Lightweight Sign Recognition for Mobile Devices

Michele Fornaciari*, Andrea Prati†, Costantino Grana* and Rita Cucchiara*
*DIEF, University of Modena and Reggio Emilia, Modena, 41125, Italy
Email: {michele.fornaciari,costantino.grana,rita.cucchiara}@unimore.it
†DPPAC, University IUAV of Venice, Venice, 30135, Italy
Email: andrea.prati@iuav.it

***Abstract*—The diffusion of powerful mobile devices has posed the basis for new applications implementing on the devices (which are embedded devices) sophisticated computer vision and pattern recognition algorithms. This paper describes the implementation of a complete system for automatic recognition of places localized on a map through the recognition of significant signs by means of the camera of a mobile device (smartphone, tablet, etc.). The paper proposes a novel classification algorithm based on the innovative use of bag-of-words on ORB features. The recognition is achieved using a simple yet effective search scheme which exploits GPS localization to limit the possible matches. This simple solution brings several advantages, such as the speed also on limited-resource devices, the usability also with limited training samples and the easiness of adapting to new training samples and classes. The overall architecture of the system is based on a REST-JSON client-server architecture. The experimental results have been conducted in a real scenario and evaluating the different parameters which influence the performance.**

I. Introduction

Mobile vision is a rather recent research field aiming at developing sophisticated computer vision algorithms on board of mobile devices. More specifically, this field addresses the use of commercial smartphones and tablets as embedded devices, thanks to their increasing capabilities to process complex data (such as images and videos) in real time and to the availability of several good-quality sensors on board.

This research field must not be considered as a mere re–engineering task of existing algorithms on a different hardware platform, since it poses several challenges which require a complete re–design of the algorithms themselves and can be of interest for the scientific community: the limited computational and memory resources available, the extreme mobility and limited connectivity, and the power constraints under which these tether-less devices operate.

The use of mobile devices as people-centric sensors and processing units opens to new and impressive classes of applications, ranging from real time object/person tracking, content-based retrieval of framed scene with markerless object recognition, face detection (and possibly recognition), blind people aid for movement, etc.. This paper addresses an application with the final aim of recognizing logos for localization purposes. The proposed application needs to identify precisely a place where the user is located, supposing the GPS-based localization is not available or reliable (both for limited precision and for the co-existence of several significant locations in the neighborhood). This identification can be used for city marketing (similarly to the "check in" used in apps like Foursquare) or for gaming purposes. Other sensors may be employed too (e.g., the gyroscope to understand the direction the user is looking at), but precise localization is still required. Moreover, the place identification through image analysis allows the system to work properly also indoor (e.g., in a mall) where GPS-based localization is unreliable.

With these premises, this paper proposes a computer vision algorithm which recognizes significant locations in real-time. The algorithm is based on ORB features which are then quantized with a newly-proposed bag-of-words (BoW) model for binary descriptors. At the best of our knowledge, this is the first time a BoW model for a binary descriptor (as ORB is) is used in the context of logo recognition. Moreover, to take into account the aforementioned bandwidth limitations, the BoW histograms are compressed using an efficient and effective compression algorithm. Server side, this compressed descriptor is used to match with the trained classes in a database and the resulting ranking is returned to the user on the device. Experiments on real images with challenging logos are reported.

II. Related Works

The problem we are addressing is somehow similar to logo recognition [1], [2], in that we aim at recognizing the location depicted in the query image by means of appearance. Logo recognition solutions rely on segmented logos or brands and learn particular configurations of local descriptors to match with the query [3], [4]. Since in our case the segmentation is not available, we consider the images as a whole and search for visually similar images, treating the problem as an image matching one [5]. However, beside the lack of segmentation, these methods are not directly applicable because we do not have large datasets for the comparison [6]. Also, they are in general computationally demanding and thus not suited for user interaction.

Most of the methods mentioned above rely on feature detection and matching. The SIFT keypoint detector and descriptor [7], although over a decade old, have proven to be remarkably successful in a number of applications using visual features, including object detection and recognition, image stitching, scene classification, etc. This descriptor has been also extended to color images in the form of RGB-SIFT, Opponent-SIFT and C-SIFT, as described by van de Sande *et al.* in [8]. However, it requires an intensive computational effort, especially for real-time systems, or for low-power devices such as cellphones. This has led to an increased research for replacements with simpler descriptors with lower computation demands. This trend started with SURF [9], but since then a lot of other descriptors have been proposed in literature,

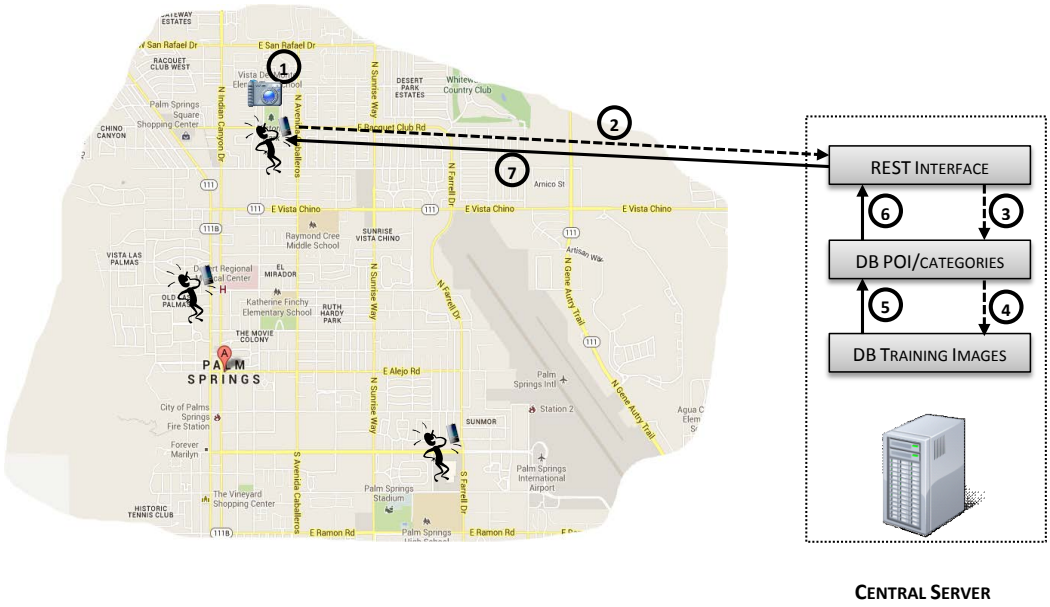

Fig. 1. Overall description of the client-server architecture.

always focusing not only on performance but also on speed: we can refer to VLAD [10], BRIEF [11], DAISY [12] among the most recent proposals. There has also been research aimed at speeding up the computation of SIFT, most notably with GPU devices [13], or the exploitation of approximate nearest neighbor techniques, starting from LSH [14] up to product quantization [15].

Also a great variety of global features has been proposed to tackle the retrieval problem, for example color histograms, GIST [16] and HOG [17]. Usually these features are easier to compute and do not require the quantization step typical of the bag-of-words model which is necessary to create a global representation (an histogram of visual words) from the aforementioned local descriptors.

In a recent paper, Rublee *et al.*[18] propose a very fast binary descriptor based on BRIEF, called ORB, which is rotation invariant and robust to noise. They demonstrate through experiments how ORB is up to two orders of magnitude faster than SIFT, while performing as well in many situations. The efficiency is tested on several real-world applications, including object detection and patch-tracking on a smartphone. The investigation of variance under orientation was critical in constructing ORB and decorrelating its components, in order to get good performance in nearest-neighbor applications. An interesting aspect is that the authors have also contributed a BSD licensed implementation of ORB to the community, via OpenCV 2.3. We provide a short description of the ORB descriptor in Section IV-A.

III. System Overview

The overall sketched architecture of the proposed system is shown in Fig. I. The system follows a standard client-server architecture, where the client side is composed of the users' mobile devices, while the server side is a central computer devoted to keep the databases updated and to provide the logo classification. When dealing with mobile vision (and mobile computing in general) there is always a crucial tradeoff between the local (on-device) computation and the remote processing. When moving towards the latter, the communication overhead required to send the raw data to the remote server can be unacceptable, especially because of the cost of transmission of the device (not-flat rates) and the delay introduced to transmit large amount of data. On the opposite side, local processing requires a sufficiently-powerful device to avoid long waits and it consumes a lot of battery.

In our system, the balance is reached as follows. First, the user will use the developed App to take a picture of a logo (phase 1 in Fig. I). It is worth saying that the picture does not have to be perfect, and it can be at some degree blurred by the user's movement, it can contain small logos, and the logo can be acquired in a tilted, partial or occluded way, at least at a reasonable extent. The only strong requirement is that the picture does not contain other logos, at a resolution and quality superior to that of the searched logo. Once the picture has been taken, it is processed locally on the device to extract and compress, in real time, a set of features and the corresponding bag-of-words (BoW) descriptor (further details are reported in Section IV). The compressed descriptor is sent to the remote server (phase 2 in Fig. I). In order to obtain en efficient communication between the clients and the server we used the REST (REpresentational State Transfer) architecture for distributed systems [19]. The main advantage is that REST architecture provides a simple interface with the server where the requests are sent through a HTTP GET with parameters passed through the URL. The server responds (phase 7 of Fig. I) using the JSON (JavaScript Object Notation) format which codes the response as a formatted string.

In order to prepare the JSON response the server needs to identify the list of POIs (Points Of Interest) which potentially match with the logo in the picture taken by the users. This list is ordered based on the score computed by the algorithm and returned to the user's device. This identification is achieved by matching the descriptor sent by the user with the descriptors

of the trained class, as detailed in Section IV. First of all, the set of potential POIs (with corresponding categories) is retrieved by the database (phase 3 in Fig. I). This retrieval can be narrowed with respect to the total number of POIs in the database by exploiting the rough GPS-based location of the device (which is sent together with the descriptor in phase 1), if available.

Given the set of potential POIs, the database containing the trained images for each one is queried (phase 4) and the server-side of the machine learning algorithm is activated (see Section IV-D). This algorithm returns (phase 5) the ordered list of POIs ranked according to a score, which is then sent back to the device through phases 6 and 7.

IV. Logo Recognition through Bag-of-Words and ORB features

The system requires to efficiently extract some informative elements from the acquired images on the device, so the choice of the ORB descriptor (IV-A) is straightforward and particularly effective, given also the availability of specific implementations for mobile devices. What we need is a way to quickly summarize these local binary descriptors, which also allows us to reduce the amount of information sent to the server. We employ two solutions for this task, leveraging a variation of the k-means algorithm (IV-B) and a very simple, but effective compression scheme (IV-C). Finally, server-side, we detect the image class, thus the sign queried by the user, by means of a simple ranking technique (IV-D) which uses a subset of the training images, selected according to their GPS position.

A. ORB descriptor

The ORB descriptor (Oriented FAST and Rotated BRIEF) builds on the well-known FAST keypoint detector [20] and the recently-developed BRIEF descriptor [11].

The original FAST proposal implements a set of binary tests over a patch, by varying the intensity threshold between the center pixel and those in a circular ring around the center. The Harris corner measure [21] has been used to provide an evaluation of the corner intensity. In ORB the missing orientation information of FAST are instead complemented with Rosin's corner intensity [22]. In particular, the moment m_{pq} of a patch (region) R is computed as:

$$m_{pq} = \sum_{x,y} x^p y^q R(x,y). \tag{1}$$

We further compute the centroid $\mathbf{c}$ (boldface is used for vectors) as

$$\mathbf{c} = \left(\frac{m_{10}}{m_{00}}, \frac{m_{01}}{m_{00}} \right), \tag{2}$$

and by constructing a vector from the patch center to the centroid $\mathbf{c}$, we define the relative orientation of the patch as

$$\omega = \text{atan2}\left(m_{01}, m_{10}\right). \tag{3}$$

The patch description has been provided starting from the BRIEF operator [11], a bit string representation constructed from a set of binary intensity tests. Given a smoothed image patch R of an intensity image I, a binary test τ can be performed as:

$$\tau\left(R, \mathbf{u}, \mathbf{v}\right) = \begin{cases} 1: & R(\mathbf{u}) < R(\mathbf{v}) \\ 0: & R(\mathbf{u}) \geq R(\mathbf{v}) \end{cases}. \tag{4}$$

Given a set of intra-patch locations

$$L = \begin{pmatrix} \mathbf{u}_1, \ldots, \mathbf{u}_n \\ \mathbf{v}_1, \ldots, \mathbf{v}_n \end{pmatrix}, \tag{5}$$

the final feature $\mathbf{b}$ is defined as an n-dimensional vector of binary tests:

$$\mathbf{b}(R) = \sum_{1 \leq i \leq n} 2^{i-1} \tau(R, \mathbf{u}_i, \mathbf{v}_i). \tag{6}$$

The intra-patch locations for the tests influences the quality of the descriptor itself. A solution could be a grid–sampling–based set of sets by taking into consideration the patch orientation, so multiplying these locations with the rotation matrix. However, by analyzing the distribution of the tests, this solution brings a loss of variance and increase the correlation among the binary tests (since tests along the edge orientation statistically produce similar outcomes). This heavily impacts the descriptor effectiveness, describing redundancy more than distinctiveness. To solve the problem, the authors [18] employed a learning algorithm, sampling tests from 5×5 subwindows of the 31×31 patch window chosen for the descriptor, running each test against all training patches. The result is a predefined set of 256 tests called rBRIEF.

Algorithm 1 K-majority algorithm

```
Given a collection D of binary vectors
Randomly generate k binary centroids C
repeat
    for d ∈ D do                                  ▷ Assign data to centroids
        c_d ← arg min_{c∈C} HammingDistance(c, d)
    end for
    for c ∈ C do                                  ▷ Majority voting
        for d ∈ D|c_d = c do
            v accumulates d votes
        end for
        c' ← Majority(v)
    end for
until centroids not changed
```

B. A Bag of Words Model for Binary Descriptors

While histogram based features are directly ready to be used in image classification or retrieval tasks, local features require an additional quantization step to be transformed into global image features. The classic approach is to employ k-means clustering using Euclidean distance between feature vectors, and this has proved to be effective, even if computationally demanding during the training phase.

Unfortunately when dealing with a vector of binary features, Euclidean distance is not the metric of choice, and the average vector is undefined. A reasonable and effective distance between binary vectors is the Hamming distance (the number of different bits in corresponding positions), but still no average is provided. We could tackle the problem reverting to

```
unsigned HammingDistance (__m128i *x, __m128i *y) {
  __m128i xorValue = _mm_xor_si128(*x,*y);
  return (unsigned)__popcnt64(xorValue.m128i_u64[0])
    + (unsigned)__popcnt64(xorValue.m128i_u64[1]);
}
```

Fig. 2. Example Hamming distance function in C language, using SSE4 instruction on a 64bit architecture.

k-medoids (PAM algorithm) [23], but this would require the computation of a full distance matrix between the elements to be clustered, even worsening the problem. Therefore, to compute the centroid of a set of binary vectors based on the Hamming distance, we introduce a voting scheme. In particular, corresponding elements of each vector vote for 0 or 1. For determining each element of the centroid, the majority rule is used, with ties broken randomly. We call this variation of the Lloyd algorithm "k-majority algorithm" and we resume it in Algorithm 1.

The algorithm processes a collection of D binary vectors and seeks for a number k of good centroids, that will become the visual dictionary for the Bag-Of-Words model. Initially, these k centroids are determined randomly (line 2). At each iteration, the initial step (lines 4-6) is the assignment of each binary vector to the closest centroid: the current binary vector d is therefore labelled with the index of the closest centroid. This part is essentially shared by many common clustering algorithms. The second step (lines 7-12) is the majority voting used to redefine the vector clustering. For each cluster c, we take into consideration every binary vector d belonging to it. Every bit of an accumulator vector v is increased by 1 if the corresponding bits in d is 1. At the end, the majority rule is used to form the new centroid c': for each element v_i in v, if the majority of vectors voted for 1 as bit value in v_i, then c'_i takes 1, otherwise c'_i takes 0. The algorithm iterates until no centroids are changed during the previous iteration.

A fundamental advantage of the k-majority approach is that both the Hamming distance and the majority voting step can work on the byte packed vector string. In particular the Hamming distance may be implemented leveraging both SSE instructions and specific bitwise hardware instructions. An optimized version of an Hamming distance function is provided in Fig. 2.[1] By using Hamming distance and majority voting in the cluster assignment step, which has to go through all elements to be clustered, we can obtain a speedup over classical k-means in the order of 100.

C. BoW Descriptors Lossless Compression Scheme

To reduce the bandwidth requirements of the system (which is of paramount importance when mobile devices are considered), an extremely simple and fast lossless compression scheme is applied to the BoW histograms. The first observation is that these descriptors are significantly sparse and become more and more sparse at growing size of the codebook. For this reason, similarly to the JPEG coding of AC coefficients, we employ a RLE of zeros and describe every non zero value as the number of zero values before the current one.

[1]If specialized hardware instructions are not available, it is still possible to employ some smart bitwise operations as those proposed in http://graphics.stanford.edu/~seander/bithacks.html#CountBitsSetParallel

TABLE I. COMPRESSION TESTS COMPARISONS. FOR BOTH GZIP AND OUR COMPRESSION SCHEME THE AVERAGE SIZE IN BYTES AND THE AVERAGE COMPRESSION RATIO ARE REPORTED

centers	uncompressed binary	compressed gzip	compressed our	our/gzip
32	128	121 (1.06)	117 (1.10)	0.96
64	256	171 (1.50)	148 (1.73)	0.87
128	512	229 (2.23)	165 (3.11)	0.72
256	1,024	297 (3.45)	197 (5.20)	0.66
512	2,048	390 (5.25)	253 (8.09)	0.65
1K	4,096	517 (7.93)	328 (12.49)	0.63
2K	8,192	674 (12.16)	426 (19.24)	0.63
4K	16,384	854 (19.20)	542 (30.25)	0.63
8K	32,768	1,057 (31.01)	669 (48.99)	0.63
16K	65,536	1,300 (50.42)	804 (81.50)	0.62
32K	131,072	1,582 (82.86)	947 (138.40)	0.60

The second observation is that the distribution of nonzero value is definitely non uniform, with some values found many times and others really seldom. Huffman coding could be applied, but this would require a specific dictionary construction for every descriptor or a predefined dictionary which could be biased toward the specific dataset in use and possibly become not really useful at changing conditions. For these reasons we apply a universal coding technique, the Elias gamma code, known also as Exp-Golomb coding as used in the H.264/MPEG-4 AVC and Dirac video compression standards.

For the sake of simplicity, the original BoW histograms are stored as a stream of 32 bits wide floating point values, so we start our representation with a list of nonzero floating point values ordered by the most probable to the least probable ones, then encode the floating point values as an Elias encoded run length of zeros and an Elias encoded index pointing to the value table. The run lengths are not exactly exponentially distributed, so Elias coding is not the perfect choice for their representation, but this still allows for a very effective representation at small values, without adding a large overhead to longer representations.

In Table I we report the average compression obtained both with GZip standard compression applied over the uncompressed binary data and with our compression scheme. Our approach produces descriptor which are on average 30% smaller then GZip compression, while being much faster, because of the use of Elias encoding. In fact no search is required and the representation is a straightforward variable-length representation of a binary coded integer value. Higher compression rates could be obtained, at the price of a higher complexity.

D. Similarity Search

Once the query descriptor is received by the server, it is compared with the descriptors of all training images. The number of comparison can be largely reduced using only the descriptors of the locations that lie within a given radius from the current user position, provided by the GPS embedded on the mobile device. The descriptors are thus ranked according to the similarity measure given by the histogram intersection. We determine the similarity for each class computing the Average Precision (AP) on this ranked list as if the given class were the correct one, and propose back to the user the list of classes sorted according to their AP.

Fig. 3. Samples from the reference dataset, 1 column per class. Images depict the same location with different viewpoints, light conditions and camera resolutions.

With this simple scheme we avoid the need to train a multi-class classifier, with several advantages: i) the exhaustive similarity search is still feasible thanks to the GPS position constraint; ii) there is no need to re-train every time a new image (of an existing or new class) is added because we can compute the global descriptors of these images leveraging the same cluster centers; iii) the number of images per class may not be balanced, thanks to the AP re-ranking and, actually, iv) a class may be composed even by a single image.

V. Experimental results

The proposed system has been deployed on a wide range of smart phones and tablets, ranging from Samsung Galaxy Tab to Sony Xperia Z. The most important differences are computational power (from single core @ 1 GHz to quad core @ 1.5 GHz) and camera resolution (from 3.2 MP to 13 MP). In order to evaluate the accuracy, we tested offline the similarity search on two datasets. The first dataset, Significant Signs (SS), consists of 614 images: 464 depict shots of 96 different significant locations, the remaining 150 are noise. As depicted in Fig. 3, within each class each photo depicts the same location, taken from different points of view, with different devices, in different light conditions. The second dataset is the FlickrLogos-32 dataset [24] (FL32), which contains 30 photos showing brand logos for 32 logo classes, as long as 6000 non–logo images. It is meant for the evaluation of multi-class logo recognition as well as logo retrieval methods on real world images.

As local image descriptor we selected the ORB (see Sect. IV-A), because it is very fast to compute, and, being a binary descriptor, allows us for fast clustering via the K-majority algorithm (see Section IV-B) that can be easily accomplished on-board of a mobile device. We also aim at demonstrating that the ORB descriptor provides also better retrieval than SIFT [7]. The accuracy obtained generating the global descriptor starting from ORB and SIFT, varying the number of cluster centers of the BoW model, i.e. the dimension of the global descriptor, has been evaluated. We computed the cluster centers starting from the local descriptors of all images, including noise, and used all images, except noise, as queries. The evaluation of the *Mean Average Precision (MAP)* in Fig. 4 shows that global descriptors generated from ORB perform in general much better than SIFT. Moreover, ORB provides the correct result as first response more often than SIFT, as depicted in Fig. 5.

We want our system to work irrespective of the device,

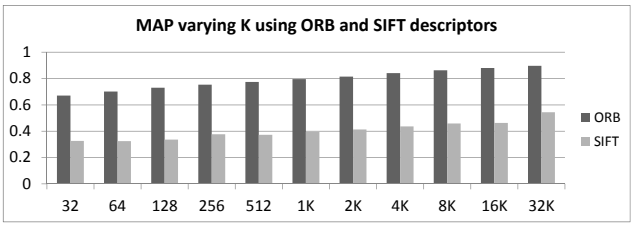

Fig. 4. Mean Average Precision computed on the SS dataset varying the number K of cluster centers.

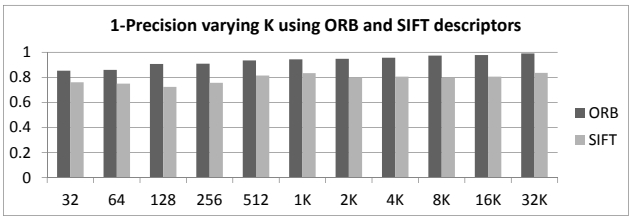

Fig. 5. 1-Precision computed on the SS dataset varying the number K of cluster centers.

so we must handle photos taken with different cameras at various resolutions. Since we work under the assumption that the logo is centered in the image and is clearly visible (some examples are reported in Fig. 3), we overcome the issue simply resizing the images. This allows us to normalize the sizes, so that their local descriptors, which are computed on fixed size patch, have comparable meanings. We report in Fig. 6 and Fig. 7 the results (obtained with $K = 512$ cluster centers) varying the dimensions of the resized images on the two datasets. The size of the images does not affect much the quality of the results, so we can use smaller sizes that guarantee a faster local descriptor computation.

We investigate also the number K of cluster centers of the BoW model, which directly affects the size of the global descriptor and the computational time. We tested our ORB–based descriptor on images resized to 320×240 on both

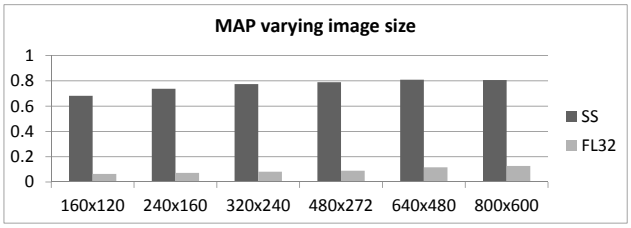

Fig. 6. Mean Average Precision computed on the two datasets (SS and FL32) varying the size of the images.

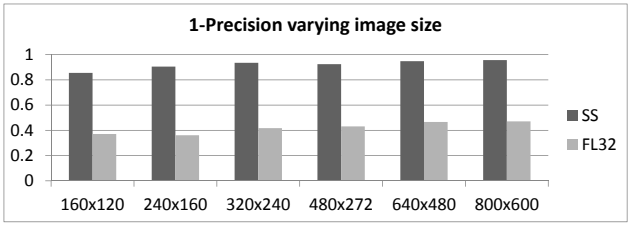

Fig. 7. 1-Precision computed on the two datasets (SS and FL32) varying the size of the images.

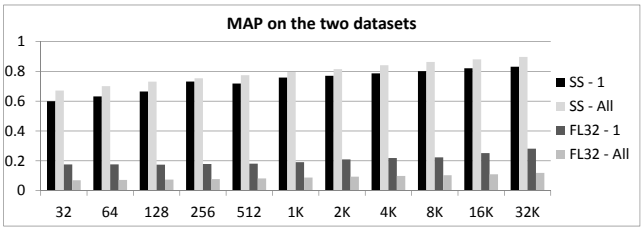

Fig. 8. Mean Average Precision computed on the two dataset (SS and FL32), using 1 or All images as training samples.

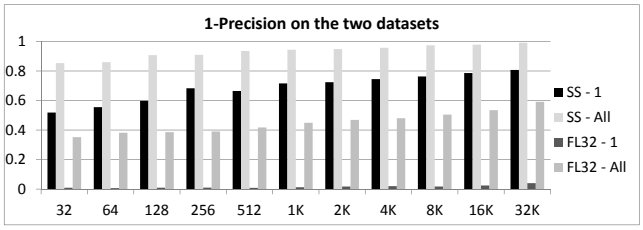

Fig. 9. 1-Precision computed on the two dataset (SS and FL32), using 1 or All images as training samples.

datasets. First we train the BoW for both datasets using 1 random image per class, and repeated the evaluation 10 times, and using as queries all other images. The average values are reported as the darker bars (noted as SS-1 and FL32-1) in Fig. 8 and Fig. 9. They show that the performance increases with the number K, but very good results are already achieved with low values of K that allow for fast computation. Second, we train the BoW for both datasets using all images for each class, using again all images as queries. The results are reported as the brighter bars (noted as SS-All and FL32-All) in Fig. 8 and Fig. 9. On the one hand, adding more images helps the search, since the training set will more easily contain images similar to the query. This is clearly visible in Fig. 9, where the brighter bars are much higher than the darker bars. On the other hand, the overall performance is better with respect to our dataset, but gets worse for the FlickrLogos32 dataset (Fig. 8). This is because adding more information in the training phase provides worse results when the images are very diverse. When images depict exactly the same location, as in our dataset, adding more examples helps in dealing the different viewpoints and lighting conditions.

VI. Conclusions

The proposed solution for location recognition has been developed for mobile application. As such, there has been the need to find efficient solutions feasible on commercial smartphones with limited resources and communication constraints. The proposed novel solution, based on ORB descriptors with BoW, resulted to be an excellent trade-off between accuracy and efficiency, as demonstrated by our tests. The exploitation of GPS localization to reduce the set of possible matches in the training images also contributed to have a highly-responsive accurate system.

References

[1] H. Pourghassem, "A hierarchical logo detection and recognition algorithm using two-stage segmentation and multiple classifiers," in *Computational Intelligence and Communication Networks (CICN), 2012 Fourth International Conference on*, 2012, pp. 227–231.

[2] M. Bagheri and Q. Gao, "Logo recognition based on a novel pairwise classification approach," in *Artificial Intelligence and Signal Processing (AISP), 2012 16th CSI International Symposium on*, 2012, pp. 316–321.

[3] S. Romberg and R. Lienhart, "Bundle min-hashing for logo recognition," in *Proceedings of the 3rd ACM International Conference on Multimedia Retrieval (ICMR)*, ser. ICMR '13. New York, NY, USA: ACM, 2013, pp. 113–120.

[4] H. Sahbi, L. Ballan, G. Serra, and A. Del Bimbo, "Context-dependent logo matching and recognition," *Image Processing, IEEE Transactions on*, vol. 22, no. 3, pp. 1018–1031, 2013.

[5] K. Grauman and T. Darrell, "Efficient image matching with distributions of local invariant features," in *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, vol. 2, 2005, pp. 627–634 vol. 2.

[6] A. Shrivastava, T. Malisiewicz, A. Gupta, and A. A. Efros, "Data-driven visual similarity for cross-domain image matching," *ACM Trans. Graph.*, vol. 30, no. 6, pp. 154:1–154:10, Dec. 2011.

[7] D. Lowe, "Object recognition from local scale-invariant features," in *Proc. IEEE Int. Conf. Comput. Vision*, vol. 2, 1999, pp. 1150–1157.

[8] K. E. A. van de Sande, T. Gevers, and C. G. M. Snoek, "Evaluating color descriptors for object and scene recognition," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 32, no. 9, pp. 1582–1596, 2010.

[9] H. Bay, A. Ess, T. Tuytelaars, and L. Van Gool, "Speeded-up robust features (surf)," *Comput. Vision Image Understanding*, vol. 110, no. 3, pp. 346–359, 2008.

[10] H. Jegou, M. Douze, C. Schmid, and P. Pérez, "Aggregating local descriptors into a compact image representation," in *Proc. IEEE Int. Conf. Comput. Vision Pattern Recognit.*, 2010, pp. 3304–3311.

[11] M. Calonder, V. Lepetit, C. Strecha, and P. Fua, "Brief: Binary robust independent elementary features," in *Proc. Eur. Conf. Comput. Vision*, 2010, pp. 778–792.

[12] E. Tola, V. Lepetit, and P. Fua, "Daisy: An efficient dense descriptor applied to wide-baseline stereo," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 32, no. 5, pp. 815–830, 2010.

[13] S. N. Sinha, J.-M. Frahm, M. Pollefeys, and Y. Genc, "Feature tracking and matching in video using programmable graphics hardware," *Mach. Vision Appl.*, vol. 22, no. 1, pp. 207–217, 2011.

[14] P. Indyk and R. Motwani, "Approximate nearest neighbors: towards removing the curse of dimensionality," in *Proc. ACM Symp. Theor. Comput.*, 1998, pp. 604–613.

[15] H. Jegou, M. Douze, and C. Schmid, "Product quantization for nearest neighbor search," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 33, no. 1, pp. 117–128, 2011.

[16] A. Oliva and A. Torralba, "Modeling the shape of the scene: A holistic representation of the spatial envelope," *Int. J. Comput. Vision*, vol. 42, no. 3, pp. 145–175, 2001.

[17] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *Proc. IEEE Int. Conf. Comput. Vision Pattern Recognit.*, 2005, pp. 886–893.

[18] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski, "Orb: An efficient alternative to sift or surf," in *Proc. IEEE Int. Conf. Comput. Vision*, 2011.

[19] R. T. Fielding and R. N. Taylor, "Principled design of the modern web architecture," *ACM Trans. Internet Technol.*, vol. 2, no. 2, pp. 115–150, May 2002.

[20] E. Rosten and T. Drummond, "Machine learning for high-speed corner detection," in *Proc. Eur. Conf. Comput. Vision*, 2006, pp. 430–443.

[21] C. Harris and M. Stephens, "A combined corner and edge detector," in *Proc. Alvey Vision Conf.*, 1988, pp. 147–151.

[22] P. L. Rosin, "Measuring corner properties," *Comput. Vision Image Understanding*, vol. 73, no. 2, pp. 291–307, 1999.

[23] L. Kaufman and P. Rousseeuw, "Clustering by means of medoids," in *Int. Conf. Stat. Data Anal.*, 1987, pp. 405–416.

[24] S. Romberg, L. G. Pueyo, R. Lienhart, and R. van Zwol, "Scalable logo recognition in real-world images," in *Proceedings of the 1st ACM International Conference on Multimedia Retrieval*, ser. ICMR '11. New York, NY, USA: ACM, 2011, pp. 25:1–25:8.