategy we applied picked out the cases which have the best combination of performance and computational cost, i.e. in the area closer to the (0,0) origin.

Table 3. Time versus Performance of hard binning using brute force. The configurations that are shown in bold are the most prominent ones and are selected for further experimentation.

	Time (ms)				**Performance (mAP)**			
	BRIEF	**ORB**	**SIFT**	**SURF**	**BRIEF**	**ORB**	**SIFT**	**SURF**
DENSE	**3130**	**2807**	**11546**	10847	**15,70%**	**15,17%**	**17,30%**	13,16%
FAST	3862	3095	**39055**	**13645**	13,26%	12,91%	**17,86%**	**17,59%**
GFTT	**555**	**564**	**2156**	**884**	**11,64%**	**11,39%**	**15,80%**	**13,27%**
HARRIS	603	873	2146	874	10,49%	9,90%	13,61%	10,72%
MSER	3322	3298	34645	3843	9,71%	8,87%	14,86%	12,40%
SIFT	631	605	3865	1400	10,04%	9,66%	11,49%	12,16%
STAR	1857	1712	12558	**2718**	9,72%	9,66%	14,96%	**13,68%**
SURF	2081	1845	56651	**5038**	11,51%	10,42%	16,92%	**15,75%**
ORB	761	913	5528	**1227**	11,06%	10,72%	13,00%	**14,15%**

For the remaining configurations we attempt to optimize both the computational cost and the performance. The first step is to replace the expensive brute

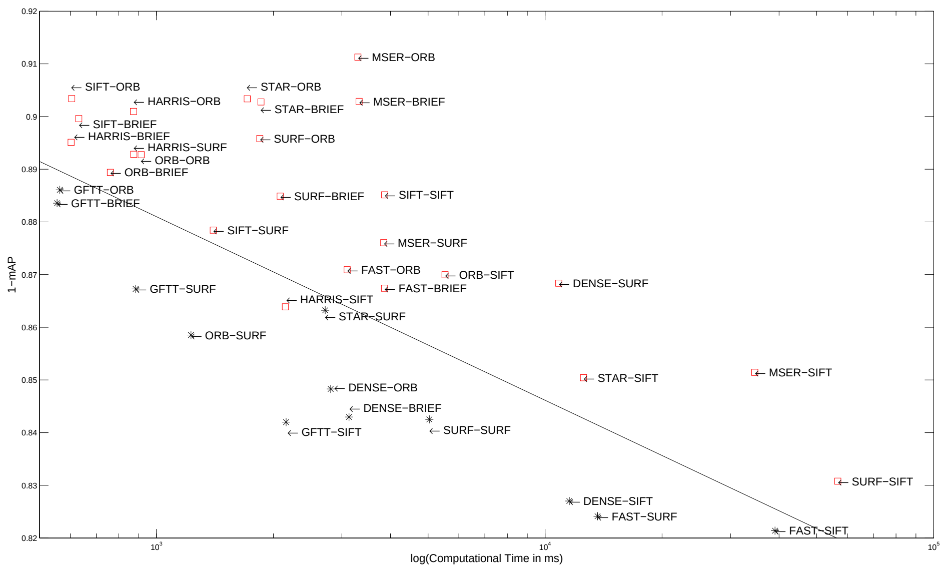


Fig. 3. Time versus Performance of hard binning using brute force

force search for the nearest neighbour in the hard binning setting with its faster approximate version, FLANN. The results are shown in Table 4, from which we can see that there is a significant gain in computational time, especially in the DENSE, FAST and SURF configurations (shown in bold). This can be explained by the fact that the these detectors tend to extract more key-points than the rest, so the main processing load is in the encoding part of the algorithm. Moreover, there are only slight variations in performance with respect to the brute force method, which verifies that the approximate nearest neighbour algorithm works almost as good as in the brute force case. Considering that in other encoding methods we will need to search for even more nearest neighbours, the use of the approximate version is mandatory.

Table 4. Time versus Performance of hard binning using FLANN. The computational cost is significantly decreased with minor changes in performance. In bold the configurations where the decrease in maximum.

	Time (ms)				**Performance (mAP)**			
	BRIEF	**ORB**	**SIFT**	**SURF**	**BRIEF**	**ORB**	**SIFT**	**SURF**
DENSE	**1948**	**1349**	**3026**		15,51%	15,92%	18,02%	
FAST	**2506**	**1757**	**29767**	**4274**	13,26%	12,90%	18,96%	17,18%
GFTT	576	573	1863	560	11,62%	11,23%	16,21%	12,63%
STAR				2063				11,14%
SURF				**2279**				17,09%
ORB				1025				13,73%

The next step is to increase the performance, and towards this goal we evaluate two popular substitutes to the hard encoding method, the soft assignment and the VLAD encoding algorithms. Tables 5 and 6 depict the results for each encoding algorithm. In the case of soft assignment, it is obvious that the performance is significantly improved in all cases with a relatively low loss in efficiency, especially in the cases where the number of detected key-points are limited (e.g. GFTT, STAR, ORB). Using the VLAD encoding method, the performance of all configurations increases, especially for the configurations with the SURF descriptor, where the gain is notably higher. The best performing configuration is the FAST-SURF-VLAD, which yields 31.35% mAP and requires 6.4 seconds to execute. However, we can see that for the minor performance loss of 0.25% we can use the SURF-SURF-VLAD combination in less than half the time of the previous scheme. For this reason, we select the SURF-SURF-VLAD scheme to continue with further enhancements.

An important note is that the time measurements were performed on a typical image taken by the cell phone which by default has a resolution of 1024×728. On the other hand, the performance measures were calculated on the PASCAL dataset which consists mainly of smaller images, on the average scale of 300×500 resolution. Thus, for the SURF-SURF-VLAD configuration, we only need to capture images at the scale of 300×500 in order to achieve the 31.1% mAP which decreases the time to only 0.5 seconds per image. This reduced time requirement renders the proposed configuration ideal for real-time mobile visual recognition. In order to achieve better performance, we can tweak the default values of OpenCV for the SURF detector. More specifically, if we change the Hessian threshold from 100 to 50 and 10, the performance increases to 31.72% and 32.6% mAP respectively, while the computational cost increases to 0.96 and 1.42 seconds respectively. Considering that our initial goal was to achieve a computational time no more than 1 second, we choose the configuration with the Hessian threshold set to 50, which yields 31.72% in 0.96 seconds. Finally, a few images with the proposed annotations from the selected configuration are shown in Table 7.

Table 5. Time versus Performance of soft binning using FLANN. The most appealing configuration is shown in bold.

	Time (ms)				**Performance (mAP)**			
	BRIEF	**ORB**	**SIFT**	**SURF**	**BRIEF**	**ORB**	**SIFT**	**SURF**
DENSE	3285	2684	4629		21,54%	21,62%	23,72%	
FAST	3990	3170	31278	5763	18,83%	19,01%	30,14%	26,01%
GFTT	602	613	1875	601	16,34%	15,85%	24,20%	18,58%
STAR				2134				15,06%
SURF				**2669**				**25,99%**
ORB				1050				19,49%

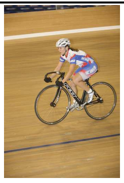
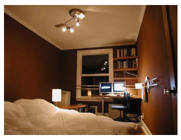
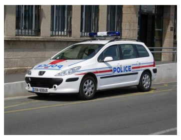
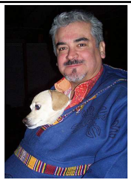
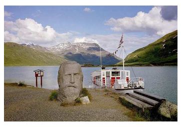
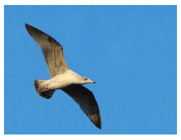
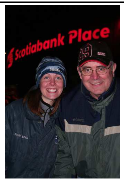
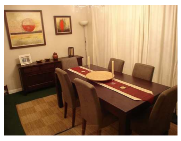
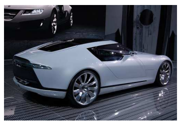
7 Conclusions & Future Work

In this paper, we have reviewed state-of-the-art algorithms for image representation and compared their performance versus the computational cost when

Table 6. Time versus Performance of VLAD encoding. The most appealing configuration is shown in bold.

	Time (ms)				Performance (mAP) (%)			
	BRIEF	**ORB**	**SIFT**	**SURF**	**BRIEF**	**ORB**	**SIFT**	**SURF**
DENSE	3381	2053	4949		20,53	20,42	27,98	
FAST			32213	6492			29,63	31,35
GFTT	526	543	1920	632	16,58	16,33	17,52	20,97
STAR				2186				17,15
SURF				**2954**				**31,10**
ORB				1019				20,84

Table 7. Images with proposed annotations by our framework. In bold the correct proposals. Some of the mistakes can be partly justified in the case of visually similar concepts, e.g. aeroplane-bird, person-statue.

	**bicycle**, **person**		**chair**, **tv-monitor**		**car**
	**person**, **dog**		person		aeroplane, **bird**
	**person**		**chair**, person, tv-monitor		**car**

applied in a mobile environment for image classification. The performance was measured in a real world dataset originating from flickr, which renders the work of visual recognition a very challenging task. The results show that close to real time performance can be achieved with various configurations by sometimes sacrificing performance for the gain of speed. The case of SURF+SURF+VLAD seems the most appealing since for 300 × 500 resolution images it requires only 0.96 seconds to achieve 31.72% mAP.

Our plans for future work are mostly directed by our intention to increase the performance of the mobile recognition framework. More specifically, they include the testing of the FREAK descriptor [2] that was recently published and released by OpenCV, the evaluation of the promising super vector encoding method [34] and the enhancement of the performance by utilizing the popular spatial pyramids [16]. Finally, the exploitation of the visual recognition framework in real cases and the evaluation of the algorithm in more representative datasets for the use cases is in our intensions as well.

Acknowledgement

This work was supported by the EU 7th Framework Programme under grant number IST-FP7-288815 in project Live+Gov (www.liveandgov.eu)

References

1. M. Agrawal, K. Konolige, and M. R. Blas. Censure: Center surround extremas for realtime feature detection and matching. In *ECCV (4)*, volume 5305 of *Lecture Notes in Computer Science*, pages 102–115. Springer, 2008.
2. A. Alahi, R. Ortiz, and P. Vandergheynst. Freak: Fast retina keypoint. In *IEEE Conference on Computer Vision and Pattern Recognition, Providence, RI, USA, June 16-21, 2012*.
3. H. Bay, A. Ess, T. Tuytelaars, and L. Van Gool. Speeded-up robust features (surf). *Comput. Vis. Image Underst.*, 110(3):346–359, June 2008.
4. D. Berg. Apple says: Mobile application performance matters. *http://www.apmdigest.com/apple-says-mobile-application-performance-matters*, October 29, 2012.
5. G. Bradski. The OpenCV Library. *Dr. Dobb's Journal of Software Tools*, 2000.
6. 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, Berlin, Heidelberg, 2010. Springer-Verlag.
7. K. Chatfield, V. Lempitsky, A. Vedaldi, and A. Zisserman. The devil is in the details: an evaluation of recent feature encoding methods. In *British Machine Vision Conference*, 2011.
8. C. Cortes and V. Vapnik. Support-vector networks. In *Machine Learning*, pages 273–297, 1995.
9. G. Csurka, C. R. Dance, L. Fan, J. Willamowski, and C. Bray. Visual categorization with bags of keypoints. In *In Workshop on Statistical Learning in Computer Vision, ECCV*, pages 1–22, 2004.
10. M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The PASCAL Visual Object Classes Challenge 2007 (VOC2007) Results. http://www.pascal-network.org/challenges/VOC/voc2007/workshop/index.html.
11. M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The PASCAL Visual Object Classes Challenge 2012 (VOC2012) Results. http://www.pascal-network.org/challenges/VOC/voc2012/workshop/index.html.
12. R.-E. Fan, K.-W. Chang, C.-J. Hsieh, X.-R. Wang, and C.-J. Lin. Liblinear: A library for large linear classification. *J. Mach. Learn. Res.*, 9:1871–1874, 2008.
13. B. Girod, V. Chandrasekhar, D. M. Chen, N.-M. Cheung, R. Grzeszczuk, Y. A. Reznik, G. Takacs, S. S. Tsai, and R. Vedantham. Mobile visual search. *IEEE Signal Process. Mag.*, 28(4):61–76, 2011.
14. C. Harris and M. Stephens. A combined corner and edge detector. In *In Proc. of Fourth Alvey Vision Conference*, pages 147–151, 1988.
15. H. Jégou, M. Douze, C. Schmid, and P. Pérez. Aggregating local descriptors into a compact image representation. In *IEEE Conference on Computer Vision & Pattern Recognition*, pages 3304–3311, jun 2010.

16. S. Lazebnik, C. Schmid, and J. Ponce. Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In *Proceedings of the 2006 Conference on Computer Vision and Pattern Recognition*, pages 2169–2178, 2006.
17. J. Li and J. Z. Wang. Real-time computerized annotation of pictures. *IEEE Trans. Pattern Anal. Mach. Intell.*, 30(6):985–1002, June 2008.
18. X. Liu, J. J. Hull, J. Graham, J. Moraleda, and T. Bailloeul. Mobile visual search, linking printed documents to digital media. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2010.
19. D. G. Lowe. Object recognition from local scale-invariant features. In *Proceedings of the International Conference on Computer Vision-Volume 2 - Volume 2*, ICCV '99, pages 1150–, Washington, DC, USA, 1999. IEEE Computer Society.
20. D. G. Lowe. Distinctive image features from scale-invariant keypoints. *Int. J. Comput. Vision*, 60(2):91–110, 2004.
21. J. Matas, O. Chum, M. Urban, and T. Pajdla. Robust wide-baseline stereo from maximally stable extremal regions. *Image Vision Comput.*, 22(10):761–767, 2004.
22. K. Mikolajczyk and C. Schmid. A performance evaluation of local descriptors. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(10):1615–1630, Oct. 2005.
23. M. Muja and D. G. Lowe. Fast approximate nearest neighbors with automatic algorithm configuration. In *International Conference on Computer Vision Theory and Application VISSAPP'09)*, pages 331–340. INSTICC Press, 2009.
24. P. Over, G. Awad, J. Fiscus, A. F. Smeaton, W. Kraaij, and G. Qunot. TRECVID 2011 – An Overview of the Goals, Tasks, Data, Evaluation Mechanisms and Metrics. In *Proceedings of TRECVID 2011*. NIST, USA, dec 2011.
25. J. Philbin, O. Chum, M. Isard, J. Sivic, and A. Zisserman. Object retrieval with large vocabularies and fast spatial matching. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2007.
26. E. Rosten and T. Drummond. Machine learning for high-speed corner detection. In *Proceedings of the 9th European conference on Computer Vision - Volume Part I*, ECCV'06, pages 430–443, Berlin, Heidelberg, 2006. Springer-Verlag.
27. E. Rublee, V. Rabaud, K. Konolige, and G. Bradski. Orb: An efficient alternative to sift or surf. In *International Conference on Computer Vision*, Barcelona, 2011.
28. J. Shi and C. Tomasi. Good features to track. In *1994 IEEE Conference on Computer Vision and Pattern Recognition (CVPR'94)*, pages 593 – 600, 1994.
29. G. Takacs, V. Chandrasekhar, N. Gelfand, Y. Xiong, W.-C. Chen, T. Bismpigiannis, R. Grzeszczuk, K. Pulli, and B. Girod. Outdoors augmented reality on mobile phone using loxel-based visual feature organization. In *Proceedings of the 1st ACM international conference on Multimedia information retrieval*, pages 427–434, 2008.
30. K. van de Sande, T. Gevers, and C. Snoek. Evaluating color descriptors for object and scene recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 99(1), 2008.
31. J. C. van Gemert, C. J. Veenman, A. W. M. Smeulders, and J. M. Geusebroek. Visual word ambiguity. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(7):1271–1283, 2010.
32. V. N. Vapnik. *Statistical learning theory*. Wiley, 1 edition, 1998.
33. D. Wagner, G. Reitmayr, A. Mulloni, T. Drummond, and D. Schmalstieg. Pose tracking from natural features on mobile phones. In *Proceedings of the 7th International Symposium on Mixed and Augmented Reality*, 2008.
34. X. Zhou, K. Yu, T. Zhang, and T. S. Huang. Image classification using super-vector coding of local image descriptors. In *Proceedings of the 11th European conference on Computer vision: Part V*, ECCV'10, 2010.