Mobile Image Analysis: Android vs. iOS

Claudiu Cobârzan, Marco A. Hudelist,
Klaus Schoeffmann, Manfred Jürgen Primus

Alpen-Adria-Universität Klagenfurt
9020 Klagenfurt, Austria
{claudiu, marco, ks, mprimus}@itec.aau.at

Abstract. Currently, computer vision applications are becoming more common on mobile devices due to the constant increase in raw processing power coupled with extended battery life. The OpenCV framework is a popular choice when developing such applications on desktop computers as well as on mobile devices, but there are few comparative performance studies available. We know of only one such study that evaluates a set of typical OpenCV operations on iOS devices. In this paper we look at the same operations, spanning from simple image manipulation like grayscaling and blurring to keypoint detection and descriptor extraction but on flagship Android devices as well as on iOS devices and with different image resolutions. We compare the results of the same tests running on the two platforms on the same datasets and provide extended measurements on completion time and battery usage.

Keywords: mobile devices, OpenCV, performance evaluation, Android, iOS

1 Introduction

When it comes to taking photos or shooting short videos, smartphones and tablet computers are nowadays often preferred over point-and-shoot cameras or camcorders. Also, they represent a popular alternative to dedicated devices like mp3 players, mobile games consoles, or even traditional laptops and desktop PCs when engaged in entertainment activities like listening to music, watching videos, browsing through photos, playing games, etc. This has been made possible, among other factors, by the increase in processing power and improved battery life. Multi-core CPUs and GPUs are no longer uncommon both on Apple and Android devices and installments that provide 64-Bit computing architectures have recently entered the market.

Due to their popularity among consumers as well as their high availability and versatility, smartphones and tablets tend to become primary computing devices and, as a result, the interest of computer vision research on such devices is on the rise. Approaches for performing visual search using mobile devices have already been proposed [6]. Recently, collaborative video browsing on mobile devices [5] has been explored for known-item search tasks, as opposed to more standard

solutions which still concentrate on desktop applications [12]. For such mobile approaches, performing feature extraction directly on the device might prove beneficial in various scenarios.

2 Related Work

In [4] the performance and computational cost of various keypoint detection, feature extraction and encoding algorithms are evaluated on a Samsung Galaxy S3 device. To the best of our knowledge, the work in [7] is the only one that deals with OpenCV measurements on iOS mobile devices (other platforms are not considered). Different generations of Apple's iPads and iPhones are used with ground truth measurements being performed on a MacBookPro. The tested OpenCV operations are grouped into three measurement phases as follows: (phase 1) typical computer vision operations: *grayscaling*, *image blurring* using Gaussian blur, *face* and *edge detection*; (phase 2) *keypoint detection* and *descriptor extraction* operations using SIFT [9] and SURF [2]; (phase 3) *matching descriptors* extracted from two subsequent frames of a video. For the first two phases, a dataset of 5000 images randomly drawn from the MIRFLICKR25000 dataset is used.

The reported results show that high end mobile devices are often only two times slower than common desktop computers. Some devices 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. Generation of HSV histograms and Canny 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 some of the considered cases. Descriptor matching showed a rather good performance in some cases (iPad Air and iPhone 5S), providing about 20 to 65 percent of the performance of a normal PC, depending on the considered descriptors. Those results show that mobile devices have already remarkable performance which will further increase in fast pace over the years, making multimedia content analysis on these devices affordable.

3 Goals and Measurements Setup

The main goal of the current work is to provide a reference as well as a comparison for common OpenCV operations on both Android and iOS mobile devices in terms of completion time and battery usage. We concentrate on those two platforms because they have the largest shares on the smartphone and tablet markets. We consider this important since it gives a clear image about which operations can be performed directly on such mobile devices and which cannot. Previous work [7] has concentrated only on the iOS platform and considered some of the OpenCV operations on a coarser level of granularity (e.g. no details on keypoint detection vs. descriptor extraction completion times). Moreover, to the best of out knowledge, there is no work about battery consumption of commonly used OpenCV operations on mobile devices.

We use the same OpenCV operations as in [7]:

- Phase 1: typical computer vision operations: grayscaling, image blurring using Gaussian blur, face and edge detection;
- Phase 2: keypoint detection and descriptor extraction operations using SIFT [9] and SURF [2].

For both phase 1 & phase 2 we have used 250 pictures drawn from the INRIA Holiday dataset (`https://lear.inrialpes.fr/ jegou/data.php` where is freely available). The file list is the one at `http://ngvb.net/?page_id=158`. The pictures all surpass 3 mega pixel and were organized in 3 setups: ***(a)*** the original 3MP images (noted as the 3MP dataset), ***(b)*** the images scaled to 50% of the original size (noted as the 3MP_50 dataset) and ***(c)*** the images scaled to 25% of the original size (noted as the 3MP_25 dataset). The original images vary in size from a minimum of 1536×20148 to a maximum of 2592×3888 (an average of 2547 horizontal by 2016 vertical pixels).

The main motivation in doing so was to perform the tests on data that better reflects the variety of image and video acquisition capabilities of the used devices. Within each set there is minor variance in image resolution since it is expected that future computer vision applications will use heterogeneous data from multiple sources, including data captured by the device as well as data coming from other sources.

The approach was the same as in [7], meaning that for phase 1 & 2 we have tested the corresponding function calls with each image five times in a raw and then we averaged those five measurement times in order to even out possible interventions by the OS. Those averaged times for each image were again averaged to get the overall performance measure for each specific OpenCV function.

The devices used for the experiment include flagship Android devices, namely the Nexus and Galaxy product lines as well as different generations of Apple's iPads and iPhones. The brief specifications of the used devices are presented in Table 1. The CPUs on the iPad Air and the iPhone 5S have a 64 Bit architecture while all the other devices have a 32 Bit CPU architecture. The System On Chip information is presented in Table 2. All Android devices use Android version 4.4.2 except for the GalaxyNote 10.1 which uses Android version 4.3. The iOS devices used version 7.1.2. All devices had OpenCV Release 2.4.9 installed via Google Play and Apple App Store respectively .

4 Results and Discussion

In the following we provide an overview of the obtained results with respect to both completion time and battery usage. It is important to mention that the measured values reflect the performance on a single core of the devices' CPUs since we did not employed custom parallelization or GPU supported calls but just used the default OpenCV functions, which are not parallelized to best

Table 1: Android and iOS devices specification breakdown

Device	CPU clock	CPU cores	Memory	Screen Size	Screen Resolution	Battery Capacity
Galaxy Note 10.1 (2014 Edition)	1.95 GHz	4+4	3 GB	10.1-inch	2560×1600	8220 mAh
Galaxy Note 3	2.3 GHz	4	3 GB LPDDR3	5.7-inch	1920×1080	3200 mAh
Galaxy S4	1.9 GHz	4	2 GB LPDDR3	5.0-inch	1920×1080	2600 mAh
Nexus 7 (2013)	1.5 GHz	4	2 GB DDR3L	7.0-inch	1920×1200	3950 mAh
Nexus 7	1.2 GHz	4	1 GB DDR3L	7.0-inch	1280×800	4325 mAh
Nexus 5	2.3 GHz	4	2 GB LPDDR3	4.95-inch	1920×1080	2300 mAh
iPad Air	1.4 GHz	2	1 GB	9.7-inch	2048×1536	8820 mAh
iPad 4	1.4 GHz	2	1 GB DDR2	9.7-inch	2048×1536	11560 mAh
iPad 3	1 GHz	2	1 GB LP DDR2	9.7-inch	2048×1536	11560 mAh
iPad Mini 1	1 GHz	2	512 MB DDR2	7.9-inch	1024×768	4382 mAh
iPad 2	1 GHz	2	512 MB DDR2	9.7-inch	1024×768	6930 mAh
iPhone 5S	1.3 GHz	2	1 GB LPDDR3	4.0-inch	1136×640	1560 mAh
iPhone 5	1.3 GHz	2	1 GB LPDDR2	4.0-inch	1136×640	1440 mAh
iPhone 4s	800 MHz	2	512 MB DDR2	3.5-inch	960×640	1432 mAh

Table 2: Android and iOS System on Chip information

Device	CPU SoC
Galaxy Note 10.1 (2014 Edition)	Samsung Exynos 5420
Galaxy Note 3	Qualcomm Snapdragon 800
Galaxy S4	Qualcomm Snapdragon 600
Nexus 7 (2013)	Qualcomm Snapdragon S4 Pro
Nexus 7	Nvidia Tegra 3 T30L
Nexus 5	Qualcomm Snapdragon 800
iPad Air	Apple A7
iPad 4	Apple A6X
iPad 3	Apple A5X
iPad Mini 1	Apple A5 (2nd Generation)
iPad 2	Apple A5
iPhone 5S	Apple A7 / Apple M7
iPhone 5	Apple A6

of our knowledge. The keypoint detection and descriptor extraction operations are measured independently. The battery usage measurements were performed with 1% granularity steps for the Android devices and with 5% granularity steps for the Apple devices in accordance with the support provided by the corresponding APIs. The tests were started on all devices with the battery fully charged. This means that there is a ±1% error margin for the Android devices and a ±5% error margin for the iOS devices since it is possible that a previous test has ended just under the threshold and part of it's battery consumption gets counted for the following test.

4.1 Common OpenCV Operations

The common OpenCV operation we have measured (corresponding to the phase 1 measurements we mentioned above) are:

- Grayscaling an image
- Blurring an image using Gaussian Blur
- Face detection in an image
- RGB and HSV histogram calculations
- Edge detection using the Canny algorithm

Blurring was performed using the OpenCV GaussianBlur-function with a kernel size of 21×21 and sigma set to 8.0 in order to produce recognizable result images. For face detection we used a trained cascade classifier and its *detectMultiScale* function. Before the actual detection, the images were grayscaled. The grayscaling was not included in the time measurement. The other used parameters were the scaleFactor set to 1.1, the minNeighbors set to 2 and the minimum size set to 30×30. The RGB histograms used 256 bins within the range of 0 to 256, had uniform set to true and accumulate set to false. The HSV histograms considered 30 hue levels and 32 saturation levels with the standard ranges and used the default values for uniformity (true) and accumulation (false). When detecting edges with the Canny algorithm we first grayscaled the image and then blurred it using a kernel size of 5×5 and a sigma of 1.2 (both operations were not included in the time measurement). After completing those two precursor operations, we measured the Canny function of OpenCV with the thresholds one and two set to 0 and 50 respectively. The measured values are presented in Table 3 and Table 4 for the 3MP and 3MP_50 datasets.

For a visual aid when analyzing the data in Table 3 and Table 4, please refer to Figure 1 and Figure 2. Both use a logarithmic scale for representing the measured values for OpenCV common operations in the case of the 3MP and 3MP_50 datasets. It can be seen that all the Android devices are faster then the fastest iOS device which is for the considered set of operations the iPhone 5S. Among the Android devices the best performances is achieved by the Galaxy Note 10.1. As already noted in [7], the iPad 2 outmatches its successor, the iPad 3 in all measurements. The explanation lies in the fact that since the iPad 3 is equipped with a Retina Display, it processes 4 times more pixels then

it's predecessor and this takes its toll on the overall performance despite the newer processor.

Table 3: Common Operations (values in ms) - 3MP images

Device	Grayscale	Gaussian Blur	Face Detection	RGB Histogram	HSV Histogram	Canny Edge Detection
Galaxy Note 10.1	8.35	592.62	2471.53	11.64	6.00	145.01
Galaxy Note 3	10.05	680.84	3297.80	19.33	6.40	153.25
Galaxy S4	14.55	886.78	4261.88	28.08	9.96	205.38
Nexus 7 (2013)	14.60	1014.20	3439.15	30.03	9.22	240.46
Nexus 7	11.43	1501.77	3273.17	27.55	20.56	193.18
Nexus 5	7.94	840.87	3494.73	25.19	7.65	144.22
iPad Air	12.95	1260.53	7930.91	69.84	25.06	282.01
iPad 4	21.28	2464.97	10400.87	61.26	33.38	384.51
iPad 3	51.97	4604.35	16556.63	81.00	87.37	797.74
iPad Mini 1	31.26	2769.81	9954.92	47.69	52.92	493.89
iPad Mini 2	13.62	1368.43	8771.00	79.85	27.76	328.01
iPad 2	30.51	2868.00	9842.43	47.09	52.20	488.23
iPhone 5S	8.12	1106.79	6338.24	42.84	15.98	179.60
iPhone 5	13.27	1705.18	6615.14	39.27	21.05	245.03

Table 4: Common Operations (values in ms) - 3MP_50 images

Device	Grayscale	Gaussian Blur	Face Detection	RGB Histogram	HSV Histogram	Canny Edge Detection
Galaxy Note 10.1	2.36	135.85	556.21	4.08	1.82	37.20
Galaxy Note 3	3.33	151.62	734.67	7.90	2.80	44.07
Galaxy S4	3.99	181.88	933.57	9.89	4.06	61.18
Nexus 7 (2013)	4.63	213.22	780.29	8.84	4.30	61.81
Nexus 7	3.33	358.98	798.09	7.11	5.36	46.60
Nexus 5	2.55	220.05	867.93	8.04	3.17	65.85
iPad Air	3.32	270.05	1815.28	19.26	6.17	69.23
iPad 4	6.20	534.86	2666.15	16.02	9.13	98.01
iPad 3	13.35	1077.73	3936.46	20.48	21.95	196.59
iPad Mini 1	13.23	1074.42	3971.17	19.84	21.88	199.14
iPad Mini 2	3.43	289.37	2376.85	22.43	6.88	79.66
iPad 2	12.84	1076.47	3930.68	19.64	21.59	196.88
iPhone 5S	3.43	288.90	2858.57	16.82	6.60	73.12
iPhone 5	6.66	573.81	2951.05	16.91	9.60	102.293

Among the considered operation, the most demanding are the face detection, Gaussian blur and Canny edge detection while the least demanding are the HSV histogram and the grayscaling.

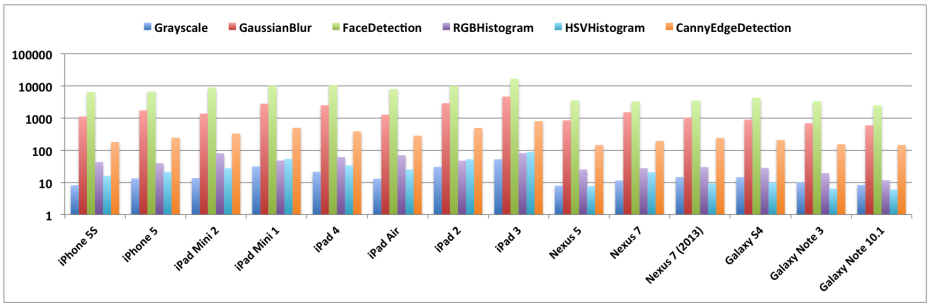

Fig. 1: Measured results of common operations in the case of the 3MP dataset.

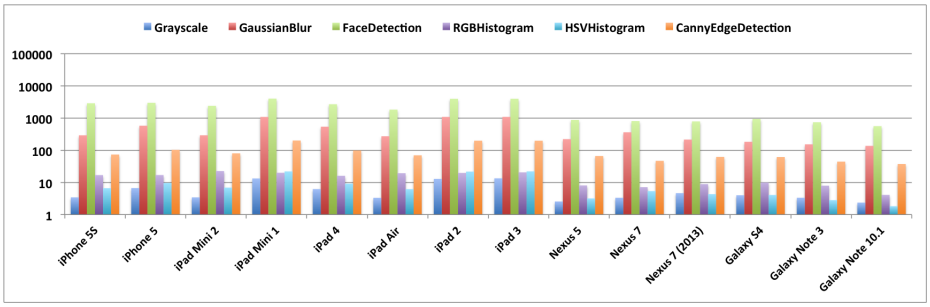

Fig. 2: Measured results of common operations in the case of the 3MP_50 dataset.

Table 5: Battery drop levels for common operations (values in %) - 3MP (D1) and 3MP_50 (D2) datasets

Device	Grayscale		Gaussian Blur		Face Detection		RGB Histogram		HSV Histogram		Canny Edge Detection	
	D1	D2	D1	D2	D1	D2	D1	D2	D1	D2	D1	D2
Galaxy Note 10.1	1	0	2	2	40	9	1	0	0	0	3	1
Galaxy Note 3	0	0	13	4	63	14	2	0	1	1	5	1
Galaxy S4	1	0	17	3	86	19	3	0	0	1	5	1
Nexus 7 (2013)	0	0	13	2	86	20	2	0	1	0	5	1
Nexus 7	0	0	17	0	83	18	3	1	2	0	5	1
Nexus 5	1	0	15	5	89	21	2	1	1	0	4	1
iPad Air	0	0	10	0	65	15	0	0	0	0	5	0
iPad 4	0	0	20	5	75	20	5	0	0	0	5	0
iPad 3	5	0	30	5	>100	35	5	0	0	0	10	0
iPad Mini 1	5	0	20	5	80	40	0	0	5	0	5	0
iPad Mini 2	0	0	10	5	70	15	0	0	0	0	5	0
iPad 2	0	0	20	5	75	35	0	5	0	0	5	0
iPhone 5S	0	0	10	5	90	35	0	5	0	0	5	0
iPhone 5	0	0	15	10	>100	45	0	0	5	0	5	0

The corresponding battery drop levels for completing the considered common operations for the 3MP and 3MP_50 datasets are presented in Table 5. As expected, there is a strong correlation between battery consumption and the time needed to complete the tests.

By far, the biggest battery drain can be observed for the face detection operation. This is true for all the devices in our tests. Moreover, in the case of the iPad 3 and the iPhone 5 a full battery charge is not sufficient for completing the face detection test for all the images in the 3MP dataset.

4.2 Keypoint Detection/Descriptor Extraction

The keypoint detection and descriptor extraction measurements (corresponding to the phase 2 measurements mentioned above) considered built-in OpenCV algorithms, namely ORB [11], BRIEF [3], BRISK [8], SIFT, SURF, FREAK [1] and FAST [10]. In the case of BRIEF and FREAK we used the GoodFeaturesToTrack algorithm [13] for keypoint detection followed by the descriptor extraction.

In Figure 3 to Figure 6 we have plotted the measurement values corresponding to keypoints extraction as well as descriptor computation for ORB, BRISK, FREAK and FAST while using the 3MP dataset (due to space considerations we do not provide the detailed measurements). The values on the Y axis are in milliseconds. It can be seen that in this case, the measured values for the considered tests on all the devices are similar, except for the iPad Mini 1, iPad 2 and iPad 3. For those 3 devices, the measured values are much higher. Within the Android family, the keypoint detection in the case of ORB clearly outperforms the iOS devices. BRISK shows similar results across all devices with the exception of the iOS devices mentioned earlier. The same is true in the case of FREAK with the observation that the Galaxy Note 10.1 and Nexus 5 provide the fastest results. FAST keypoints detection also takes less on all the Android devices.

In the following we focus on SIFT, SURF as well as BRIEF. The measurements in those cases, also considered the time needed to detect the keypoints as well as the time needed to extract the descriptors based on the detected keypoints. It is important to note that due to patent issues, SIFT and SURF are not included in the official release package of OpenCV for the Android platform, being categorized into the nonfree module. In order to perform the measurements we had to build this nonfree module for Android native projects. One option is to rebuild the whole OpenCV library. We have opted instead for rebuilding only the missing nonfree module[1]. For this we have used the Android NDK, Revision 9c (December 2013).

The exact measured values for the 3MP, 3MP_50 and 3MP_25 datasets are listed in Table 6, Table 7 and Table 8. Although in the case of BRIEF the Android devices achieve better measurement results, for both SIFT and SURF the measured values are much larger than in the case of iOS devices.

[1] We took as reference the instructions from `http://web.guohuiwang.com/technical-notes/sift_surf_opencv_android`

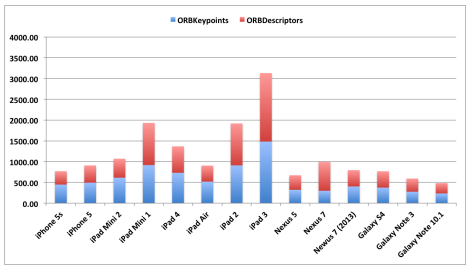

Fig. 3: ORB keypoints and descriptors

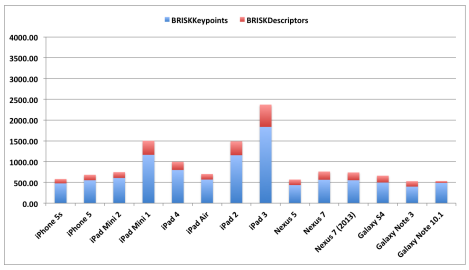

Fig. 4: BRISK keypoints and descriptors

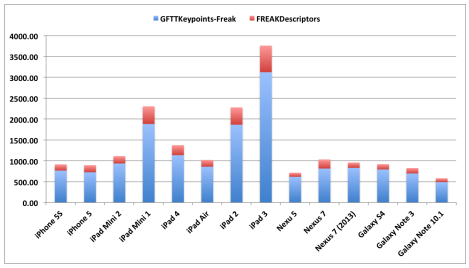

Fig. 5: FREAK keypoints and descriptors

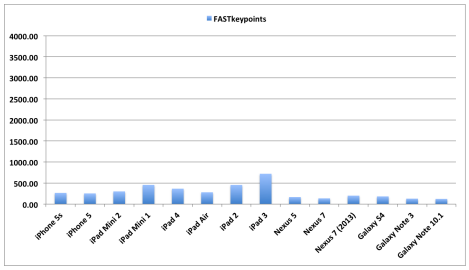

Fig. 6: FAST keypoints (no descriptors computed)

Table 6: Keypoint detection and descriptor extraction (values in ms) - 3MP images

Device	BRIEF		SURF	
	Keypoints	Descriptors	Keypoints	Descriptors
Galaxy Note 10.1	488.78	42.97	82662.90	184723
Galaxy Note 3	677.24	52.20	58601.10	134267.00
Galaxy S4	801.14	75.76	74455.90	173836
Nexus 7 (2013)	824.41	73.78	58248.60	136172.00
Nexus 7	811.50	71.41	45270.00	97972.20
Nexus 5	582.38	48.87	63328.40	148495.00
iPad Air	901.05	54.38	7115.26	20084.18
iPad 4	1134.90	79.62	6552.49	17761.25
iPad 3	3116.00	177.17	12447.22	35201.64
iPad Mini 1	1879.90	126.21	8810.79	22985.91
iPad Mini 2	952.29	56.97	7735.29	21879.85
iPad 2	1866.40	123.42	8586.47	22650.45
iPhone 5S	569.57	36.58	6631.87	18658.79
iPhone 5	729.66	56.50	5120.15	13744.48

Table 7: Keypoint detection and descriptor extraction (values in ms) - 3MP_50 images

Device	BRIEF		SIFT		SURF	
	Keypoints	Descriptors	Keypoints	Descriptors	Keypoints	Descriptors
Galaxy Note 10.1	126.26	16.82	18553.60	20168.90	24220.60	51071.20
Galaxy Note 3	157.51	21.18	21520.80	27542.70	15096.30	33957.30
Galaxy S4	211.50	30.13	26907.70	33548.60	20023.70	46063.10
Nexus 7 (2013)	225.88	33.10	28082.10	33697.40	15718.30	34687.50
Nexus 7	220.16	33.97	38399.30	47780.40	12494.10	27438.70
Nexus 5	180.35	25.91	27057.40	34179.10	18131.50	41822.00
iPad Air	226.52	16.49	3276.46	4780.03	1890.82	5106.70
iPad 4	286.16	25.47	4883.35	6847.57	1846.23	4902.68
iPad 3	756.95	61.63	11560.26	15891.07	3449.07	9811.72
iPad Mini 1	756.18	62.07	–	–	3492.48	9879.97
iPad Mini 2	240.21	17.35	3548.15	5186.26	2236.38	6059.69
iPad 2	751.41	60.96	–	–	3436.40	9795.54
iPhone 5S	218.44	15.75	4209.18	6161.39	2670.63	7242.44
iPhone 5	304.46	26.66	5191.81	7302.17	2085.50	5567.79

Table 8: Keypoint detection and descriptor extraction (values in ms) - 3MP_25 images

Device	BRIEF		SIFT		SURF	
	Keypoints	Descriptors	Keypoints	Descriptors	Keypoints	Descriptors
Galaxy Note 10.1	33.59	7.89	4833.04	5371.13	6201.35	12579.20
Galaxy Note 3	32.55	8.00	4757.97	6533.47	4156.58	8940.18
Galaxy S4	47.74	11.77	6283.44	8221.01	5253.89	11394.90
Nexus 7 (2013)	50.72	12.12	7233.23	9301.98	4043.44	8614.13
Nexus 7	50.33	16.41	9445.54	12471.60	3250.60	6820.61
Nexus 5	39.55	9.75	5940.75	8033.79	4888.02	10580.00
iPad Air	58.52	5.76	914.51	1407.81	587.21	1447.07
iPad 4	76.20	9.38	1185.85	1767.18	564.72	1360.12
iPad 3	187.33	26.09	2874.33	4191.65	905.31	2406.50
iPad Mini 1	186.45	26.56	2955.26	4310.72	919.02	2428.89
iPad Mini 2	61.80	6.09	956.89	1475.02	623.83	1540.25
iPad 2	186.10	25.96	2888.90	4224.87	899.57	2397.98
iPhone 5S	55.33	5.43	1127.66	1742.89	677.56	1674.53
iPhone 5	80.94	9.82	1300.93	1933.76	514.59	1267.40

Table 9: Battery drop levels (values in %) for the 3MP (D1), 3MP_50 (D2) and 3MP_25 (D3) datasets for keypoint detection and descriptor extraction

Device	BRIEF			SIFT			SURF		
	D1	D2	D3	D1	D2	D3	D1	D2	D3
Galaxy Note 10.1	9	2	1	–	55	15	>100	>100	30
Galaxy Note 3	15	3	1	–	93	19	>100	98	24
Galaxy S4	18	5	2	–	>100	25	>100	>100	33
Nexus 7 (2013)	15	4	1	–	94	18	>100	>100	33
Nexus 7	20	6	1	–	>100	35	>100	>100	26
Nexus 5	15	4	1	–	>100	25	>100	>100	34
iPad Air	5	5	0	–	60	15	>100	55	15
iPad 4	10	5	0	–	80	20	>100	45	15
iPad 3	25	10	0	–	>100	45	>100	100	25
iPad Mini 1	10	5	5	–	–	55	>100	>100	25
iPad Mini 2	10	0	0	–	70	15	>100	60	15
iPad 2	15	5	0	–	–	50	>100	100	25
iPhone 5S	10	5	0	–	>100	35	>100	>100	35
iPhone 5	10	5	0	–	25	40	>100	>100	30

For the 3MP dataset, the SIFT measurements could not be performed on any of the devices because the test applications crashed on both Android and iOS with an insufficient memory error message (therefore they are omitted from Table 6). This also happened in the case of the 3MP_50 dataset for the iPad Mini 1 and iPad2. The corresponding values are marked as – in Table 7 and Table 9.

The battery drop information in the case of BRIEF, SIFT and SURF for the 3 considered datasets are provided in Table 9. Please note that for the 3MP dataset, the SURF measurements needed more than one full charge to complete on all tested devices.

5 Conclusion

In this paper we showed results for performance measurements of common OpenCV functions executed on Android and iOS mobile devices. Our measurements were grouped into common operations, keypoint detection and descriptor extraction. The obtained results show that most modern mobile devices can achieve good and in some cases excellent results for the considered operations, meaning that powerful computer vision applications will be possible on such devices in the near future. Overall, better performance was achieved with Android devices. We speculate that this is mainly due to the more powerful hardware. We did not further investigate this issue since this does not constitute the main focus of our work. Another aspect that might have impacted the results, but on a smaller scale, is the difference in compiler implementations used for building the OpenCV library for the considered architectures. Future work will concentrate on parallel and GPU assisted computations for the OpenCV operations on Android, iOS and Windows platforms.

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 by Lakeside Labs GmbH, Klagenfurt, Austria, and funding from the European Regional Development Fund (ERDF) and the Carinthian Economic Promotion Fund (KWF) under grant KWF-20214/22573/33955.

References

1. 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.
2. 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, 2006.
3. 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, 2010.
4. E. Chatzilari, G. Liaros, S. Nikolopoulos and Y. Kompatsiaris A comparative study on mobile visual recognition In *9th Int. Conf. on Machine Learning and Data Mining (MLDM 2013)*. New York, USA, July 19-25, 2013
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 Anniv. Int. Conf. on MultiMedia, Part II - MMM 2014*, volume 8326 of *Lecture Notes in Computer Science*, pages 402–406, 2014.
6. B. Girod, V. Chandrasekhar, D. M. Chen, N.-M. Cheung, R. Grzeszczuk, Y. A. Reznik, G. Takacs, S. S. Tsai, and R. Vedantham. Mobile visual search. In *IEEE Signal Processing Magazine*, 2011
7. M. A. Hudelist, C. Cobârzan, , and K. Schoeffmann. Opencv performance measurements on mobile devices. In *Proc. of Int. Conf. on Multimedia Retrieval ICMR 2014, April 01 - 04 2014, Glasgow, United Kingdom*, pages 479–482, 2014.
8. S. Leutenegger, M. Chli, and R. Siegwart. Brisk: Binary robust invariant scalable keypoints. In *IEEE Int. Conf. on Computer Vision (ICCV)*, pages 2548–2555, 2011.
9. D. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110, 2004.
10. 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, 2006.
11. 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.
12. 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.
13. J. Shi and C. Tomasi. Good features to track. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 593–600, 1994.