OpenCV Performance Measurements on Mobile Devices

Marco A. Hudelist
Klagenfurt University, ITEC
Universitätsstraße 65-67
9020 Klagenfurt, Austria
marco@itec.aau.at

Claudiu Cobârzan
Klagenfurt University, ITEC
Universitätsstraße 65-67
9020 Klagenfurt, Austria
claudiu@itec.aau.at

Klaus Schoeffmann
Klagenfurt University, ITEC
Universitätsstraße 65-67
9020 Klagenfurt, Austria
ks@itec.aau.at

ABSTRACT

Mobile devices like smartphones and tablets are becoming increasingly capable in terms of processing power. Although they are already used in computer vision, no comparable measurement experiments of the popular OpenCV framework have been made yet. We try to fill this gap by evaluating the performance of a set of typical OpenCV operations, on mobile devices like the iPad Air and iPhone 5S. We compare those results with the performance of a consumer grade laptop PC (MacBook Pro). Our tests span from simple image manipulation methods to keypoint detection and descriptor extraction as well as descriptor matching. Results show that the top performing device can match the performance of the PC up to 80 percent in specific operations.

Categories and Subject Descriptors

H.3.4 [**Systems and Software**]: Performance evaluation (efficiency and effectiveness)

General Terms

Measurement, Experimentation, Performance

Keywords

Mobile devices, OpenCV, performance evaluation

1. INTRODUCTION

Mobile devices like smartphones and tablets are a popular replacement for point-and-shoot cameras, music players and to some degree even traditional laptops and desktop PCs. At the same time they remain lightweight, offer simpler user interfaces and provide in some cases impressive processing power when compared to classical computers. Over the last years such devices experienced big improvements also in terms of battery life. Multi-core CPUs and GPUs have been introduced and the latest installment of Apple's iPhones and iPads even provide 64-Bit computing architectures. Since they have the ability to become the primary computing device for the majority of consumers, it makes sense to expand the research of computer vision to such devices. For tasks of type "known item search" within picture or video archives, the focus is still on desktop solutions [10], but there are approaches which employ collaborative browsing on mobile devices [5]. For such solutions, performing feature extraction directly on the device might prove beneficial. The OpenCV framework [1] is a very popular tool in this field and is available for various mobile platforms. In spite of the fact that various device performance benchmark results are available, none of those specifically target the needs of computer vision applications and research. This work aims at providing the computer vision community with a first set of performance measurements of popular OpenCV operations on tablets and smartphones. We have used for our tests different generations of iPad and iPhone devices and contrasted the results with those obtained for the same set of operations on a typical consumer grade laptop PC (MacBook Pro). By doing this we aim at providing a helpful and substantiated reference for choosing appropriate parings of type *device - OpenCV operation*.

In the following we lay out the details of our experiment including the datasets used and the exact procedure with emphasis on the different phases of the measurements. The results are presented, discussed and recommendations concerning the use of OpenCV operations on mobile devices are given.

To the best of our knowledge there are no other works dealing with similar performance measurements of OpenCV operations on mobile devices.

To appear in Proc. ICMR 2014, April 1-4, 2014, Glasgow, Scotland, UK ©ACM 2014

2. MEASUREMENT SETUP AND RESULTS

We group the tested OpenCV operations into three measurement phases. In the first phase we test typical operations used in the computer vision area like blurring an image using Gaussian blur, detecting faces and detecting edges. In the next phase we group functions calls to common keypoint detection and descriptor extraction operations like SIFT [7] and SURF [3]. The last phase groups operations that match already extracted descriptors of two subsequently frames of a video.

For the measurements of the first two phases we use a dataset consisting of 5000 images[1]. The images are randomly drawn from the freely available MIRFLICKR25000 dataset[2].

[1]The file list is the one at `http://ngvb.net/?page_id=158`
[2]The MIRFLICKR2500 image dataset is available at `http://press.liacs.nl/mirflickr`

The images differ in resolution and have an average of 463 horizontal and 397 vertical pixels. Each function call is tested with each of these images five times in a row and the measured times are averaged. This is done in order to even out measurement differences caused by unpredictable interventions of the OS. The averaged performance times for each image are again averaged to get an overall performance measure for each specific OpenCV function.

The last phase which concentrates on descriptor matching uses a different dataset than the phases before. We extract the first 5000 frames of the *test video 031*[3] of the Video Browser Showdown 2013 [10] - a recording of a dutch news journal. The video is encoded in H.264 with an average bitrate of 619.7 kBit/s and a resolution of 640 x 360 pixels. The tested matching approaches differ in terms of the used descriptors but all use the same brute force matcher (BFMatcher) that is part of the OpenCV framework. In this phase, only the actual matching process is measured, not the keypoint detection and descriptor extraction. Each matching approach has to find matches between two subsequently frames, starting with the first and second frame of the video, continuing with the second and third frame, etc. This continues until the last pair of the 5000 frames. Each matching process of a frame pair is repeated five times and the measured times are averaged. The averaged times of all frame matches are again averaged to get the overall measurement for each matching approach.

We use different generations of Apple's iPads and iPhones as hardware, including their latest and fastest versions, the iPad Air and the iPhone 5S. As ground truth, we also evaluate the performance of a MacBook Pro 13" with Retina Display (late 2012 version) with an Intel Core i5 at 2.5 GHz, 8 GB of RAM and integrated Intel HD Graphics 4000. For a comparison of the specifications of the devices please see Table 1. On all of the mobile devices we use the latest available OS version iOS 7.0.4. The MacBook Pro uses Mac OS X 10.8 Mountain Lion. Further, version 2.4.7 of the OpenCV framework is used on all devices.

Table 1: Specification breakdown

Device	CPU Clock	CPU Cores	CPU Architecture
MBP 13" (2012)	2.5 GHz	2	64 Bit
iPad Air	1.4 GHz	2	64 Bit
iPad 4	1.4 GHz	2	32 Bit
iPad 3	1 GHz	2	32 Bit
iPad Mini 1	1 GHz	2	32 Bit
iPad 2	1 GHz	2	32 Bit
iPhone 5S	1.3 GHz	2	64 Bit
iPhone 5	1.3 GHz	2	32 Bit
iPhone 4s	800 MHz	2	32 Bit

It has to be noted that we did not use in any of our measurements neither custom parallelization nor GPU supported calls. The values therefore indicate only the performance on a single core of the devices' CPUs.

2.1 Common OpenCV Operations

In this testing phase we evaluate how long it takes to:

- Grayscale an image
- Blur an image with Gaussian Blur
- Detect faces in an image
- Calculate the RGB and HSV histograms
- Detect edges using Canny edge detection

For the Gaussian blur we use the OpenCV GaussianBlur-function with a kernel size of 21 x 21 and a sigma of 8.0 to produce recognizable blurred result images. The face detection is realized by using a trained cascade classifier and its detectMultiScale function. Images are converted to grayscale before the actual detection takes place. We use a scaleFactor of 1.1, minNeighbors of 2 and a minimum size of 30 x 30. The RGB histograms are calculated with a size of 256 bins, a range from 0 to 256 and uniform set to true and accumulate set to false. The HSV histograms are calculated with 30 hue levels and 32 saturation levels with the standard ranges. The default values for uniformity (true) and accumulation (false) are used. For the edge detection using the Canny algorithm we first grayscale the image and blur it with a kernel size of 5 x 5 and a sigma of 1.2. We then measure the Canny function of OpenCV with thresholds one and two set to 0 and 50 respectively. For a breakdown of the measurement see Table 2.

2.2 Keypoint Detection and Descriptor Extraction

For the keypoint detection and descriptor extraction part, we evaluate several popular algorithms that come built-in in the OpenCV framework. The measurements results of Table 3 include the time needed to detect keypoints as well as to extract descriptors based on the detected keypoints. The algorithms that we evaluate in this block are ORB [9], BRIEF [4], BRISK [6], SIFT, SURF, FREAK [2], FAST [8]. In case of BRIEF and FREAK we pare the descriptor extraction with the GoodFeaturesToTrack algorithm [11] to detect keypoints. For a breakdown of all measures please see Table 3.

2.3 Descriptor Matching

In the matching part we evaluate the performance of SIFT, SURF, BRISK and BRIEF descriptors regarding how fast they can be matched for two successive frames of a video. For each of the frames the preceding process of keypoint detection and descriptor extraction is not part of the measurement. For the actual matching we use the brute force matcher (BFMatcher) that comes with the OpenCV framework. For a breakdown of all matching results please see Table 4.

3. DISCUSSION

It is clear that even high end mobile devices still do not match the processing power of typical consumer grade PCs. Nevertheless, their performance is promising and can be sufficient for certain kinds of applications. Especially the latest edition tablets and smartphones (iPad Air and iPhone 5S) showed that they could match the power of the PC up to 60 percent for certain scenarios concerning face detection and up to 80 percent in terms of grayscaling images, as can be seen in Figure 1. Generation of HSV histograms and Canny

[3]http://www.videobrowsershowdown.org

Table 2: Common Operations (values in ms)

Device	Grayscale	Gaussian Blur	Face Detection	RGB Hist.	HSV Hist.	Canny Edge Detection
MBP 13" (2012)	0.26	3.97	134.27	0.40	0.30	3.10
iPad Air	0.32	42.31	214.10	2.48	0.89	9.34
iPad 4	0.58	80.66	282.30	2.47	1.26	15.49
iPad 3	1.18	153.64	502.37	2.88	3.10	26.22
iPad Mini 1	1.17	151.19	505.66	2.94	3.17	26.14
iPad 2	1.15	153.30	500.07	2.91	3.14	26.11
iPhone 4s	1.30	190.00	620.62	3.55	3.84	32.10
iPhone 5	0.61	86.32	296.73	2.63	1.35	16.39
iPhone 5S	0.34	45.28	235.56	2.49	0.92	9.73

Table 3: Keypoint detection and descriptor extraction (values in ms)

Device	ORB	BRISK	FREAK	FAST	BRIEF	SIFT	SURF
MBP 13" (2012)	10.73	248.14	17.04	1.07	1.87	172.71	341.32
iPad Air	52.37	775.70	46.49	7.72	3.64	877.09	805.50
iPad 4	80.30	1167.24	84.48	15.03	6.89	1378.78	842.3.0
iPad 3	162.99	2155.05	184.84	24.74	17.32	3518.18	1606.63
iPad Mini 1	162.96	2156.24	185.66	24.86	17.63	3586.21	1627.40
iPad 2	162.23	2150.66	184.24	24.70	17.20	3515.13	1599.75
iPhone 4s	200.54	2673.05	222.74	30.59	20.77	4320.10	1980.38
iPhone 5	84.41	1232.42	89.92	16.11	7.42	1474.19	804.29
iPhone 5S	55.15	829.54	49.04	8.22	3.76	1028.17	1024.97

Table 4: Frame matching (values in ms)

Device	SIFT	SURF	BRISK	BRIEF
MBP 13" (2012)	51.98	33.98	2.11	15.69
iPad Air	254.24	163.93	3.23	23.63
iPad 4	394.12	204.11	6.68	37.78
iPad 3	1671.09	746.41	14.38	76.103
iPad Mini 1	1643.27	756.31	14.47	76.08
iPad 2	1610.51	744.96	14.72	75.89
iPhone 4s	1888.81	918.46	17.94	94.405
iPhone 5	421.38	243.62	8.43	40.46
iPhone 5S	390.00	223.53	3.74	32.52

edge detection could be performed with about 30 percent of the processing performance of a traditional PC.

In terms of keypoint detection and descriptor extraction, the top devices could provide up to 50 percent the performance of the traditional PC in case of BRIEF and up to 40 percent in case of SURF (see Figure 2).

Descriptor matching showed a rather good performance of the iPad Air and iPhone 5S with BRISK and BRIEF descriptors. The devices are able to provide about 65 percent of the performance of a normal PC (see Figure 3). In case of matching SIFT and SURF descriptors, the results are not that good with only 20 percent of performance of a traditional PC.

Another interesting result is that the iPad 3 is slightly outmatched by its predecessor, the iPad 2 in every measurement. We expect this has to do with the fact, that the iPad 3 was the first iPad with a Retina Display. It has to process four times the pixel amount of the predecessor, which likely causes the performance boost of the newer processor to be eaten up.

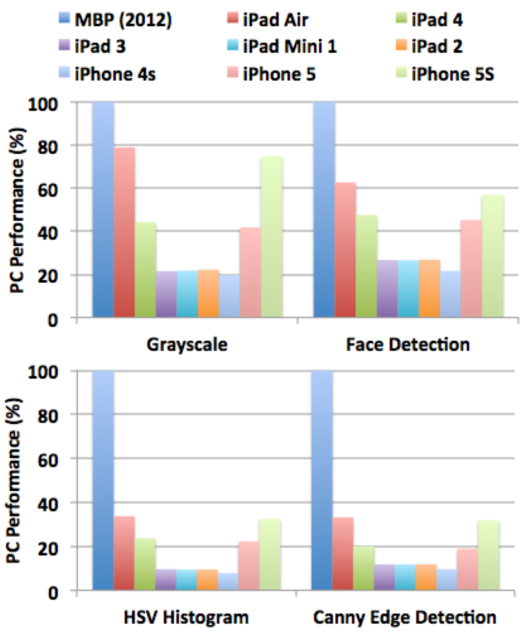

Figure 1: Top results of common operations.

4. CONCLUSION

In this paper we showed results for performance measurements of common OpenCV functions executed on mobile devices. Our measurements were grouped into common operations, keypoint detection and descriptor extraction as well as descriptor matching. We could show that certain operations were only two to three times slower on the top performing device compared to a typical consumer grade laptop

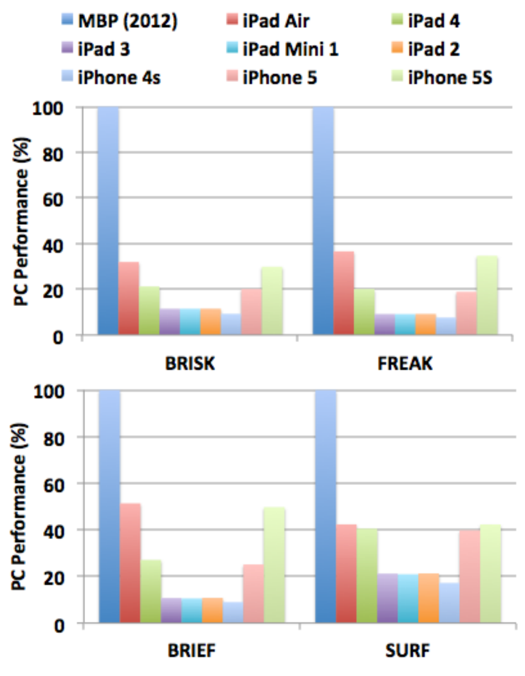

Figure 2: Top results of keypoint detection and descriptor extraction.

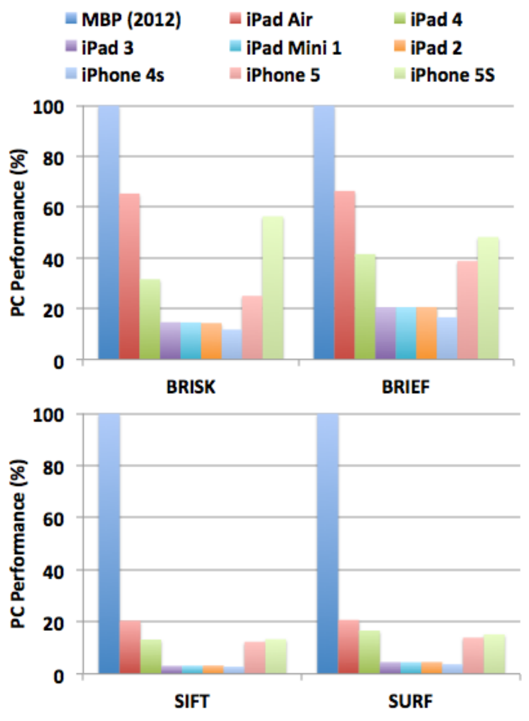

Figure 3: Results of the matching measurements.

computer. In future work we would like to expand our measurements in terms of tested operations as well as in terms of tested hardware. Additionally to iPads and iPhones we would like to add popular Android and Windows tablets and smartphones. Further, we want to explore the parallel and GPU supported computation possibilities that are available on the devices and in the OpenCV framework and compare their performaces to those of the CPU only versions.

5. ACKNOWLEDGMENTS

This work was funded by the Federal Ministry for Transport, Innovation and Technology (bmvit) and the Austrian Science Fund (FWF): TRP 273-N15 and the European Regional Development Fund and the Carinthian Economic Promotion Fund (KWF), supported by Lakeside Labs GmbH, Klagenfurt, Austria.

6. REFERENCES

[1] http://www.opencv.org (last visited 27.11.2013).

[2] A. Alahi, R. Ortiz, and P. Vandergheynst. Freak: Fast retina keypoint. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 510–517, 2012.

[3] H. Bay, T. Tuytelaars, and L. Gool. Surf: Speeded up robust features. In A. Leonardis, H. Bischof, and A. Pinz, editors, *Computer Vision - ECCV 2006*, volume 3951 of *Lecture Notes in Computer Science*, pages 404–417. Springer Berlin Heidelberg, 2006.

[4] M. Calonder, V. Lepetit, C. Strecha, and P. Fua. Brief: Binary robust independent elementary features. In K. Daniilidis, P. Maragos, and N. Paragios, editors, *Computer Vision - ECCV 2010*, volume 6314 of *Lecture Notes in Computer Science*, pages 778–792. Springer Berlin Heidelberg, 2010.

[5] C. Cobârzan, M. A. Hudelist, and M. Del Fabro. Content-based video browsing with collaborating mobile clients. In C. G. et al., editor, *20th Anniversary International Conference on MultiMedia, Part II - MMM 2014*, volume 8326 of *Lecture Notes in Computer Science*, pages 402–406. 2014.

[6] S. Leutenegger, M. Chli, and R. Siegwart. Brisk: Binary robust invariant scalable keypoints. In *IEEE International Conference on Computer Vision (ICCV)*, pages 2548–2555, 2011.

[7] D. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110, 2004.

[8] E. Rosten and T. Drummond. Machine learning for high-speed corner detection. In A. s. Leonardis, H. Bischof, and A. Pinz, editors, *Computer Vision - ECCV 2006*, volume 3951 of *Lecture Notes in Computer Science*, pages 430–443. Springer Berlin Heidelberg, 2006.

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

[10] K. Schoeffmann, D. Ahlström, W. Bailer, C. Cobârzan, F. Hopfgartner, K. McGuinness, C. Gurrin, C. Frisson, D.-D. Le, M. Del Fabro, H. Bai, and W. Weiss. The video browser showdown: A live evaluation of interactive video search tools. *International Journal of Multimedia Information Retrieval*, 2014. accepted on Dec. 4, 2013.

[11] J. Shi and C. Tomasi. Good features to track. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 593–600, 1994.