

# Technical Perspective

## Open Platforms for Computational Photography

By Richard Szeliski

COMPUTATIONAL PHOTOGRAPHY IS AN emerging discipline that enables the creation of enhanced-quality photographs through novel combinations of digital images, algorithms, optics, and sensors.<sup>2,5</sup> The field lies at the intersection of image processing, computer vision, and computer graphics, and has spawned its own workshops and conferences. It has also engendered many new features used in digital cameras and smartphones.

While scientists have applied image analysis and enhancement techniques to images for decades, the application of sophisticated algorithms to consumer photography started in the mid-1990s. Early examples of such algorithms include stitching multiple images into seamless panoramas, merging multiple exposures to create and display high dynamic range (HDR) images, and combining flash and no-flash images to provide better details in dark regions without harsh shadows.

As with most of computing, computational photography algorithms were originally developed and deployed on professional workstations and desktop personal computers. Unfortunately, the inability to deploy these algorithms inside cameras has severely limited real-world experimental validation and the percolation of these scientific advances into consumer products.

The migration of these algorithms into hardware and firmware has been hampered by a number of factors.<sup>1</sup> For example, digital image processing algorithms used by cameras are protected by patents and trade secrets. Vendors also tightly control the user experience, rather than taking the more open approach embraced by the app development community.

An even more fundamental impediment to the widespread development and deployment of in-camera algorithms is the lack of a clean open architecture for controlling camera features and writing the correspond-

ing real-time processing and viewing algorithms. The following article by Adams et al. is the first to address this problem, and it does so in a beautiful and elegant fashion.

The need for real-time processing and immediate feedback requires cameras to perform many different tasks in parallel. For example, cameras need to determine the optimal exposure time, aperture, analog gain, and focus settings for each picture.

Coming up with an elegant, programmable architecture and the APIs that support the deployment of sophisticated computational photography algorithms is a challenging architectural design problem. The authors show that in order to achieve this, the architecture must allow the specification of parameter sets (called shots) that control (or suggest) how individual images should be taken.


Because setting up these parameters can take time, the architecture keeps the desired and actual parameters tightly coupled with raw (unprocessed) images returned to the image processor. The complete architecture proposed in the paper therefore consists of shots (desired parameter sets), sensors that capture either individual, burst, or continuous streams of shots, frames that return the captured images and metadata, and devices such as lenses and flash units that can be controlled by the program.

To demonstrate the utility and generality of their approach, the authors built a custom-made experimental Frankencamera from commercial imaging parts and also reprogrammed an existing Nokia N900 smartphone. They then developed a collection of useful and compelling computational photography algorithms.

Since its original publication at SIGGRAPH 2010, the Frankencamera paper and associated hardware/firmware systems have had a dramatic impact on computational photogra-

phy research and teaching, as well as consumer-level photography devices. The Frankencamera devices and software have been used in the Stanford CS 448A course on Computational Photography<sup>4</sup> as well as computational photography courses at other universities. Numerous computational photography apps can now be found for smartphones, and ideas inspired by the paper are also being incorporated into upcoming versions of smartphone operating systems and libraries.

One additional ingredient needed to make computational photography algorithms easy to develop is a high-level language and compiler tailored to such programs. Fortunately, a SIGGRAPH 2012 paper describing a system called *Halide* promises to do just that by enabling programmers to write high-level array-like descriptions of algorithms and then giving hints to the compiler about the desired levels of tile-based caching, parallelism, pipelining and reuse.<sup>3</sup>

Computational photography is blossoming as both a research field and a vibrant application area affecting all aspects of digital photography. The following paper provides an elegant example of how well-designed architectures in computer science can facilitate and accelerate the adoption of new technologies and expose novel capabilities to new generations of students. 

### References

1. Levoy, M. Experimental platforms for computational photography. *IEEE Computer Graphics and Applications* 30, 5 (2010), 81–87.
2. Nayar, S. K. Computational cameras: Redefining the image. *Computer* 39, 8, (2006), 30–38.
3. Ragan-Kelley, J., Adams, A., Paris, S., Levoy, M., and Amarasinghe, and Durand, F. Decoupling algorithms from schedules for easy optimization of image processing pipelines. *ACM Transactions on Graphics* 31, 4 (2012).
4. Stanford CS 448A: Computational Photography; <http://graphics.stanford.edu/courses/cs448a-10/>.
5. Szeliski, R. *Computer Vision—Algorithms and Applications*. Springer, 2010.

Richard Szeliski (szeliski@microsoft.com) is a distinguished scientist at Microsoft Research, Redmond, WA.

© 2012 ACM 0001-0782/12/11 \$15.00