EUROGRAPHICS '0x / N.N. and N.N.
(Editors)

Volume 0 (2011), Number 0

Mobile Computational Photography: Exposure Fusion on the Nokia N900

Jose I. Echevarria and Diego Gutierrez

Universidad de Zaragoza, Spain

Abstract

This paper explores the Nokia N900 platform as a versatile playground for mobile computational photography. We present the implementation of a simple yet practical application to acquire high quality tone mapped HDR images, and a relatively straightforward white balance selection. We have built everything from scratch: from the capture of the bracketed sequence of images with different exposures, to the final tone mapping algorithm. Completed in less than one week, showcasing the flexibility and productivity of the platform, our experiments have encouraged us to continue exploring all the possibilities this kind of devices offer.

1. Introduction

Computational photography is an exciting and hot new field of research. In fact, forty percent of the 439 papers submitted to SIGGRAPH 2009 had to do with 2D imaging techniques applied to photography or video [Lev10]. The overall goal of the field is to overcome the limitations of traditional image capture methods. By encoding the light reaching the sensor in smart ways, it is possible to later decode it and extract information that would have been lost otherwise. Thus, the information stored at the sensor is usually *not* the final image, but an encoded representation of the scene: a computational process needs to be applied afterwards, before the final image is ready for consumption. Examples of recent advances include extended dynamic range, deblurring out-of-focus images, digital refocusing or light field capture. These examples show how photography can be taken to the next level, and how consumer cameras could benefit from them, offering improved versions to the market.

A few years ago, the estimated number of digital cameras in the world broke the one *billion* mark. This is mainly due to the ubiquitous presence of mobile devices (mainly cell phones) with a built-in camera, which in turn has triggered an increasing number of photoblogs and online picture collections. Images can not only be taken anywhere, anytime and by anybody now; they can also be almost instantaneously shared, duplicated and manipulated. This poses the challenge of how to make computational techniques useful for this kind of mobile devices, which are hampered by several obvious limitations, both in terms of hardware (limited extensible lenses, reduced sensor size...) and software (less computational power).

One key limitation that was slowing progress down was the impossibility to access all the camera features, sometimes controlled by firmware and out of reach for the user (or programmer). It was only a few months ago that a team led by Stanford and Nokia Research [AJD*10] released an API to make cameras fully programmable, allowing researchers to come up with new, customizable algorithms. The new open platform is called the Frankencamera, and the authors offered implementations for two platforms: the F2 (built from off-the-shelf components), and the Nokia N900 smartphones.

We have reached a collaboration agreement with Nokia Research Center, who kindly donated seven N900 smartphones for teaching and research purposes. We have so far used them both in undergrad and graduate computer graphics and computational photography courses; despite the short period of time, initial results are very promising, and students were able to complete simple assignments in less than a week. In this paper, we focus on one such assignment: programming the complete high dynamic range imaging pipeline on the N900, from multi-bracketed capture of low dynamic range images, to final exposure fusion. Our implementation produces better results in general than the *HDR Capture* app provided by Nokia Beta Labs, reproducing more detail in over- and under-exposed areas. Finally,

we leverage the versatility of the platform to program a very simple white-balancing algorithm that cycles through different settings, in order to offer the user multiple depictions of the captured scene.

2. Previous Work

Mobile computational photography is a quite recent research field, so little literature can be found yet. However, there are already good examples of some of its goals and applications.

The Frankencamera project [AJD*10] is the seminal work for the field. The goal was to provide the community a base hardware specification and an API for C++, to make cameras fully programmable. Its architecture permits control and synchronization of the sensor and the image processing pipeline. It also offers support for external hardware like lenses, flashes, accelerometers, etc. A more detailed overview of it can be found in Section 3. Along with the technical specifications, they also showcase the platform with applications like HDR viewfinding and capture, low-light viewfinding and capture, automated acquisition of extended-dynamic-range panoramas, foveal imaging, gyroscope-based hand-shake detection and rephotography.

During the gestation of Frankencamera, other interesting experiments were performed, mainly following the same approach as in this paper: students were given the platform, and assigned a small task [Lev10]. For instance, 4D Light fields capture and display was made possible by waving a cell phone around an object. The captured light field can then be displayed in the same or another phone, computing its spatial location and orientation based on a card with markers, and displaying the appropriate slice of the light field on the viewfinder. A virtual rearview mirror with no blind spots was also built by combining the video streams from five Nokia N95 phones mounted facing backward on the roof of a car.

Real-time viewfinder alignment [AGP08] is another example of algorithms that can be useful for a wide variety of mobile applications, from assistance in panorama capture, to low-light photography and camera-based controller input for games. Finally, focusing on external components apart from camera sensors, intelligent use of flash can provide favorable light conditions to take photos in and interesting illumination effects, as Adelsberger and colleagues demonstrated [AZG08]. They presented a spatially adaptive photographic flash, in which the intensity of illumination is modified on a per-pixel basis depending on the depth and reflectivity of features in the scene. Their results show images that are better illuminated than the original scene, still looking natural and plausible.

3. The Frankencamera architecture

The Frankencamera [AJD*10] is both an open source architecture and an API for cameras. It includes a base hardware specification, a Linux-based operating system and a C++ API. It is intended to be flexible enough to implement most of the techniques proposed in the computational photography literature. So, it presents a comprehensive list of features: the factor form and usability of consumer level cameras, touchscreen for designing new user interfaces, easy to program for with standard libraries, low-level access to principal components affecting the capture process on a per-frame basis, access to raw sensor pixels, acceptable processing power and compatibility and upgradability with standard and experimental camera accessories, to name a few.

The abstract architecture encompasses the image sensor, the fixed-function imaging pipeline that obtains and produces image data and optional photographic devices such as lens, flash, etc. Figure 1 shows a diagram explaining how these elements interact. The main characteristic of the image sensor is its stateless behavior, meaning that capture parameters must be set on a per-frame basis. The imaging processor has two main roles: to generate useful statistics like histograms and sharpness maps that are attached to each generated frame; to perform basic operations as demosaicking, white-balancing, gamma correction, etc. Additional devices can be connected to the platform in order to expand the information obtained by the sensor [AJD*10].

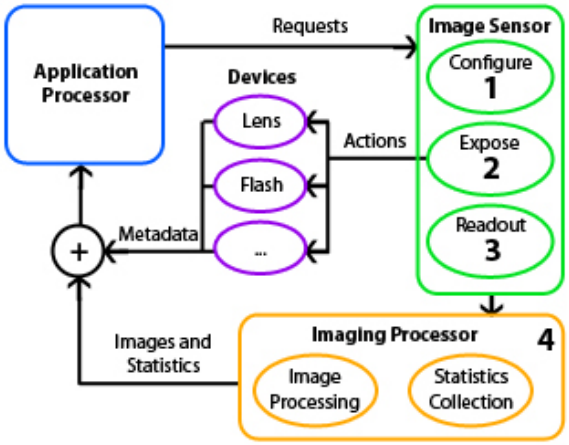

Figure 1: *Overview of the Frankencamera architecture (adapted from [AJD*10]). The image sensor produces captures according to the requested parameters by the application. Optionally, external devices can be brought in during the exposure, altering the capture and providing additional useful info. Captures are then transformed into images by the imaging processor, which will be finally used by the application processor for further tasks.*

Two platforms were constructed following the enumerated principles: the F2 camera, built from different off-the-shelf components; and the Nokia N900 smartphone. In this work we focus on our available version, which is the N900 smartphone.

3.1. The Nokia N900

The Nokia N900 is a smartphone provided with an ARM CPU running at 600 MHz, 256 MB of RAM plus 768 MB of virtual memory and a 5 MP camera with Carl Zeiss Optics, all of them governed by a *Maemo* Linux-based operating system, which makes the platform instantly compatible with tons of standard libraries for Unix development. Figure 2 shows some views of the real model.

To start developing for the N900, Nokia provides their Qt Creator SDK, an IDE to easily connect with the phone, which integrates the compiler and libraries needed to quickly start developing programs for the platform. Some additional setup needs to be performed on the device (firmware updates, drivers...), before it is finally ready to make the most of the unit through the FCam API. This API provides low level access to most of the parameters of N900's sensor, lens and flash. With the FCam installed, the N900 turns into an affordable playground for computational photography.

4. HDR Acquisition

Our main goal is to study the Nokia N900 smartphone along with the FCam API as a platform for mobile computational photography. For this project, we have focused on the full pipeline of automatic acquisition of high-dynamic range (HDR) images, including multi-bracketing, exposure fusion and tone mapping. The objective is to have the N900 performing *all* the calculations without any user intervention, before presenting the final image. Note that a similar example application is presented in the original Frankencamera paper [AJD*10]. However, as we will see, we propose two different scene-independent mechanisms to choose the three input exposures used to create the HDR image, which in general yield better results than Nokia's own implementation.

Given the low dynamic range sensor of the N900, we start from a multi-exposure bracketed sequence of low dynamic range images. The usual next step would be to create an intermediate physically-based HDR image [DM97] which would be finally tone mapped [DD02, RSSF02, LSA05]. However, keeping in mind our goals and the N900 specifications, we opt for using a more computational affordable but still high-quality solution: *Exposure Fusion* [MKVR09].

Figure 2: *Front and rear views of the Nokia N900 smartphone.*

The original exposure fusion technique fuses a bracketed exposure sequence of low dynamic range images directly into a high quality low range tone mapped image. By skipping the intermediate HDR assembly it simplifies the acquisition process and camera calibration, still producing results comparable to widely used tone mapping operators. First, it measures the quality of each pixel from each image of the sequence according to their contrast, saturation and *well-exposedness*. Then it makes a weighted sum of all the images, using the corresponding weight maps, for obtaining the final result. In order to avoid artifacts due to this blending, each color image is transformed into a Laplacian Pyramid, and each weight map is transformed into a Gaussian Pyramid. The weighted sum is performed at each level of the pyramids, fusing them into a single one, which is finally collapsed to obtain the final full resolution image. In order to maximize the ratio between quality and required computational power, we choose to implement a simplified version of the algorithm, leveraging only well-exposedness information and simply ignoring contrast and saturation. Again, we have found that this not only streamlines the process, but it tends to yield better results than the implementation by Nokia Labs.

5. Implementation

First of all, we need to obtain the bracketed sequence of images with appropriate exposure values. In the original Frankencamera implementation, the authors propose to adjust the exposure times based on an analysis of the scene content, following the work by Kang et al [KUWS03].

In this work we aim to make this process scene-independent, while providing good results in most cases, so we follow a different route. We rely on the fact that on the N900, we can set the exposure time to any length we want (in microseconds) inside the supported range of the device. However, in digital photography it is more standard to work with EV (Exposure Value) stops [JRAA00], which depend on exposure time and aperture size. Given that the N900 aperture is fixed at $F/2.8$, we can obtain the range of N900 EVs by setting up exposure time values as in Table 1.

EV	4	6	8	10	12
Exposure time (s)	1/2	1/8	1/30	1/125	1/500

Table 1: *Exposure times for the selected EV values that tipically maximize the N900's sensor range, given its* $F/2.8$ *aperture. N900's Minimum EV is* 2 *and maximum is* 16.

For capturing input images with better quality, we have also added auto focus and white balance to the acquisition process. Source code for these tasks can be found in the documentation examples of the FCam API.

To create the final HDR image from the bracketed sequence, we use *Exposure Fusion* [MKVR09], which can blend together an arbitrary number of images. However, one important limitation of general HDR acquisition techniques is the fact that both the camera and the scene need to be static, to avoid ghosting artifacts. This suggests that finding the *minimum* number of necessary low-dynamic images is desirable, to reduce the total acquisition time. In our experiments, we found that nice results can be obtained with just three images (in accordance to the proposed method described in [AJD*10]). Instead of letting an analysis of scene content fix the three exposure times, we capture images with the five EVs that usually maximize the valid range of the sensor (EVs 4, 6, 8, 10 and 12). We then select the three resulting images containing the largest total number of useful pixel values, avoiding the two images with more over- or under-exposed pixel values (note that these two are not necessarily the lowest and highest EVs respectively; the discarded images will be a function of the scene original lighting level).

As an alternative, we have implemented another fully automated option, which makes use of the built-in metering functions inside the FCam API. Along with the auto focus and white balance, we additionally use the *auto exposure* to let the camera deduce what would the optimal exposure be, if just a single LDR image were to be taken. Then, we move two EV stops back and forth from that value (two steps proven to be the better choice in our tests). This way we have to capture just three images, lowering capture times and making HDR imaging obtention even more straightforward.

As outlined before, the original work proposes three different quality measures for each pixel: contrast, saturation and *well-exposedness*. As the authors discuss in their paper, the most influential measure seems to be the last one, while the other two add really subtle differences most of the times. So, due to our limited processing power, we use only the *well-exposedness* measure, which accounts for pixels that are not under nor overexposed, based on how close to 0.5 is the intensity of the pixel using a Gauss curve:

$$W_{ij,k} = \prod_c \exp\left(-\frac{(I_{ijc,k} - 0.5)^2}{2\sigma^2}\right) \tag{1}$$

where $I_{ijc,k}$ corresponds to the intensity of the pixel ij for each channel c, and σ equals to 0.2. This weights are calculated for each image k, and then normalized so $\sum_{k=1}^{3} \hat{W}_{ij,k} = 1$, with $\hat{W}_{ij,k}$ being the normalized weights.

At this point, if we make a naive weighted sum of all the images, artifacts will appear due to the different absolute intensities. So, before the blending, a Laplacian pyramid is built for the color images, $L\{I\}$, and a Gaussian one for the weights maps, $G\{\hat{W}\}$. Now, the blending can performed at each level l, for each channel c:

$$L\{R\}^l_{ijc} = \sum_{k=1}^{3} G\{\hat{W}\}^l_{ijc,k} L\{I\}^l_{ijc,k} \tag{2}$$

The multi resolution approach provided by the Laplacian pyramid helps smoothing sharp discontinuities produced by Equation 2 during the collapse of $L\{R\}$ in order to obtain the final full resolution image. For building and collapsing the pyramids [BEA83], we use the separable convolution kernel $f = [0.0625, 0.25, 0.375, 0.25, 0.0625]$ [MKVR09].

6. Results and Discussion

All of our examples have been obtained by blending a bracketed sequence of three images with different EVs. In order to make computing times acceptable for taking photos on the go, we capture the images at 640*x*480 resolution. The three images are taken in 2*s* (5*s* for the sequence of five images), weights calculation (Equation 1) take 5*s*, and pyramids are built, mixed (Equation 2) and collapsed in 11*s*. Our pyramids have 8 levels, the maximum number of levels for the images resolution. As they have the biggest impact on the processing time of our application, we have tested less levels for sparing calculations. Unfortunately the results always show artifacts like halos an color shifts.

Figure 3 shows two challenging examples (as the low dynamic range insets show) where our application produces natural looking results. Figure 4 shows a comparison between two captures with our application (fully automatic version) and the results obtained via Nokia Beta Labs *HDR Capture* application [AJD*10], with automatic exposure enabled and forced to use three photos as input. We can see how, although built with the same tools, the flexibility this FCam API provides can make results vary significantly depending on its use.

Limitations of our application come in form of slightly loss of contrast and saturation due to the exclusive use of the *well-exposedness* quality measure, as explained in Section 5. However, we could directly introduce them, with a higher execution time for the complete process.

Just right now, there are already other commercial platforms on the market that are capable of simulate this kind of applications [App10, Goo10]. The difference is that those platforms just provide limited high-level access to some of the camera functions, thus making the exploration of other techniques difficult. In contrast, the N900 in conjunction with the FCam API offers direct low-level access to parameters that provide the flexibility and power needed for real mobile computational photography research.

There are virtually infinite applications that can be devised and implemented on the N900. Another simple exam-

Figure 3: *Two examples of photos captured and processed by our method. The insets show the bracketed sequence input. The complete pipeline takes less than* 20*s for a resolution of* 640*x*480*. Left: auto exposure. Right: manual selection of three EVs out of five.*

Figure 4: *Results obtained with the N900 using our application (left) and* HDR Capture *app (right) by Nokia Beta Labs. Insets show the images used to build the final one. Examples taken with auto exposure in both applications. We can see how our application produces a more detailed and natural look.*

ple application is shown in Figure 5, where the camera cycles through seven white balance settings and offers the user a combined mosaic of the results. The user simply chooses the one that best depicts the captured scene, and the system keeps that one and (optionally) deletes the rest. In this case, the exposure time has been set at 50ms, with a fixed gain of 1.0. The white balance range goes from 2800K to 11000K.

7. Conclusions and Future Work

We have demonstrated how the Nokia N900, via the FCam API, can be used as a powerful and flexible platform for mobile computational photography. We have shown an efficient implementation of the full HDR pipeline, from the automatic acquisition of a bracketed sequence of images to the final fusion of the multiple exposures. We have obtained good results by just modifying exposure times of the sensor and adding an efficient algorithm for image processing. These results most of the times surpass Nokia's own implementation. Future work will include exploration of concepts and techniques that make use of more components of the mobile phone like lenses, flash, etc.

This project has been successfully completed in less than one week, from initial problem description to final implementation, showing the power and flexibility of the platform. We believe there is a great deal of future applications that

Figure 5: *From left to right, up to down: sequence of images obtained by capturing the same scene with white balance ranging from* 2800*K to* 11000*K. Bottom right shows the selection image presented to the user in order to choose the preferred color temperature.*

will find their way into mobile devices, thanks to the release of the FCam API. Finally, we plan to develop a full computational photography course at graduate level, leveraging Nokia's recent hardware donations to our group.

8. Acknowledgements

We thank Kari Pulli and the Nokia Research Center at Palo Alto for the donation of seven N900's and their financial support. We also thank Javier Delgado for the white balance application. This research has been funded by a Marie Curie grant from the Seventh Framework Programme (grant agreement no.: 251415), the Spanish Ministry of Science and Technology (TIN2010-21543) and the Gobierno de Aragón (projects OTRI 2009/0411 and CTPP05/09).

References

[AGP08] ADAMS A., GELFAND N., PULLI K.: Viewfinder alignment. *Computer Graphics Forum 27*, 2 (2008), 597–606. 2

[AJD*10] ADAMS A., JACOBS D. E., DOLSON J., TICO M., PULLI K., TALVALA E.-V., AJDIN B., VAQUERO D., LENSCH H. P. A., HOROWITZ M., PARK S. H., GELFAND N., BAEK J., MATUSIK W., LEVOY M.: The frankencamera: an experimental platform for computational photography. *ACM Trans. Graph. 29* (July 2010), 29:1–29:12. 1, 2, 3, 4

[App10] APPLE: ios 4.1 update. http://arstechnica.com/apple/news/2010/09/hdr-photography-with-iphone-4-and-ios-41.ars, September 2010. 4

[AZG08] ADELSBERGER R., ZIEGLER R., GROSS M.: Spatially adaptive photographic flash, 2008. 2

[BEA83] BURT P. J., EDWARD, ADELSON E. H.: The laplacian pyramid as a compact image code. *IEEE Transactions on Communications 31* (1983), 532–540. 4

[DD02] DURAND F., DORSEY J.: Fast bilateral filtering for the display of high-dynamic-range images. *ACM Trans. Graph. 21* (July 2002), 257–266. 3

[DM97] DEBEVEC P. E., MALIK J.: Recovering high dynamic range radiance maps from photographs. In *Proceedings of the 24th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 1997), SIGGRAPH '97, ACM Press/Addison-Wesley Publishing Co., pp. 369–378. 3

[Goo10] GOOGLE: Android 2.2 update. http://developer.android.com/sdk/android-2.2-highlights.html, September 2010. 4

[JRAA00] JACOBSON R. E., RAY S. F., ATTERIDGE G. G., AXFORD N. R.: *The Manual of Photography: Photographic and Digital Imaging*, 9th ed. Focal Press, 2000. 3

[KUWS03] KANG S. B., UYTTENDAELE M., WINDER S., SZELISKI R.: High dynamic range video. *ACM Trans. Graph. 22* (July 2003), 319–325. 3

[Lev10] LEVOY M.: Experimental platforms for computational photography. *IEEE Computer Graphics and Applications 30* (2010), 81–87. 1, 2

[LSA05] LI Y., SHARAN L., ADELSON E. H.: Compressing and companding high dynamic range images with subband architectures. *ACM Trans. Graph. 24* (July 2005), 836–844. 3

[MKVR09] MERTENS T., KAUTZ J., VAN REETH F.: Exposure fusion: A simple and practical alternative to high dynamic range photography. *Computer Graphics Forum 28*, 1 (2009), 161–171. 3, 4

[RSSF02] REINHARD E., STARK M., SHIRLEY P., FERWERDA J.: Photographic tone reproduction for digital images. *ACM Trans. Graph. 21* (July 2002), 267–276. 3