Evaluation of free libraries for visual recognition of art imagery on mobile devices

Ramiro Oliva Navas, a research project for MSc Free Software (Open University of Catalonia)

***Abstract*— This paper describes a systematic research about free software solutions and techniques for art imagery computer recognition problem. We started researching state-of-the-art landscape, conducting surveys, shortlisting candidate solutions aligned to free/open source software and ultimately we carried out speed and efficacy benchmarking of different techniques combinations on mobile devices using the popular OpenCV library. We also compare results to SIFT and SURF methods which are still reference standards in the field.**

***Index Terms*—image recognition, computer vision, mobile, benchmarking, homography, free software, OpenCV**

I. INTRODUCTION

THE study of techniques of computer image recognition has been for years one of the most exciting fields of study within artificial intelligence, both by the algorithmic challenge posed by the analysis of large volumes of unstructured information, such as the various practical applications that may involve such techniques in the field of robotics, security, automation, document management, augmented reality, etc..

The increasing computing power of information systems, thanks to increasingly powerful processors, distributed computing, mobile devices and new ultra-fast analysis techniques derived from brain research are enabling solutions and practical applications in many different scenarios like document reading and automatic indexing, biometrics, traffic signs recognition and other codes or augmented reality.

The goals of this research are:

–Knowing the current state of art techniques.

–Study the potential of most appropriate free implementations in this regard.

–Evaluate the ability of found libraries in an experiment, namely the recognition of artworks.

–List constraints encountered and future research.

II. STATE-OF-THE-ART REVIEW

A. Methods

After a careful review of the scientific literature on the subject it shows that two of the most promising techniques used in the computer image recognition are SIFT and SURF algorithms or variations of them.

We found particularly interesting a study [1] circa 2005 that examines the effectiveness of different techniques which concludes that the SIFT-based descriptors, which are based on regions, are more robust and distinctive, and therefore are best for picture comparison (this study did not cover SURF).

Regarding SURF, we found an analysis [2] of 2007 showing that it provides similar results to SIFT with the particularity of being a faster algorithm, thanks to the use of integer arithmetic.

However, another recent study [3] (2011) suggests that other techniques (SLF, PHOG, Phow, Geometric Blur and V1-like) can provide more efficiency in terms of invariance with respect to transformations, especially when dealing with photo-realistic images with complex backgrounds.

This literature review was complemented by a survey [4] conducted in May 2012 in a discussion group on the LinkedIn professional social network whose results however recommended using SIFT and SURF, possibly because, as of today, they still have a higher degree of implementation.

B. Basis of selected techniques

The goal of the techniques discussed (SIFT, SURF, and others) is to "understand" a reference or training image, by analyzing their numerical representation, detecting and extracting elements distinguishable from a geometric point of view or concerning the histogram (such as points, corners or edges which have a high degree of contrast with respect to the contour). These key elements are called features.

The image recognition process, in summary, covers the following stages:

-Feature detection. The detection algorithm (e.g. FAST or MSER) analyzes a good quality reference image under optimal lighting conditions to facilitate detection of the distinctive elements (features).

-Feature description. We then proceed to describe the outcome of the previous stage by a numerical matrix or structure similar. There are several methods of representation (eg. BRIEF, SIFT, SURF). After this phase we usually obtain a database of precomputed descriptors from a large bank of images.

-Scene recoginition. The descriptors are extracted from the picture to be analyzed (this can be a photograph taken of a scene at any angle and under different lighting conditions, possibly containing a previously trained image). At this stage and for real-time recognition the speed and quality of the

extraction of descriptors is critical, especially in realtime scenarios.

-Comparison of sets of descriptors and determination of the degree of similarity. Using probabilistic techniques we determine which matches are actually false positive hits with the aim to reduce them to a set of matching pairs that have a high correlation.

Both SIFT and SURF allow high quality descriptors extraction which possess high invariance with respect to brightness, scale, rotation or distortion.

SIFT (Scale-Invariant Feature Transform) was published [5] by David Lowe in 1999. The algorithm is patented in the United States of America; the owner of the patent [6] is the University of British Columbia.

SURF (speeded Up Robust Feature) is a robust detector of local descriptors, first presented by Herbert Bay et al. in 2006 [37]. It is partially based on SIFT but is much faster. A patent application can be found in USA [7].

C. Software libraries

To identify image recognition software libraries we conducted a wide search on the Internet, as well as a survey on the social network LinkedIn [8]. We found a significant high number of candidate libraries which implement in more or less degree image analysis algorithms that can be used in the recognition such as OpenCV, MatLab, SimpleCV, VIGRA, VXL, Matrox MIL / ONL, Cognex VisionPro, Halcon, vlfeat, IPL98, LabView, CCV (Community Core Vision) and Into.

Of these, we found more recommendations and referrals to OpenCV [9]. This library is distributed under a BSD license and can be legally used for both academic and commercial scope. It includes interfaces for a wide variety of programming languages (such as C, C++ and Python) and is available for Windows, Linux, Android and iOS platforms. The library contains over 2500 optimized algorithms and their applications range from interactive art, inspecting mining, mapping and robotics.

OpenCV includes implementations for SIFT, SURF and other algorithms like ORB and RANSAC. Since SIFT and SURF are subject to patents, we will add ORB to the selection of techniques in our experiment, along with RANSAC (traditional method), in order to get richer results and including methods which are 100% free.

For the remainder of the experiment we then will focus on OpenCV both because it's free and also due to the large number of references and recommendations, as noted above.

D. Similar experiments

Regarding the recognition of images in museums, we have found several similar experiments:

-April 2012, Project "FACES: Faces, Art, and Computerized Evaluation Systems" [10]. Three researchers at the University of California, Riverside, have launched a research project to test the use of facial recognition software to help identify these unknown subjects within the portrait, a project that ultimately can enrich European understanding of political, social and religious history.

-May 2006, "Interactive museum guide: Fast and robust recognition of museum objects". H. Bay, B. Fasel, and L. van Gool analyze the feasibility of software that implements SURF for recognizing artworks in museums using digital camera of a mobile phone, although the article does not specify what libraries have been used.

Besides these projects we have found diverse international calls that serve as a meeting point for demonstration of techniques and advances in computer image recognition solutions, such as ICCV [11] (International Conference on Computer Vision), CCVEs [12] (European Conference on Computer Vision) or CVPR [13] (Conference on Computer Vision Pattern Recognition).

In these events the outcome of open competitions are shown up describing benefits of the new techniques which have been applied to two banks of images that have been consolidated as toolsets for measuring techniques:

- "The PASCAL Visual Object Classes" [14], which aims to provide a common database of images (including people, animals, vehicles and other objects) and support tools for testing the efficacy of new image recognition algorithms . This toolset is the basis for the PASCAL Challenge annual competition held since 2005.

The winner algorithms techniques [15] in one of the forms of competition on 2011, such as NUSPSL_CTX_GPM NLPR_SS_VW_PLS, combined several algorithms, like SIFT derivatives for obtaining descriptors, a technique named "Bag of Words" (BOW) that seeks to relate these descriptors describing the image in terms of "words", eg, similarly to how a book is made up of sequences of words, and other techniques that seek to increase the success rate in recognizing catalog images.

OpenCV supports partially BOW, so this could be a line of analysis to consider if the previously selected algorithms do not offer a satisfactory level of recognition.

"CALTech101/256" [16] is another bank of images which is also used in various competitions. Any researcher can test their research progress using these images and comparing the results with other testers.

III. RESEARCH STRATEGY

Once we carried out a research about the state of the art in computer image recognition, reviewed the literature and selected associated libraries (OpenCV) and most appropiate techniques (SIFT and SURF), the next step is to prove, by experiment, the following hypothesis:

One) OpenCV is a suitable library for image recognition in art museums. Specifically: "given a photograph taken arbitraly with a digital camera on a real scene it can obtain a correspondence between the scene and some image contained in an reference art image bank."

Two) SIFT and SURF, available in OpenCV, are the most effective algorithms for this purpose. That is, "there are no

other techniques available in OpenCV with higher rate of success in recognition for the same set of photographs of real scenes."

At the time of our research, on August 2012, latest OpenCV version was 2.4.2 and was available on four platforms: Windows, Linux, iOS and Android.

We found several comparisons and similar experiments [17, 18], however none of them allows us to test the initial hypotheses regarding OpenCV and their use in the field of mobility, so we decided to target the Android port, especially given the interest aroused around mobile apps and being Android platform more widespread among smartphones [19].

A preliminary review of the Android version helped us to discover that both SIFT and SURF were not available in the Android OpenCV distribution due to patent restrictions (however, in the desktop version for Linux it is still available, but in a separate module (non-free) which must be linked manually).

This situation made us rethink the hypothesis number two (in relation to the benefits of SIFT and SURF) and we redirected the experiment to the evaluation of existing algorithms that support free solutions and mobility applications, obtaining data that allow us to compare the effectiveness (% correlation) and efficiency (time required) of the various techniques available in OpenCV for Android. Additionally we would contrast the results with the reference methods available in the desktop version (SIFT, SURF), and the latest method to the experiment date, BRISK [20]. BRISK is a relatively new method so we also wanted to know if it could provide any breakthrough improvement with respect to to the two former reference methods.

IV. EXPERIMENT DESIGN

The experiment consists of image capture and recognition, obtaining detection rates and speed as well as comparative analysis of these data using a mobile application for Android platform using version 2.4.2 of OpenCV, following systematic observation scheme .

The reference image bank consists of 10 photographs of works of art. Each of them will be projected on a monitor and we will obtain through the mobile application a sequence of frames extracted from a continuous recording, phone in hand (flyover).

Fig. 1 Some of the 194 frames extracted from the flyover over reference image #1.

The difference between the algorithms are based primarily on their degree of robustness in invariability to changes on the reference image (such as scaling, rotation, affine transformation) as well as distortions, noise and lighting conditions. To incorporate such variables to experiment, during the recording of the sequence we'll perform approach (scaling), rotation (360 °), translation (panning) and inclination (affine transformation up to about 60 or 70 °) movements in order to simulate a real usage in which a person with his/her own personal mobile device wants to identify a work of art from different angles. In addition, many of these frames have different degrees of blur or light reflections (we will apply extra lightning on the dataset #6).

Once frames are captured they will be processed from within the same mobile application using each of the different algorithms that OpenCV provides, and data will be obtained for each sequence and algorithm which will allow quantitatively comparison of the effectiveness of each technique for recognition.

A. Data collection

As seen above, the recognition process involves at first the detection of spots or areas of special interest in the image (features), extract data representing these features (descriptors) and comparison (matching) to obtain correlation between pairs of descriptors.

The number of features / descriptors and data correspondences will allow us to compare the efficacy of recognition. In addition to the descriptors matching, we proceed to calculate the homography, allowing us to locate the reference image in the scene, which is ultimately the data we are looking for.

OpenCV includes two algorithms for homography computation (RANSAC and LMEDS or "Least Median of Squares"). We will use RANSAC due to a slightly better performance in complex scenes [21]. The calculation result is a matrix H such that:

Having:

$$p_a = \begin{bmatrix} x_a \\ y_a \\ 1 \end{bmatrix}, p'_b = \begin{bmatrix} w'x_b \\ w'y_b \\ w' \end{bmatrix}, \mathbf{H}_{ab} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix}$$

Then:

$$p'_b = \mathbf{H}_{ab} p_a$$

Thus, considering a homogeneous geometric space between the two images (reference and scene), given a point p contained in the reference image, we can calculate its location in the scene (p '= Hp). Applying the matrix to the 4 corners of the reference image we can obtain the location within the scene.

For comparative data collection, the application will record continuously and in an unattended fashion the results for each combination of recognition methods on each of the frames of

the sequences of the reference images, dumping the numeric data into a text file for each combination of methods (see Annex 2 for technical specifications and combinations of methods).

Fig. 2. Computed homography using RANSAC and keypoints and descriptors obtained with MSER/FREAK. OpenCV supports methods for matrix calculation which allows the projection of any point P from the reference image into the scene.

Note that the calculated homographies can have a margin of error because RANSAC (the OpenCV method used for calculating the transform) uses a probabilistic approach. Therefore it will be necessary to perform a manual and individualized observation of the processed frames in order to validate the perimeter obtained by projecting corners of the reference image in the scene.

To do this, the application generates a copy of the frame in the device's external memory card containing the print out of the parameters used as well as the corresponding projection of the reference image.

By manual observation of each of these frame copies we will dictate whether the homography is valid and complete data matrix:

-If the homography allows identifying the object in the scene will be given a score of 1.

-If the homography to identify the object in the scene accurately fits the contour of the work of art, it will be given a score of 2.

Fig. 3. Imprecise hommography example (this will get only 1 point).

We found evidences during preliminary testing of this experiment and work of other researchers showing that the value of the determinant [22] of the transformation matrix and the value called SVD (Singular Value Decomposition) [23] can be used to establish an automatic degenerated homography filter based on minimum or maximum threshold values for these values. So, we will also collect both values, which can be obtained with OpenCV, in order to try to find later some correlation with the quality of the homography.

Finally, the data dictionary is:

Datum	Abbrev.	Type	Range
Reference image number	Img	Ordinal	[1,10]
Frame number	Frame	Ordinal	[1..X] (X=frame count)
Method combination (detector, descriptor, matcher)	Methods	Nominal	See annex 2
Number of descriptors	D	Ratio	D>=0
Number of matches	M	Ratio	M<=D
Homography score	H	Nominal	{0, 1, 2}
Number of *inliners* (valid matches, which have not been rejected after applying RANSAC)	I	Ratio	I<=M
Time required	T	Ratio	>0
Determinant value	DT	Ratio	>0
Singular value decomposition ratio	SVD	Ratio	>0

B. Statistical analysis of results.

For statistical analysis of results we followed the methodology proposed by Demšar [24] to compare a set of classifiers over multiple datasets. The first goal of our analysis was to determine whether the scores of each classifier on each dataset differ significantly from scores of other classifiers (which would show us that there are different degrees of effectiveness or efficiency in the set of classifiers). And, we also wanted to determine from the results which algorithms were significantly different from each other (and to conclude with a ranking of efficiency or speed).

The first step is, therefore, to determine whether the null hypothesis, which states that all algorithms behave similarly, may be rejected. Demšar proposes the Friedman test which compares average ranks of methods, $= \frac{1}{N}\sum_i R_i^j$. Under the null hypothesis, all methods are equivalents, so their average ranks Rj are equal, the Friedman statistic

$$\chi_F^2 = \frac{12N}{k(k+1)}\left[\sum_j R_j^2 - \frac{k(k+1)^2}{4}\right] \qquad \text{(D.1)}$$

is distributed according to χ_F^2 with $k-1$ degrees of freedom, when N and k are big enough (as a rule of a thumb, $N > 10$ and $k > 5$). For a smaller number of algorithms and data sets, exact critical values have been computed [27-28].

Iman and Davenport [29] showed that Friedman's χ_F^2 is undesirably conservative and derived a better statistic

$$F_F = \frac{(N-1)\chi_F^2}{N(k-1)-\chi_F^2} \qquad \text{(D.2)}$$

which is distributed according to the F-distribution with k−1 and $(k-1)(N-1)$ degrees of freedom. The nul hypothesis can be rejected only if F_F is less than the critical value of F-distribution for a confident threshold of α. The critical values can be found on any statistical book or online [26].

If the null hypothesis is rejected, we can continue with a post-hoc test. The Nemenyi test [30] is used when all classifiers are compared to each other. The performance of two classifiers is significantly different if the corresponding average differs by at least the critical distance

$$CD = q_\alpha \sqrt{\frac{k(k+1)}{6N}} \qquad \text{(D.3)}$$

where critical values q_α are based on the Studientized range statistic divided by $\sqrt{2}$ (see table D.4).

#classifiers	2	3	4	5	6	7	8	9	10
$q_{0.05}$	1.960	2.343	2.569	2.728	2.850	2.949	3.031	3.102	3.164
$q_{0.10}$	1.645	2.052	2.291	2.459	2.589	2.693	2.780	2.855	2.920

Table D.4 Critical values for the two-tailed Nemenyi test

V. RESULTS

A. Detection test

This test examines the ability of each combination of techniques to obtain a non-degenerated homography (eg. excludes deltoides) even if the projected reference image does not match the scene perfectly.

Of 30 combinations of techniques available in OpenCV for Android and analyzed in our experiment (see Appendix A), we have selected those that have more occurrences among the top 10 of each resulting dataset. Two matchers were used, BRUTEFORCE-HAMMING2 - BF(H2) from now on - and FLANN. In all cases where FLANN was involved, except for FAST/BRIEF, it produced worst recognition rates and the overall speed was inferior than BF (H2) so we decided to consider only the matcher BF (H2) for the rest of detectors.

We found that some combinations, such as STAR/FREAK, throwed in some serious errors in some datasets (inability to recognize any frame or extracting a very low number of features from the reference image). These combinations have been excluded from comparative analysis.

Table 1 shows the percentage of success in the detection of the reference image for each dataset (each one containing 150-250 frames corresponding to the recorded flyover) and also the avg. rank for the selected methods, having into account the above selection criteria and their availability in the OpenCV Android port.

Following the methodology suggested by Demšar introduced in the previous section we used the Friedman test to check the level of significance of these results and showed the critical distance with the avg. ranks graphically, which allows quick visualization of groups of algorithms with similar performance.

The Friedman test verifies that the recorded average ranks from the 9 algorithms are significantly different than the mean rank which would be 4.5. The null hypothesis states that all algorithms behave similarly. It's what we have to verify. Applying the formulas [D1] and [D2], we have χ_F^2 = 56.28 y F_F = 21.35. With 9 methods and 10 datasets, Friedmann statistic is distributed according F-distribution with 9-1 = 8 y (9-1)x(10-1) = 72 degrees of freedom. The critical value of F(8,72) for a confidence of 5% ($p<0.05$) is 2.07 which is less than 21.34, then we can reject the null hypothesis. To determine whether there is a significant difference in efficiency between each of the algorithms, otherwise if there are groups of algorithms with similar performance, we used the Nemenyi [24] post-hoc test. This test compares all algorithms between them, allowing us to determine whether there is a significant difference between families or groups of algorithms.

The critical distance for p = 0.10, applying [D.3] is 3.50. Figure 4 shows the critical distance (3.50) together with the average rank of each of the methods evaluated.

The algorithms whose mean difference of rankings is below the critical distance are displayed in groups, united by a thick line, indicating that the results are not significant enough to conclude that individual methods in those groups are more effective than others of the same group. Particularly, this post-hoc analysis indicates that, although the results on the selected datasets shows us that GFTT/FREAK offers the best average detection rate, these are not significant enough to conclude that in all cases this method exceeds ORB, MSER/FREAK, FAST/FREAK or GFTT/BRIEF although we can conclude that it can detect images better than ORB/BRIEF, FAST/BRIEF and MSER/BRIEF.

DataSet	MSER/ FREAK	ORB	GFTT/ FREAK	FAST/ FREAK	ORB/ BRIEF	GFTT/ BRIEF	FAST/ BRIEF/ BF(H2)	FAST/ BRIEF/ FLANN	MSER/ BRIEF
#1	99.48 (1)	93.30 (2)	65.46 (3)	61.86 (4)	54.12 (5)	52.58 (6)	51.55 (7)	26.29 (8)	45.88 (8)
#2	83.60 (4)	91.60 (1)	87.20 (2)	86.40 (3)	67.20 (7)	68.80 (5.5)	68.80 (5.5)	58.80 (9)	60.80 (8)
#3	70.27 (3)	90.81 (1)	74.59 (2)	69.73 (4)	59.46 (5.5)	59.46 (5.5)	56.22 (7)	43.24 (9)	49.19 (8)
#4	56.71 (2.5)	56.71 (2.5)	59.76 (1)	48.78 (4)	42.07 (6)	42.68 (5)	37.80 (7)	20.12 (9)	37.20 (8)
#5	48.47 (5)	51.53 (4)	55.10 (1)	54.08 (3)	29.08 (8)	54.27 (2)	37.24 (6)	27.55 (9)	36.73 (7)
#6	1.78 (9)	17.79 (8)	56.23 (1)	52.31 (2)	29.54 (5)	35.94 (3)	30.25 (4)	21.00 (7)	22.78 (6)
#7	79.12 (1)	59.34 (4)	63.74 (2)	59.89 (3)	39.01 (8)	45.05 (5)	41.76 (7)	28.57 (9)	42.31 (6)
#8	81.48 (1)	75.13 (2)	67.72 (4)	71.43 (3)	51.32 (5.5)	51.32 (5.5)	49.74 (7)	36.51 (9)	48.68 (8)
#9	73.63 (2)	79.12 (1)	67.58 (3)	67.03 (4)	56.04 (6)	61.54 (5)	54.40 (7)	48.35 (9)	51.65 (8)
#10	88.36 (1)	84.13 (2)	74.60 (3)	73.54 (4)	55.03 (6)	68.78 (5)	54.50 (7)	42.33 (9)	50.26 (8)
Avg. Rank	2.95	2.75	2.2	3.4	6.2	4.75	6.45	8.7	7.5
Winners	3	2	1						

Table 1 Average rank of methods used in detection test

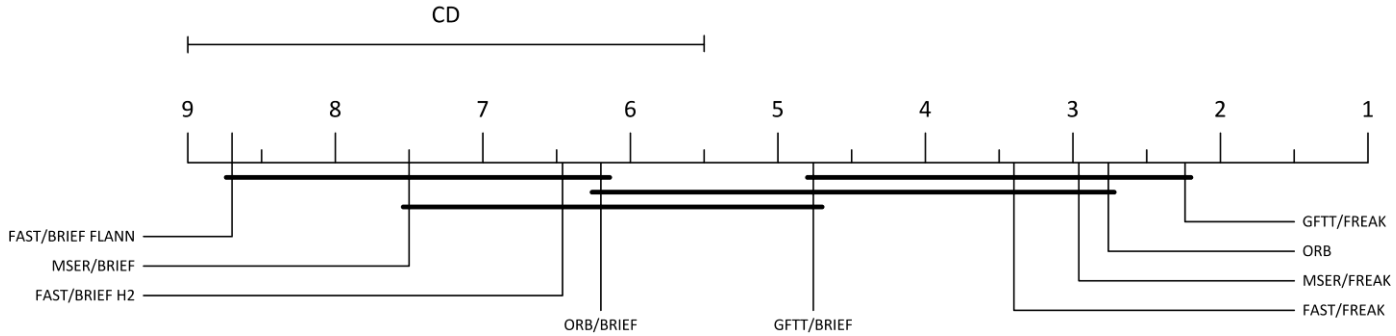


Figure 4 Avg. rank comparison in detection test. Groups of methods with similar performance are shown linked

Additionally, even though SIFT, SURF and BRISK are not yet available in the version of OpenCV for smartphones, we thought it would be interesting to compare the results of the techniques available in OpenCV for Android with these reference methods found in the state of the art but that are only available in the desktop version.

Although the hardware where we ran the experiment is different, we wanted to compare the effectiveness of the algorithm itself, not its speed. The table 2 shows the results for SIFT, SURF, BRISK and GFTT/FREAK (the best representative of the group of algorithms with better detection rates in previous test).

Again we apply here the Friedman test. We have $\chi_F^2 = 21.06$ and $F_F = 21.20$. With 4 techniques and 10 datasets, Friedman function is distributed as the F distribution with 4-1 = 3 and (4-1) x (10-1) = 27 degrees of freedom. The critical value of F (3.27) for $p < 0.05 = 2.96 < 21.20$, then we reject the null hypothesis. The Nemenyi post-hoc test showed us that, for a critical distance of 1.32, the methods can be classified in three groups according to the significancy of the results (see figure 5). SIFT belongs to the most effective group while GFTT/FREAK to the worst one. The data is not significant enough to conclude in which group are neither SURF nor BRISK. Nevertheless they resulted in second and and third positions respectively against our datasets.

We can see that from all available OpenCV methods, SIFT wins in 8 of the 10 datasets, followed by SURF and BRISK:

DataSet	SIFT	SURF	BRISK	GFTT/FREAK
#1	100 (2)	100 (2)	100 (2)	65.46 (4)
#2	99.60 (1)	97.20 (2.5)	97.20 (2.5)	87.20 (4)
#3	97.30 (1)	92.97 (3)	94.59 (2)	74.59 (4)
#4	95.12 (1)	81.71 (2)	53.05 (4)	59.76 (3)
#5	92.86 (1)	75.00 (3)	79.59 (2)	55.10 (4)
#6	61.57 (2)	62.28 (1)	45.20 (4)	56.23 (3)
#7	89.01 (1)	86.81 (2)	75.27 (3)	63.74 (4)
#8	92.06 (1)	90.48 (2)	85.19 (3)	67.72 (4)
#9	92.31 (1)	86.81 (3)	87.91 (2)	67.58 (4)
#10	98.94 (1)	89.42 (2)	86.77 (3)	74.60 (4)
Avg. Rank	1.2	2.25	2.75	3.8

Table 2 Avg. detection rank comparison including desktop methods

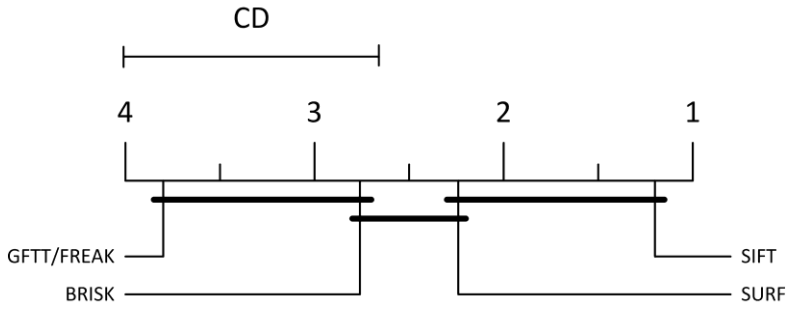


Fig 5. Comparisson of available mobile vs non-mobile methods in OpenCV

B. Location Test

This test is useful to determine which algorithms allow precise positioning of the work in the scene. To verify this condition, we had to check visually that each homography fitted perfectly the image in the scene at a pixel-perfect level. We followed the same criteria for method selection used in the detection test. Of 30 techniques combinations, we chose the 10 algorithms with most occurency within the top 10 positions in each dataset. Table 3 shows the results. We see that, of the methods available in OpenCV for Android, GFTT/FREAK has won the best score (rank 1.8), followed by MSER/FREAK (2.5) and FAST/FREAK (2.7). ORB, which resulted second in the detection test resulted is fifth place this time (rank 4.8), probably due to default parameter used for limiting the detection to 500 features. Increasing this parameter could lead to greater accuracy of generated homographies but at the cost of speed. It would be interesting to confirm this with a recalculation of the critical distance with Bonferroni-Dunn test [24].

DataSet	MSER/ FREAK	ORB	GFTT/ FREAK	FAST/ FREAK	MSER/ BRIEF	GFTT/ BRIEF	FAST/ BRIEF/ BF(H2)	FAST/ BRIEF/ FLANN	GFTT/ ORB
#1	94.85 (1)	77.84 (2)	60.82 (3)	57.22 (4)	29.38 (8)	40.21 (5)	38.14 (6)	13.40 (9)	33.51 (7)
#2	77.60 (3)	70.00 (4)	83.60 (1)	81.60 (2)	47.20 (8)	58.80 (5)	52.40 (7)	38.80 (9)	55.20 (6)
#3	57.30 (2)	45.95 (5)	71.89 (1)	54.59 (3)	36.76 (7)	48.65 (4)	32.43 (8)	18.38 (9)	43.78 (6)
#4	33.54 (3)	27.44 (6)	53.05 (1)	42.68 (2)	19.51 (8)	29.88 (4)	25.61 (7)	13.41 (9)	29.27 (5)
#5	40.31 (4)	44.39 (3)	50.51 (1)	50.00 (2)	30.61 (5.5)	30.10 (7)	30.61 (5.5)	21.94 (9)	26.53 (8)
#6	1.07 (8)	0.38 (9)	51.60 (1)	43.06 (2)	11.03 (7)	24.56 (3)	24.20 (4)	14.95 (6)	22.06 (5)
#7	71.98 (1)	48.35 (4)	62.09 (2)	57.14 (3)	36.26 (7)	40.66 (5)	37.36 (6)	23.08 (9)	35.71 (8)
#8	77.25 (1)	68.78 (2)	62.96 (4)	66.67 (3)	43.92 (5.5)	43.92 (5.5)	42.33 (7)	25.93 (9)	38.10 (8)
#9	64.84 (1)	43.96 (6)	63.74 (2)	59.34 (3)	37.36 (7)	46.15 (4)	36.26 (8)	26.37 (9)	44.51 (5)
#10	77.78 (1)	41.80 (7)	70.37 (2)	68.78 (3)	38.10 (9)	44.97 (5)	40.21 (8)	29.10 (10)	42.33 (6)
Avg. Rank	2.5	4.8	1.8	2.7	7.2	4.75	6.65	8.8	6.4
Winners	2		1	3					

Table 3. Average rank of methods used in location test

We proceeded with the significancy test. Having $\chi_F^2 = 69.13$ y $F_F = 57.24$, 9 methods and 10 datasets, Friedman statistic is distributed according to the F-distribution with 9-1 = 8 and (9-1) x (10-1) = 72 degrees of freedom. The critical value of F(8,72) for $p<0.05$ is $2.07 < 57.24$, then we reject the null hypothesis.

In terms of individual performance the Nemenyi post-hoc test tells us that we can classify the 9 algorithms into 3 groups (CD = 3.50, see Figure 6). The most precise group of methods includes GFTT/FREAK, MSER/FREAK, FAST/FREAK, GFTT/BRIEF and ORB, same results as in the detection test. Our observation showed up that the descriptor FREAK was rotation-invariant and provided superior recognition rates and higher quality homographies than BRIEF when combined with FAST, MSER and GFTT detectors.

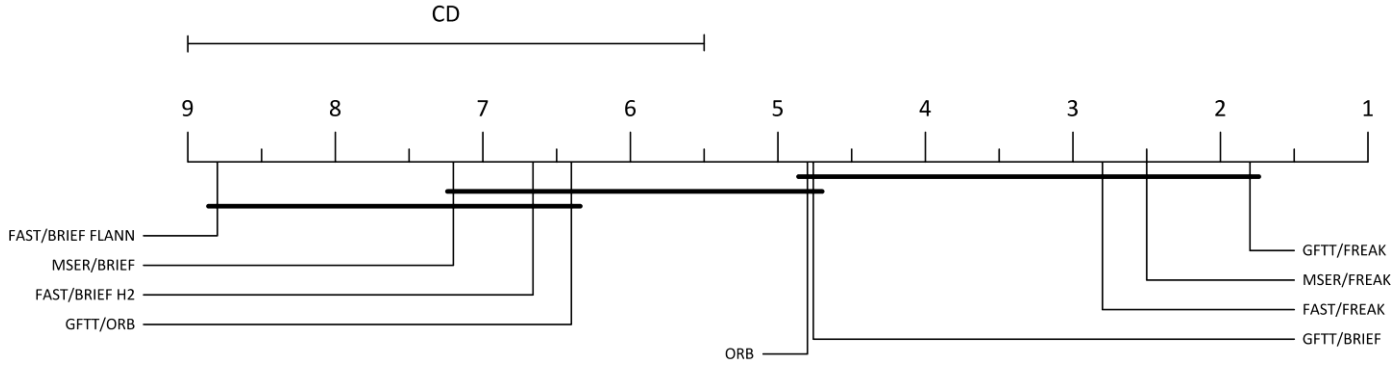


Figure 6. Avg. rank of precise location capabilities comparisson against the critical distance.

We compared again GFTT/FREAK with reference methods, SIFT, SURF and BRISK (available only in Desktop version of OpenCV). We used BF matcher with SIFT and SURF and BF(H2) in the case of BRISK and remaining methods in mobility. This time we also included the average percentage of frames in which the reference image has been successfully located in the scene. Table 4 shows the results.

Before concluding, we conduct the significancy tests. Having χ_F^2 = 23.16 and F_F = 30.47, 4 techniques and 10 datasets, Friedman statistic is distributed according to F-distribution with 4-1 = 3 and (4-1) x (10-1) = 27 degrees of freedom. The critical value of F (3, 27) for p <0.05 = 2.96 < 30.47 then we reject the null hypothesis.

The Nemenyi post-hoc test indicates that SIFT is definitely the most accurate method available and that we have two groups of algorithms in second and third place, based on the separation of their avg. rank respect the critical distance (1.32). In these results, BRISK offers similar accuracy to SURF or GFTT/FREAK. Figure 7 shows graphically the differences between algorithms with a confidence level of 10% (CD = 1.32).

Interestingly enough, GFTT/FREAK had a behavior similar to SIFT in dataset 6, characterized by a bright light which added extra illumination to the scene. Also, it will be necessary to increase the number of images to determine if there's really a significant difference regarding SURF, BRISK and GFTT/FREAK methods.

C. Speed test

In this third test we only considered the methods analyzed in the precise location test, with the aim of comparing the detection capabilities and the speed efficiency of the same algorithms. Table 5 shows for each of the combinations chosen and for each dataset, the average time (in seconds) of homography computation, including the calculation of the determinant and the SVD (singular value decomposition), values which will be used later to automatically detect and discard degenerated homographies.

Besides rejecting the null hypothesis (χ_F^2 = 58.14 and F_F = 20.23), the result of the post-hoc test (p = 0.10, critical distance 3.50) is shown graphically in Figure 8. We see that, in view of the selected datasets, FAST/BRIEF (both H2 and with FLANN matchers) is faster, followed by ORB and GFTT/BRIEF. These three techniques would be faster, although it should be noted that FAST/BRIEF with FLANN matcher resulted in the worst rank in our previous precise location rate. Note how FAST/FREAK won in dataset #6 which had a bad illumination condition.

DataSet	SIFT	SURF	BRISK	GFTT/FREAK
#1	100 (1)	98.97 (2)	97.94 (3)	60.82 (4)
#2	96.80 (1)	92.00 (3)	94.00 (2)	83.60 (4)
#3	89.19 (1)	83.78 (2)	65.95 (4)	71.89 (3)
#4	85.98 (1)	76.22 (2)	44.51 (4)	53.05 (3)
#5	87.24 (1)	64.80 (3)	70.92 (2)	50.51 (4)
#6	53.38 (1)	39.86 (3)	33.10 (4)	51.60 (2)
#7	87.91 (1)	85.16 (2)	72.53 (3)	62.09 (4)
#8	92.06 (1)	84.66 (2)	79.89 (3)	62.96 (4)
#9	81.87 (1)	76.37 (2)	70.33 (3)	63.74 (4)
#10	96.30 (1)	81.48 (2)	77.25 (3)	70.37 (4)
Avg.	87%	78%	71%	63%
Avg. Rank	1	2.3	3.1	3.6

Table 4. Efficacy comparison between mobile and desktop reference methods

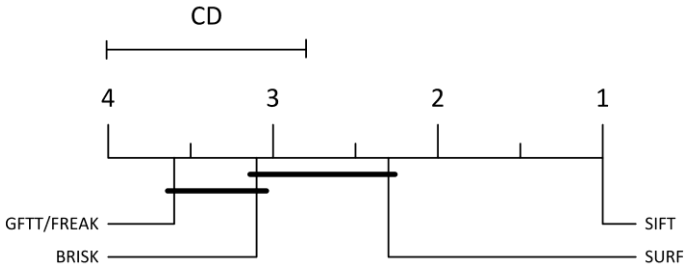


Figure 7. Avg. rank of precise location capabilities comparisson against the critical distance.

DataSet	MSER/ FREAK	ORB	GFTT/ FREAK	FAST/ FREAK	MSER/ BRIEF	GFTT/ BRIEF	FAST/ BRIEF/ BF(H2)	FAST/ BRIEF/ FLANN	GFTT/ ORB
#1	1.83 (8)	0.62 (2)	2.18 (9)	1.31 (5)	1.60 (7)	1.13 (4)	0.61 (1)	0.81 (3)	1.37 (6)
#2	1.50 (8)	0.59 (3)	1.91 (9)	0.97 (6)	1.03 (7)	0.94 (5)	0.12 (1)	0.48 (2)	0.84 (4)
#3	1.35 (8)	0.46 (2)	1.58 (9)	0.99 (7)	0.98 (6)	0.86 (4)	0.36 (1)	0.48 (3)	0.88 (5)
#4	0.83 (4)	0.78 (3)	1.18 (8)	1.45 (9)	1.10 (7)	0.94 (5)	0.64 (1)	0.72 (2)	0.95 (6)
#5	1.56 (8)	0.84 (3)	2.07 (9)	1.19 (5)	1.29 (7)	1.11 (4)	0.58 (1)	0.64 (2)	1.24 (6)
#6	0.61 (4)	0.85 (5.5)	0.92 (8)	0.48 (1)	0.95 (9)	0.85 (5.5)	0.52 (2)	0.6 (3)	0.87 (7)
#7	1.84 (7)	0.92 (2)	2.15 (9)	2.07 (8)	1.45 (6)	1.14 (4)	0.98 (3)	0.90 (1)	1.28 (5)
#8	0.98 (6)	0.62 (1)	1.07 (7)	1.35 (9)	1.2 (8)	0.89 (2.5)	0.95 (4.5)	0.89 (2.5)	0.95 (4.5)
#9	0.87 (5)	0.6 (4)	1.16 (9)	0.49 (2)	1.06 (8)	0.9 (6)	0.44 (1)	0.56 (3)	0.93 (7)
#10	0.87 (5)	.52 (1)	1.19 (9)	0.77 (4)	1.12 (8)	0.97 (6)	0.63 (2)	0.74 (3)	0.99 (7)
Avg. Rank	6.3	2.65	8.6	5.6	7.3	4.6	1.75	2.45	5.75
Winners		3					1	2	

Table 5 Comparativa de velocidad de los métodos disponibles en movilidad

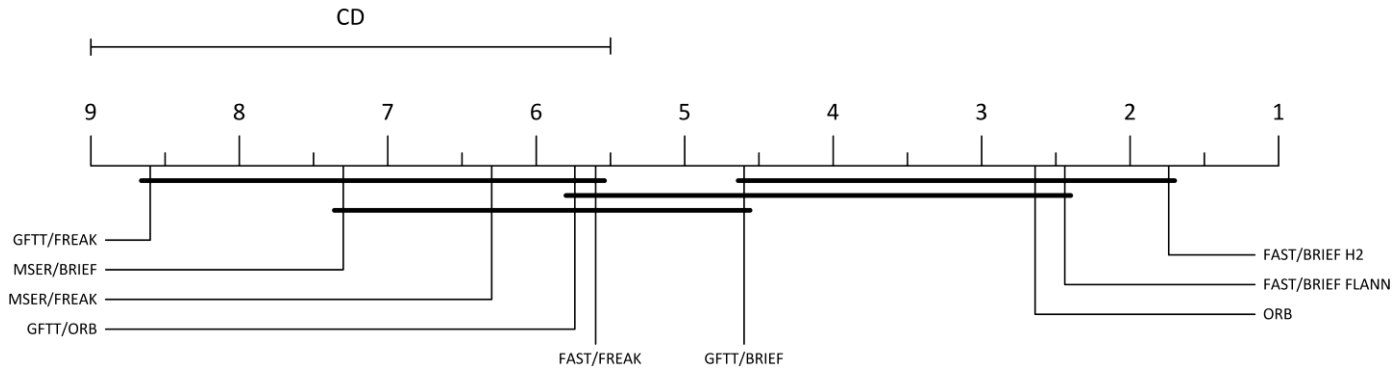


Figure 8. Avg. rank of speed test comparisson against the critical distance

As previously, we also recorded the speeds for desktop methods, in order to conduct a particular comparative against SIFT, SURF and BRISK. Since the desktop runtime environment for these methods is obviously different we have not considered mixing results in the comparative analysis.

Table 6 shows the average time (in seconds per frame) for each dataset. The Friedman test (χ_F^2 = 15.45 y F_F = 30.56) exceeds the critical value for F(2,18) = 3.56 < 30.56 so we reject the null-hypothesis

The Nemenyi post-hoc test (shown graphically in Figure 9, CD = 0.92) shows that SIFT and BRISK, with a similar performance, are in fact faster than SURF. In our datasets, surprisingly BRISK was even faster than SIFT in 6 of the 10 datasets, tied in the fourth dataset while SIFT won the remaining 3 datasets.

DataSet	SIFT	SURF	BRISK
#1	0.22 (2)	0.88 (3)	0.15 (1)
#2	0.17 (2)	0.53 (3)	0.10 (1)
#3	0.17 (2)	0.47 (3)	0.07 (1)
#4	0.19 (1.5)	0.74 (3)	0.19 (1.5)
#5	0.18 (2)	0.60 (3)	0.15 (1)
#6	0.21 (2)	0.5 (3)	0.17 (1)
#7	0.25 (1)	1.02 (3)	0.30 (2)
#8	0.23 (1)	0.94 (3)	0.43 (2)
#9	0.2 (2)	0.65 (3)	0.17 (1)
#10	0.16 (1)	0.81 (3)	0.27 (2)
Avg. Rank	1.65	3	1.35

Table 6. Speed rank comparisson of Desktop methods

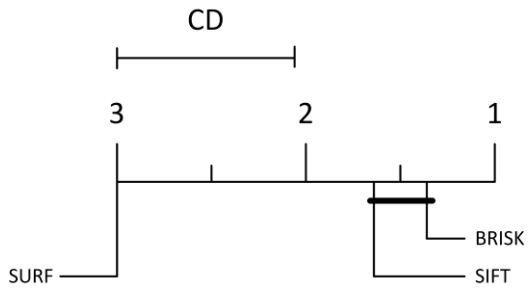
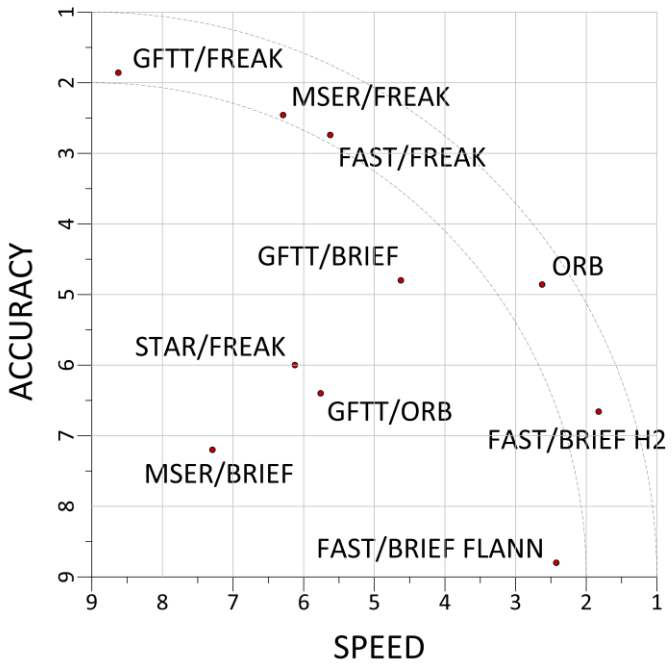


Figure 9. Avg. speed rank comparison (p=0.10) for reference methods.

D.Mixed chart

As a final comparative we plotted the results of the speed and location tests in the following 2D chart where we can see the goodness of the algorithms having into account both dimensions (accuracy and speed):



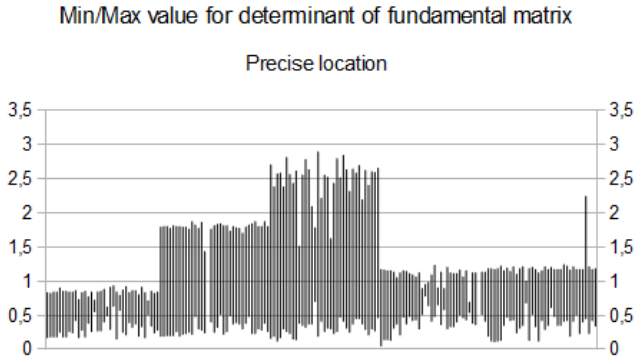
The algorithms closer to the bisector are those with a better quality/efficiency balance (MSER/BRIEF, STAR/FREAK, GFTT/ORB, GFTT/BRIEF and ORB), while those who are positioned between the two outer arcs stand out for their speed, precision or a better balanced behavior than other combinations.

Among this recommended group of techniques we found GFTT/FREAK, MSER/FREAK, FAST/FREAK, ORB and FAST/BRIEF w/BF-(H2) resulted the most recommended techniques OpenCV for Android can provide according to our tests.

Discarding wrong homographies

Following analysis of the methods provided by OpenCV, we wanted to find indicators that allow us to automatically discard homographies which could lead to false detections or imprecise locations (what is useful in augmented reality applications for example).

Our experiment recorded for each frame the value of the determinant, a value we found in the bibliography review that can be be used for this purpose. We showed this value for the frames of the flyovers of in the graph below:



In those frames in which the determinant was outside the minimum and maximum values shown in the graph, the homography presented problems or was degenerated and had to be discarded. So, if we seek greater precision, as a general rule of thumb we can discard homographies in cases in which the determinant is outside the range [0.2, 2.5].

We also observed that the homographies tended to degenerate as the ratio between the first and last singular value of SVD grows above a certain value. In our tests, it was very rare to find valid homographies with SVD value above 600,000. Above this value (even 500K for safer zone), we can safely discard homographies.

VI. Conclusion

The experiment has revealed that OpenCV effectively provides a sufficient variety of techniques in both desktop and Android versions (using a midrange device) that can be used effectively for recognizing images of works of art. Furthermore, regarding mobility methods and according to our findings, we conclude:

-The combination of methods GFTT (detector) FREAK (descriptor) and BF (H2) (matcher) provides in most cases the best recognition accuracy, accurately locating the artwork in the scene; however the combined speed is the slowest of all methods tested.

-The combination of FAST (detector), BRIEF (descriptor) and BF (H2) (matcher) is the fastest one. BRIEF can be applied only when the processed image is oriented vertically, since it's not rotation invariant. If speed is an important factor, an efficient solution could make use of mobile gyroscope to automatically rotate the captured image and thus allow the use of this faster combination.

-ORB is an option that combines a high speed (similar to FAST+BRIEF) and a high detection rate. The ORB implementation in OpenCV defaults the limit of the number of descriptors to 500, but you can specify a larger number, thereby increasing the precission of the homography. However, the processing time is also increased.

-FAST/FREAK produced excellent results in dataset # 6 (ranked second in location test and first in speed). This dataset presented a extreme high light condition that made very difficult the extraction of features, a capability in which FAST is remarkable.

-MSER/FREAK had similar performance to GFTT/FREAK, being somewhat less accurate in some cases but faster.

-Matcher FLANN produced lower recognition rates than BRUTEFORCE-HAMMING2 in all registered cases and only in a few isolated cases involving a large number of keypoints (obtained by FAST) it resulted faster.

Overall, these five mobility methods have produced very satisfactory results, considering that the device used for the experiment was not a first-class. We used default parameters for all methods, except for FAST for which we set the threshold to 50 (default value was 30). Another determining factor to have into account is the quality of the device builtin camera. Image must be focused properly at all times as well.

Regarding the reference methods in the industry, we found that SIFT continues to offer better rates of image recognition. However, we have found that BRISK significantly exceeds in speed to both SURF and SIFT, which is remarkable considering that BRISK also provides a similar recognition rate compared to SURF. It will be particularly interesting to analyze the performance of BRISK in mobility against GFTT/FREAK, MSER/FREAK, FAST/FREAK, ORB and FAST/BRIEF, once it's included in the mobile version of OpenCV. It is likely that, according to the results of our experiment, BRISK will become soon the reference free method in many future projects in mobility.

The findings of our research were put into practice with the development of a complementary mobile application which we describe in Appendix B. This app allows you to try the recommended methods (FAST/BRIEF, ORB, GFTT/FREAK as well as FAST/FREAK, MSER/FREAK) in two modes: exploration mode (real-time capture and recognition) and frozen/static image recognition.

APPENDIX A

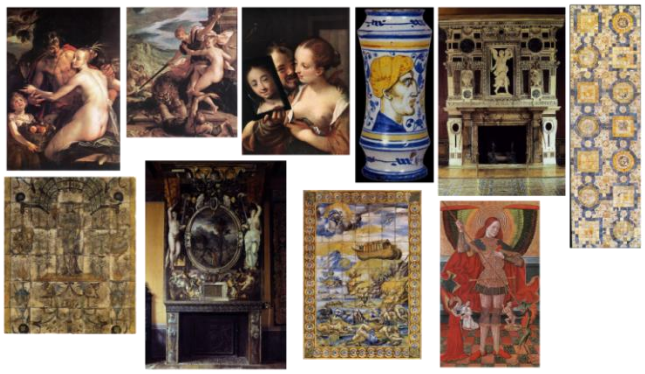
The mobile device used in the experiment was a LG Optimus SOL E730 smartphone, with a 5 MPX camera and operating system Android 2.3.4 (Gingerbread).

Number of images used: 10

Total flyover keyframes: 2.012
Method combinations used: 36
Total number of homographies checked: 61.348

For data logging and frame analyzing and recording a custom application was developed in Java and C++ for OpenCV interfacing through JNI on a Linux system. We downloaded the source code for BRISK and recompiled OpenCV for the desktop experiment. At the time BRISK code was OpenCV-ready but was not part of any OpenCV distribution yet.

The images were stored by the application in PNG format on the external micro-SD memory card after converting them to grayscale and rescaling to the device screen resolution (768x432 pixels, landscape mode).



Individual images can be viewed online at Web Gallery of Art (http://www.wga.hu):

"Allegory", AACHEN, Hans von,
http://www.wga.hu/art/a/aachen/allegory.jpg
"Bacchus, Ceres and Cupid", AACHEN, Hans von,
http://www.wga.hu/art/a/aachen/bacchus.jpg
"Joking Couple", AACHEN, Hans von,
http://www.wga.hu/art/a/aachen/j_couple.jpg
"The Archangel Michael", ABADIA, Juan de la,
http://www.wga.hu/art/a/abadia/michael.jpg
"Albarello", ABAQUESNE, Masséot,
http://www.wga.hu/art/a/abaquesn/albarell.jpg
"Ceramic Floor", ABAQUESNE, Masséot,
http://www.wga.hu/art/a/abaquesn/floor1.jpg
"Ceramic Floor", ABAQUESNE, Masséot,
http://www.wga.hu/art/a/abaquesn/floor2.jpg
"The Flood", ABAQUESNE, Masséot,
http://www.wga.hu/art/a/abaquesn/theflood.jpg
"Chimney breast", ABBATE, Niccolò dell',
http://www.wga.hu/art/a/abbate/chimney1.jpg
"Chimney breast", ABBATE, Niccolò dell' (2)
http://www.wga.hu/art/a/abbate/chimney2.jpg

The complete catalog can be downloaded from:
http://www.wga.hu/frames-e.html?/database/download/index.html

List of methods involved in our experiment:

DETECTORS	DESCRIPTORS	MATCHERS
FAST MSER GFTT STAR ORB	BRIEF ORB FREAK	FLANN BF(H2)
SIFT * SURF *		FLANN BF
BRISK*		FLANN BF(H2)

*SIFT, SURF and BRISK were only available in desktop version.

-Feature from Accelerated Segment Test (FAST) is a very fast corner detector algorithm which considers a circle of 16 pixels around the corner candidate [31]. The pixels are classified into dark, similar, and brighter subsets and the ID3 algorithm from [34] is used to select the pixels which yield the most information about whether the candidate pixel is a corner.
-Maximally Stable Extremal Regions (MSER) belongs to the blob detectors family. It extracts from an image a number of regions, called MSERs, which are connected components of some level sets of the image [32].
-Good Features to Track (GFTT) is designed to detect cornerness patterns in an image as described in [33]. The algorithm seeks to to detect strong corners and to facilitiate individual object tracking on different frames in a videostream.
-The STAR detector is derived from the CenSurE (Center Surround Extrema) detector [35]. Instead of a circle, the STAR feature detector uses a center-surrounded bi-level filter of two rotated squares.
-Oriented FAST and Rotated BRIEF (ORB) builds on FAST keypoint detector and BRIEF descriptor, which have demonstrated a good performance, adding centroid technique to provide orientation to the corners and using this operator to address the in-plane rotation limitation of BRIEF [36].
-Scale Invariant Feature Transform (SIFT) method provides features that are translation, rotation and scale invariant, using descriptors based upon gradient histograms with contextual information [5].
-Speeded Up Robust Features (SURF) is a scale and rotation nvariant interest point detector/descriptor which relies on integral images for image convolutions to reduce computation time [37].
-Binary Robust Invariant Scalable Keypoints (BRISK) is a very fast detector/descriptor based on the application of a novel scale-space FAST-based detector in combination with the assembly of a bit-string descriptor from intensity comparisons retrieved by dedicated sampling of each keypoint neighborhood [20].
-Fast Keypoint Retina (FREAK) is a highly reliable descriptor which uses a cascade of binary strings computed by efficiently comparing image intensities over a retinal sampling pattern [39].
-Fast Approximate Nearest Neighbors with Automatic Algorithm Configuration (FLANN) achieves high performance in approximate nearest neighbor matching in high-dimensional problems automatically choosing between k-means or k-d trees based upon a sample dataset and the importance of minimizing memory or build time rather than just search time [38]
-BruteForce (BF) descriptor matcher finds the closest descriptor (using Euclidean metric) in the second set to the first one set by trying each one. BF(H2) uses Hamming distance.

APPENDIX B

In this appendix we describe a sample real application for detecting art imagery on Android mobile devices which corroborates the results obtained in our research.

Using this application the user can check the performance of the OpenCV methods recommended by us: FAST/BRIEF, ORB, GFTT/FREAK as well as FAST/FREAK and MSER/FREAK.

At the start, the application allows a basic image catalog management, for which algorithms will be trained extracting the descriptors required for subsequent exploration phase.

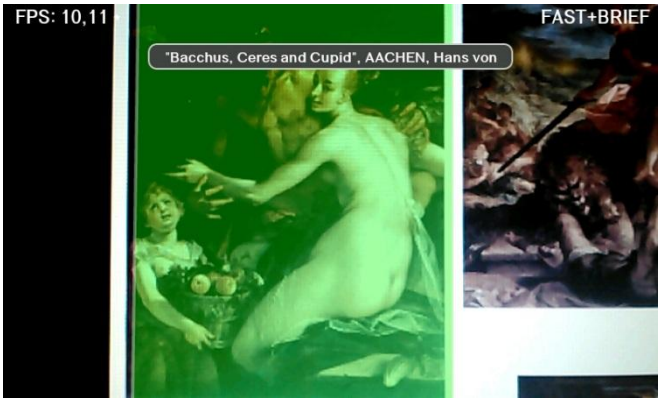
The first time the application is executed it automatically adds 10 sample images that have been used for the experiment of this research and which are listed in Appendix A.

It is possible though to photograph small objects and add them to the catalog. We recommend placing the object over a sheet of white paper. Thus, the application automatically crops the image removing the extra white space.

In the exploration mode, the application takes the captured frames by the camera and analyzes them in real-time using different threads so the UI doesn't get blocked (it got 10-12 frames per second while running the recognition in the background thread, which depending on the case happened at intervals from less than 1 second to a few seconds between each homography computation and visualization).

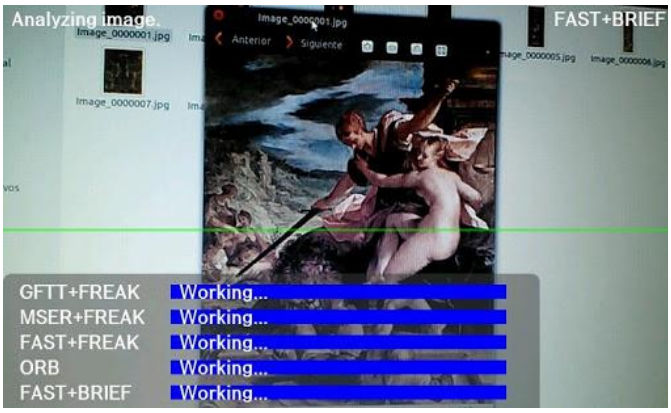
The default method selected is FAST/BRIEF which is the fastest one, but clicking on the menu button on your Android

device allows you to select among the 5 methods. In this mode we can fly a camera phone over the trained images, projected on a monitor or printed on paper, and check the performance of each algorithm in a real time scenario. When the application detects an image it fills the perimeter of a quadrilateral homography in green and displays the name of the artwork and its artist.



The frozen mode takes a snapshot of the scene and analyzes it sequentially using the 5 methods. The application then registers a ranking of success, allocating one point to each method that successfully detects any trained image.

We can see how FAST/BRIEF fails to detect images if we rotated the camera. On the other hand we can clearly observe the difference in speed between GFTT/FREAK (slowest) and ORB or FAST/BRIEF for example.



The recognition process in real time have to conduct a match between the current frame descriptors and the descriptors for each of the reference images in the catalog. Our application calculates the homography for methods that have found 10 or more matches (we also implemented a cross-matching filter) and discards it automatically depending on determinant and SVD values. We sort the number of matches and start computing the homography with the most promising method. If a valid homography is found we assume that the detected image is correct, ignoring possible successes with other reference images. In the event that the homography cannot be computed or it presents errors, the application attempts with the next choices according to the number of matches.

This application was developed in Java (for the user interface) and C++ for interaction with OpenCV 2.4 through JNI. It's available for download on Google Play (click below link or scan QR code to download):

https://play.google.com/store/search?q=%23cvriousity

Source code is available as well on GitHub (https://github.com/RamiroOliva/CVriousity) and is licensed under Apache 2.0 License.

FUTURE WORK

There is a significant difference in efficacy between the methods of desktop and mobility for OpenCV. Since SIFT and SURF are subject to patents, from a free software view, it would be very interesting to see BRISK incorporated into OpenCV mobile versions, a novel method we found far superior in effectiveness (and possibly quicker) to current methods in mobility.

The strategy of recognition of images in real time is very different from batch or sequential recognition. In fact, the real-time recognition of large banks of images requires appropriate analysis that has not been addressed in this research. During the development of the sample application we have found that the homography computation and verification process is very time expensive and should be carried out only when there is adequate evidence of not failing (for example considering a large number of matchings and their quality). We have observed that FAST produced higher number of matches which slows down the RANSAC algorithm used in homography computation. It is desirable to have a lower number but highter quality of descriptors which can be achieved through other methods such as GFTT or MSER. Reducing the number of features found by boosting techniques would be another line of possible research [25].

Finally a proper frame pre-processing is critical to improving the quality of recognition (as the focus or scaling). Our application focus and rescale the frame to match the screen resolution and thus reduce the number of features. When the artwork is far from the user and appears small on the screen, it would be an interesting possibility, since the cameras have a higher resolution than the screen, to recognize different sizes of the same captured frame, cropping around the center of the scene, producing actually a digital zoom effect.

ACKNOWLEDGMENT

Special thanks to Ph. D. Xavier Baró Solé, for their advice and assistance both in the field of image recognition by computer as in the preparation of this article, as well as a Ph. D. Alexandre Viejo Galicia, who has guided me in the development of the project and research methodology.

REFERENCES

[1] Mikolajczyk, K., and Schmid, C., "A performance evaluation of local descriptors", IEEE Transactions on Pattern Analysis and Machine Intelligence, 10, 27, pp 1615--1630, 2005.
[2] Bauer, J., S¨underhauf, N., & Protzel, P. (2007). Comparing Several Implementations of Two Recently Published Feature Detectors. In Proc. of the International Conference on Intelligent and Autonomous Systems, IAV, Toulouse, France.
[3] Comparing State-of-the-Art Visual Features on Invariant Object Recognition Tasks, Nicolas Pinto1, Youssef Barhomi1, David D. Cox2, and James (2011).
[4] En el grupo de la red social LinkedIn "Computer Vision and Pattern Recognition", integrado por más de 6000 miembros, se formuló la pregunta "What is, in your opinion, the most advanced technique for recognizing objects in a photograph in terms of accuracy?". Available: http://lnkd.in/m3NSNk.
[5] Lowe, David G. (1999). "Object recognition from local scale-invariant features". Proceedings of the International Conference on Computer Vision. pp. 1150–1157.DOI:10.1109/ICCV.1999.790410.
[6] U.S. Patent 6,711,293, "Method and apparatus for identifying scale invariant features in an image and use of same for locating an object in an image", David Lowe's patent for the SIFT algorithm, March 23, 2004.
[7] US 2009238460.
[8] En la red social LinkedIn, en el grupo de discusión "Computer Vision and Pattern Recognition", se formuló la pregunta "What is your favorite framework/library for image understanding/recognition projects?" obtuviendo 197 votos, de los cuales más del 90% fueron a la librería libre OpenCV. Available:http://lnkd.in/_dv6X2
[9] Open Source Computer Vision (http://opencv.willowgarage.com/wiki/).
[10] Press reference: http://ucrtoday.ucr.edu/5453
[11] http://www.iccv2011.org
[12] http://eccv2012.unifi.it/
[13] http://www.cvpr2012.org
[14] http://pascallin.ecs.soton.ac.uk/challenges/VOC
[15] http://pascallin.ecs.soton.ac.uk/challenges/VOC/voc2011/results/index.html#KEY_NUSPSL_CTX_GPM
[16] http://www.vision.caltech.edu/Image_Datasets/Caltech256/intro
[17] "Comparison of OpenCV's feature detection algorithms" - http://computer-vision-talks.com/2011/07/comparison-of-the-opencvs-feature-detection-algorithms-ii/
[18] Krystian Mikolajczyk and Cordelia Schmid , "A performance evaluation of local descriptors", 23 Febrero 2005.
[19] Source: nota de prensa IDC, 8 Agosto 2012, "Android and iOS Surge to New Smartphone OS Record in Second Quarter, According to IDC", http://www.idc.com/getdoc.jsp?containerId=prUS23638712
[20] BRISK: Binary Robust Invariant Scalable Keypoints. Available: http://www.asl.ethz.ch/people/lestefan/personal/BRISK
[21] Zhaowei Li and David R. Selviah , "Comparison of Image Alignment Algorithms ". Available: http://www.ee.ucl.ac.uk/lcs/previous/LCS2011/LCS1115.pdf
[22] Etienne Vincent and Robert Laganiere, "Detecting Planar Homographies in an Image Pair".
[23] Lionel Moisan , Pierre Moulon , Pascal Monasse, "Automatic Homographic Registration of a Pair of Images, with A Contrario Elimination of Outliers". Available: http://www.ipol.im/pub/art/2012/mmm-oh/article.pdf
[24] Janez Demšar, "Statistical Comparisons of Classifiers over Multiple Data Sets", Journal of Machine Learning Research 7 (2006) 1–30.
[25] Xavier Baró i Solé, "Probabilistic Darwin Machines: A new approach to develop Evolutionary Object Detection Systems", UAB, Feb 2009, Appendix B.
[26] http://www.sussex.ac.uk/Users/grahamh/RM1web/F-ratio%20table%202005.pdf
[27] J.H. Zar. Biostatistical Analysis. Prentice Hall, 1998.
[28] D.J. Sheskin. Handbook of parametric and nonparametric statistical procedures. Chapman & Hall/CRC, 2000.
[29] R.L. Iman and J.M. Davenport. Approximations of the critical region of the Friedman statistic. In *Communications in Statistics*, pages 571-595, 1980.
[30] P.B. Nemenyi. *Distribution-free multiple comparisons*. PhD thesis, Princeton University, 1963.
[31] FAST: Edward Rosten and Tom Drummond, "Machine learning for high-speed corner detection", 2006. Available: http://www.edwardrosten.com/work/rosten_2006_machine.pdf
[32] MSER: J. Matas, O. Chum, M.Urban, T. Pajdla, "RobustWide Baseline Stereo from Maximally Stable Extremal Regions", 2002. Available: http://cmp.felk.cvut.cz/~matas/papers/matas-bmvc02.pdf
[33] GFTT: Jianbo Shi, Carlo Tomasi, "Good Features to Track", 1994. Available: http://www.cs.duke.edu/~tomasi/papers/shi/shiCvpr94.pdf
[34] J. R. Quinlan, "Induction of decision trees," Machine Learning, vol. 1, pp. 81–106, 1986.
[35] Motilal Agrawal and Kurt Konolige. "CenSurE: Center Surround Extremas for realtime feature detection and matching". In ECCV, 2008.
[36] Ethan Rublee, Vincent Rabaud, Kurt Konolige, Gary Bradski, "ORB: an efficient alternative to SIFT or SURF". Available: http://www.willowgarage.com/sites/default/files/orb_final.pdf
[37] Herbert Bay, Tinne Tuytelaars, and Luc Van Gool, "SURF: Speeded Up Robust Features". Available: http://www.vision.ee.ethz.ch/~surf/eccv06.pdf
[38] Marius Muja, David G. Lowe, " Fast Approximate Nearest Neighbors with Automatic Algorithm Configuration (FLANN)", 2009. Available: http://people.cs.ubc.ca/~mariusm/uploads/FLANN/flann_visapp09.pdf
[39] Alexandre Alahi, Raphael Ortiz, Pierre Vandergheynst, "FREAK: Fast Retina Keypoint". In IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2012 Open Source Award Winner. Available: http://www.ivpe.com/papers/freak.pdf

LICENSE