For generations, people have taken pictures and videos to preserve memories for personal reflection as well as for sharing experiences with others. As a result of the convenience, ease of use, and quality improvements in smartphone cameras, many people now opt to use a smartphone instead of a digital camera for multimedia capture. Sales statistics confirm this trend as global shipments of point-and-shoot cameras were down 42 percent in the first five months of 2013. Advances in photo optics and digital imaging are making it possible for smartphones to embed high-quality digital cameras with the added bonus of connectivity, value, and convenience. In addition, many smartphone devices now have both front-facing and rear-facing cameras, which further highlights the pivotal part that the smartphone camera plays in smartphone design and usage.

This article centers on the world of mobile media processing, looking through the “lens” of the smartphone camera. Smartphone camera quality has continually improved over the last few years, but mobile media processing can still provide significant enhancements. This article focuses on technologies within mobile media processing, including include color and contrast enhancement, backlight image compensation, deblurring, denoising, and video stabilization, that improve smartphone camera quality with the goal of enhancing the user’s imaging experience.

The Art of Mobile Media Processing

Many smartphones can do a good job of delivering high-quality photos under normal lighting. However, the quality can noticeably deteriorate under other lighting conditions. Because of this, picture quality under low-light conditions has become one of the major evaluation criteria of smartphone cameras. Under low-light conditions, a smartphone camera picture degrades mainly because of three factors: reduced color dynamic range, distorted color distribution, and excessive sensor noise. Contrast and color enhancement and denoising can help reduce picture degradation resulting from these effects.

Many features, such as auto focus, auto exposure, and different scene modes, have been developed to help people take beautiful pictures. However, backlight images are still common because of camera quality, lack of skill in using available photography techniques, and differing photo environments. In cases where there is a strong light in the background environment, the key foreground objects, such as faces, are often under exposed, appearing too dark. Because many image editing tools and advanced cameras require user intervention to fix the problem, an automatic method of backlight compensation would be preferable.

When taking image and video with a smartphone camera, many users also have a tough time holding the camera steady. Camera movement causes blurred images and shakiness within video. To correct these issues and improve the image and video quality, deblurring and video stabilization processes can be performed and optimized for the mobile platform.

A set of key mobile media processing techniques can be introduced to mitigate such quality problems. The art of mobile media processing technologies lies within the capability of enhancing digital imaging sensing via postprocessing to deliver picture-perfect photos to nonprofessional consumers. Figure 1 illustrates several crucial mobile visual media processing components, which we will discuss here, to achieve that goal.

Color and Contrast Enhancement

As we mentioned earlier, a smartphone camera picture degrades under low-light conditions as a result of reduced color dynamic range, distorted color distribution, and excessive sensor noise.
The human visual system is often modeled to develop color and contrast enhancement algorithms in order to compensate for the visual degradation caused by the imaging sensor. The ACE algorithm is one example of such an algorithm.\(^2\)

Given a color image, the ACE algorithm computes an ACE value, \(ACE(x)\), through adapting local contrast by computing intensity differences, performing weighting based on distance, and utilizing a slope function to adjust the dynamic range. The local adaptation step is followed by a global adaptation, which performs a dynamic tone reproduction scaling to produce the final adjusted intensity values. An improved ACE algorithm has been developed that consists of eight steps:

1. An optional tiling mode is performed during which an image is partitioned into overlapping tiles, and each tile is processed in parallel from steps 2 through 7.

2. The image is downsampled into a lower resolution, and the scaling factor is determined based on a threshold dependent on the desired processing speed and image size.

3. The image is transformed into the HSV color space.

4. For the V channel in the HSV color space, the ACE values are calculated using local adaptation for each pixel.

5. The ACE values are extended to the full range of \([0, 1]\), and gamma correction is applied.

6. The enhanced image is transformed back to its original color space.

7. The enhanced image is upsampled to its original resolution by utilizing the joint bilateral upsampling algorithm.

8. If the tiling mode is enabled in step 1, all enhanced tiles are fused together.

The first improvement to the ACE algorithm involves revisiting the gray-world assumption, which assumes that the mean of the intensity values for an image lies at the midpoint of the dynamic range. We propose an adaptive local gray-world assumption to take into account the image context that reflects the lighting condition.

To realize this, two methods can be used: linear correction and gamma correction. Both methods extend the ACE values to the full range, calculate the current mean intensity value \(m'\), and adjust the mean intensity value toward \(m^*\), where \(m^* = (1 + 2 \cdot \text{mean}(I))/4\) and \(\text{mean}(I)\) is the mean intensity value. Both scale the ACE enhanced intensity values toward 0 or 1, depending on whether \(m^*\) is larger than \(m'\), to produce the final enhanced intensity values. In general, visual results have shown that the gamma correction produces a better overall enhanced contrast when compared with the linear correction.

To further reduce the computational complexity, we employ joint bilateral upsampling, the HSV color space, and an optional tiling mode. When applying joint bilateral upsampling, denoising is performed simultaneously. Using the HSV color space, the brightness and contrast of low-light pictures are greatly enhanced, while the color balancing is well maintained along with only processing one channel versus three channels for RGB. The tiling mode partitions the image into overlapping tiles and processes each tile in parallel.

Subjective tests have validated the visual quality enhancement of our improved ACE algorithm over the (standard) ACE algorithm. Figure 2 shows one example. Figure 2a shows a smartphone camera picture taken under a low-light condition. The ACE algorithm’s approximate solution\(^3\) and the resulting enhanced images is shown in Figure 2b. Lastly, Figure 2c shows the results of our improved ACE algorithm, which produces a vivid color balancing with a better reduction in the noise level, resulting in better overall visual quality. In addition
to visual results, we performed computational complexity tests and found a speed improvement on the order of one magnitude.

**Backlight**

We propose a method of content-aware adaptive backlight image compensation that analyzes two aspects of the image. One is determining the backlight level based on a gray-world method. Another is detecting the image as a portrait image or scenic image based on face detection. After determining the backlight level, the exposure compensation function is utilized to make a fully automatic, parameter-free adjustment. Based on the statistical analysis of large sets of backlight images, one observation is that for scenic pictures, or scenic backgrounds in portrait pictures, adjusting the V channel in HSV color space can result in more vivid colors. For face regions in portrait pictures, the HSV color space has more distortions. However, adjusting the luminance in the YCrCb color space is more natural. In this method, image compensation is jointly performed with adaptive weights on the HSV and YCrCb color spaces. The weights depend on whether the image’s main scene is portrait or scenic.

The first step is to obtain the exposure-compensation level. To begin, the image color space is transformed into the YCrCb color space. An exposure-density function can be used to simulate the relationship between the pixel luminance value and incoming light intensity or exposure value $S$:

$$I = f(s) = \frac{255}{1 + e^{-A_s^3}}$$

where $I$ is the pixel luminance value in the YCrCb color space, $S$ is the exposure, and $A$ is the contrast level parameter $0.85$ ($0.85–1.20$).

The key idea behind the exposure-correction method is to adjust the average exposure of the area of interest toward the ideal exposure, which corresponds to a gray level of 128. In this process, we first obtain the average low-light pixel luminance value (for luminance values smaller than 128) in the YCrCb color space. Then, the difference between the idea exposure and the average exposure for the low-light regions is calculated:

$$\text{Difference} = f^{-1}(128) - f^{-1}(\text{AvgLowLight})$$

In the next step, we perform pixel-wise compensation in the YCrCb color space by only compensating for the pixels with a $Y$ value that is smaller than 128:

$$\text{Difference'} = \text{Difference} \times \frac{128 - I}{128}$$

Then, we re-expose it as follows:

$$S' = f^{-1}(I) + \text{Difference'}$$

Then, the new pixel luminance value is calculated in the YCrCb color space based on the density-exposure function:

$$I' = f(S')$$

---

**Figure 2. Improved ACE algorithm results.** (a) Original image, (b) ACE approximate solution, and (c) our improved ACE.
For pixel-wise exposure compensation in the HSV color space, the following steps are performed. Based on Retinex theory, the illumination component is estimated by Gaussian blurring in the RGB color space. Then, the reflectance component $R(x, y)$ is calculated as follows:

$$R(x, y) = \frac{I(x, y)}{L(x, y)}$$

Color cast is removed from the illumination component by stretching out R, G, B to [0, 255] to get the new illumination component $L_0(x, y)$ as follows:

$$r_0(x, y) = \frac{r(x, y)}{\max(r(x, y))}$$

$$g_0(x, y) = \frac{g(x, y)}{\max(g(x, y))}$$

$$b_0(x, y) = \frac{b(x, y)}{\max(b(x, y))}$$

$L'(x, y)$ is converted to the HSV color space, and the same exposure-compensation level as step 1 is used along with the same exposure compensation function to compensate for the $V$ component in the HSV color space to get the new $L''(x, y)$. Next, $L''(x, y)$ is converted to RGB color to get the enhanced illumination component $L_{enh}(x, y)$. Finally, the compensated image is obtained by the following:

$$I_{enh}(x, y) = L_{enh}(x, y) \times R(x, y)$$

To obtain the optimal image quality, the compensation on both the YCrCb and HSV color spaces are combined based on the content analysis result. A weighted sum of the YCrCb and HSV color spaces is performed. Figures 3 and 4 show results on portrait and scenic images using the content-aware backlight compensation technique, respectively.

**Denoising**

Denoising is important for improving the quality of photos and videos captured by mobile phones. Compared with digital single-lens reflex (DSLR) and point-and-shoot cameras, mobile phone cameras usually have smaller lens and sensors, resulting in less light captured
by the sensor, and therefore a smaller signal-to-noise ratio and more noise.

One significant denoising issue on mobile phones is the low computation and memory capacity of mobile devices compared with PCs or laptops. On the other hand, today’s smartphone cameras typically output HD videos and photos with more than 5 megapixel (MP) resolution. Although many high-performance denoising algorithms already exist, only a limited number of denoising algorithms can be feasibly implemented on smartphone platforms for real applications. For example, the bilateral denoising algorithm with a 2D implementation, which is a well-known denoising algorithm, is too slow for smartphone camera applications. For fast denoising on mobile platforms, we developed accelerated denoising algorithms for both photo and video processing.

For video denoising, previous algorithms often use a motion-compensated spatio-temporal algorithm for denoising. However, motion estimation itself is a computationally expensive process and is difficult to implement in real time for HD videos. Because videos are usually compressed and saved after capture, we developed an encoder-integrated spatio-temporal denoising algorithm, which is based on H.264 compression. The algorithm can be easily adapted to other block-based compression schemes, such as MPEG.

The two-tap recursive temporal filter (implemented in the “filtering + residual comp” block in Figure 5) is equivalent to a nonrecursive temporal filter with a long filter length. Therefore, the fast recursive design ensures effective denoising performance.

In our experiments, the algorithm is implemented in the x264 open source encoder. Our proposed algorithm significantly improved the peak signal-to-noise ratio (PSNR) of the processed video compared with the prior encoder-integrated denoising approach with an average PSNR gain of 1.85 dB and little computation penalty (approximately 1 fps slower). It also yielded better visual quality, as Figure 6 shows.

For photo denoising, it is even more difficult to realize real-time or close to real-time performance because photo resolution usually is much larger than HD. For common smartphones or tablets, we found that even the accelerated bilateral filtering algorithm (for example, using separable approximation) would take 4 to 10 seconds to denoise one photo. Therefore, we developed a much faster algorithm based on the idea similar to our previously developed recursive temporal filter.

The new denoising algorithm uses a spatial recursive bilateral filter. However, the recursive filter is a causal filter, so it is not symmetric in the spatial domain. To realize symmetric filtering, we designed a two-pass algorithm, where the forward pass scans the image from the left to right, and the backward pass scans from the opposite. The result is the average of the output of the two passes (see Figure 7).

This recursive spatial bilateral filter is separable and has only two taps, so it is much faster than the traditional bilateral filters. It can be implemented with an in-place algorithm, which requires no intermediate memory buffer for the filtering process. Overall, denoising of an 8 MP photo can be completed in 1 second even for the low-end smartphones or tablets (such as a Nexus 10), which is 4 to 10 times faster than the conventional bilateral filters with separable implementations, and the denoising performance is even better than that obtained by separable implementations.

Deblurring

The blur process caused by camera shake during capture can be modeled as a latent image \( x \) convolved with a point spread function
(PSF) $k$. With an unknown $k$, recovering $x$ from $y$ (aka, blind deconvolution) is challenging. One popular approach is to assume sparsity in $x$. For example, earlier works used a generalized $L_0$ sparse expression for sparsity-pursuit regularization or directional filters to reconstruct $k$. To better estimate $k$, Neel Joshi and his colleagues used additional hardware assistance, where the full 6D camera motion is physically measured by a hybrid system consisting of a DSLR camera and several inertial sensors. However, existing methods generally do not perform well on real mobile photos for several reasons:

- Requiring additional hardware support other than the smartphone itself is impractical.
- Consumer photos are usually large, greater than 8 MP. Previous methods mostly work on less than 1 MP images, which already take minutes or hours on PCs.
- Real mobile photos have PSFs with simple, yet dense shapes because of the fast shutter speed (in milliseconds), which are difficult to be estimated accurately.
- Normal users do not have an interest in parameter tuning, so it should be avoided.
- Users have a low tolerance for artifacts, especially over human faces. A practical system should gracefully fail without generating excessive artifacts.

We propose an efficient sensor-assisted image deblurring framework on mobile devices. Modern smartphones are usually equipped with many sensors. Information about the camera motion obtained from such sensors can be used to compute the PSF. Figure 8 shows the overall framework of our deblurring system. We fuse information from gyroscopes, accelerometers, and magnetometers at high speed, which alleviates the gyro drift. The PSF is then estimated based on the 3D camera orientation computed from the fused sensor data. To accommodate the inconsistent time delay between the sensor output and the captured photo, we assume constant camera motion during photo capture and model the effect caused by such a time delay by an additional 2D Gaussian term. This is reasonable because of the fast shutter speed. We only consider 3D camera orientation because the inertial-sensor-based camera translation estimation is prone to error. For calibration, the camera parameters can be computed using only a set of consecutively captured photos by the mobile device. Addressing users’ sensitivity in artifacts over human faces, we further propose face-adaptive deblurring. We compute several features to measure the image quality over automatically detected face regions, based on which the appropriate deblurring algorithm is selected for each face via support vector machine (SVM) classifiers.

We tested our approach on 450 photos from seven users using an Android tablet and smartphone. We defined five blur categories: very
small ($k \leq 11$ pixels), small ($11 < k \leq 21$ pixels), medium ($21 < k \leq 41$ pixels), large ($41 < k \leq 61$ pixels), and very large ($k > 61$ pixels). We then compared our approach with two recent state-of-the-art algorithms: L0Reg and DirFil (directional filters). Figure 9 gives an example of the results.

In general, it is hard for the L0Reg and DirFil algorithms to recover good PSFs. They either prefer a nonblurred solution or generate severe artifacts. As for speed, L0Reg and DirFil took 10 minutes and 3 hours, respectively, to process an 8 MP image, whereas our method took only 35 sec.

Our results were presented side by side with the original photos to three users for subjective evaluation. Figure 10 shows the statistics of users’ evaluation. Based on the results in Figure 10, with a very small or small-sized blur, we do not degenerate the image quality at all. With a medium-sized blur, we fail only for 1 percent of cases. With a relatively large-sized blur, we can still slightly improve the overall image quality in most photos.
Video Stabilization

A video captured by a mobile device (such as a smartphone or tablet) often exhibits certain degree of jitter due to hand shakiness. Video stabilization aims to reduce or remove the undesirable artifact to improve video quality.

There are different approaches to stabilizing a shaky mobile video, but here we will focus on digital video stabilization. Digital video stabilization relies on CPU power and video processing algorithms to realign video frames so that they appear to be visually stable. Compared with the optical stabilization often found on high-end mobile devices, this is a low cost and flexible digital solution. However, it is still a challenge for a mobile digital stabilizer to produce stable videos comparable to the ones produced by an optical mechanism, especially on the fly. Video stabilization has been an active research topic for more than a decade. Despite its remarkable advances in performance,9,10 the majority of the existing algorithms are designed for offline editing, which means that they often require scanning a video multiple times and are computationally demanding.

In this section, we describe a digital video stabilizer that is suitable for real-time mobile video applications, such as video conferencing and instant video messaging. It features a one-pass processing structure as well as low-complexity algorithm components. On the top level, the video stabilizer consists of three major modules: global motion estimation, motion filtering, and view synthesis. Additionally, an internal sliding observation window helps process the information extracted from local neighboring frames to provide online processing capability.

Global Motion Estimation. The GME module takes in the original shaky video frames and estimates the camera motion (or global motion) by examining the relative motion between two consecutive frames. To maintain low complexity, diamond-search (DS) based motion estimation (ME) is applied. DS produces a motion field between each pair of consecutive frames, based on which a motion model is adopted to describe the motion, which can stabilize common camera jitters resulting from zooming, rotation, and translations.

Motion Filtering. A common strategy for real-time video stabilization is to treat intentional camera motion and unwanted camera shakiness as the low- and high-frequency components in a camera trajectory, respectively, and apply low-pass filtering to obtain a smoothed trajectory. However, it is difficult to set an appropriate cut-off frequency and have a perfect low-pass filter, so residual jitters often remain in a stabilized video.

We tackle the stabilization problem from a different angle. Denote $M_i$ as the model that describes the relative global motion between video frames $f_{i-1}$ and $f_i$, and $\tilde{M}_j$ as the accumulated motion between $f_i$ and $f_{j}(i < j)$ defined as

\[ \tilde{M}_j = M_i \cdot M_{i+1} \cdots M_{j} = \prod_{k=1}^{j} M_k. \]

In the ideal case, if we apply $\tilde{M}_j$ to warp $f_i$—that is, $\tilde{f}_i = \tilde{M}_j(f_i)$—then the warped frame $\tilde{f}_i$ will be temporally aligned with $f_i$. However, in reality, any intentional camera motion is also accumulated in $\tilde{M}_j$, and it may cause the warped frame to drift relative to a normal video frame (see Figure 11).

To counter the drifting effect, we need a mechanism that detects and compensates for intentional camera motion. To that end, we created an observation window to inspect camera motion statistics. A set of geometric means of the camera motions over that period is calculated, including the average camera zooming factor, rotation angle, and horizontal/vertical
shifts. As the current frame proceeds to the next, the window moves accordingly in a sliding window fashion with a new set of calculated geometric means. These values are continuously fed into a Kalman filter to estimate the associated intentional camera motion that occurs when a frame is captured. Compared with short-term, random vibrations, intentional camera motions are more consistent and last longer and thus should have a more substantial motion accumulation over the observation period. Therefore, we compare the Kalman filter estimate against a set of preset thresholds and declare an intentional camera motion when any of its motion parameters is significant enough. A detected intentional motion is then compensated by “subtracting” its effect from the corresponding motion model. Finally, the modified frame motion model is accumulated into $\hat{M}_i$, as shown in Equation 1.

The motion filtering module continuously accepts the frame motion models from GME, detects and compensates any underlying long-term intentional camera motion, and outputs accumulated motion models that track random camera vibrations over time.

View Synthesis. In the last step, the main task of the view synthesis module is to take the accumulated motion models output from the motion filtering module and apply them to warp the corresponding frames.

By operating within a sliding observation window (usually 8 frames long), the proposed algorithm realizes one-pass video stabilization. Together with the light-weight algorithm components such as DS-ME, a simple geometric motion model with the simplified calculation, it proves to be a good candidate for real-time online video stabilization for mobile platforms.

Our experimental results show that our algorithm produces more stable video than digital-stabilizer-based smartphones such as the Samsung Galaxy S4 and Apple iPhone 5. It also has more dampened vibration than the optical stabilizer based Nokia Lumia 1020 with occasionally less continuous motion in case of excessive shakiness.

New Paradigms
In the future, new applications will continue to emerge for mobile devices along with new mobile devices being created. As mobile devices continue to make hardware speed and quality improvements and new capabilities materialize, mobile media processing will also need to continue to evolve in order to effectively utilize different sources of information and improve the overall user experience.

References


John Wus is a senior researcher in the Media Lab at Futurewei Technologies. He has a PhD in electrical engineering from Lehigh University and is a senior member of IEEE. Contact him at john.wus@huawei.com.

Jun Tian is a senior researcher in the Media Lab at Futurewei Technologies. He has PhD in mathematics from Rice University and is a member of IEEE. Contact him at juntian@huawei.com.

Hongbing Li is a senior researcher in the Media Lab at Futurewei Technologies. He has a PhD in computer science from Nanjing University of China. Contact him at lihongbing@huawei.com.

Dong-Qing Zhang is a senior researcher in the Media Lab at Futurewei Technologies. He received his PhD in electrical engineering from Columbia University. Contact him at Dongqing.Zhang@huawei.com.

Wei Jiang is a senior researcher in the Media Lab at Futurewei Technologies. She has a PhD in electrical engineering from Columbia University and is a member of IEEE. Contact her at weijiang@huawei.com.

Zhenyu Wu is a senior researcher in the Media Lab at Futurewei Technologies. He has a PhD in electrical engineering from the University of Arizona. Contact him at zhenyu.wu@huawei.com.

Heather Yu is director of the Media Lab New Jersey office at Futurewei Technologies. She received her PhD in electrical engineering from Princeton University. Contact her at heatheryu@huawei.com.