Journal of Signal Processing Systems (2019) 91:9–20
https://doi.org/10.1007/s11265-018-1410-7

Image Restoration in Portable Devices: Algorithms and Optimization

Jan Kamenický[1] · Filip Šroubek[1] · Barbara Zitová[1] · Jari Hannuksela[2] · Markus Turtinen[2]

Received: 27 September 2017 / Revised: 15 March 2018 / Accepted: 25 September 2018 / Published online: 30 October 2018

Abstract
Image and video data acquired by portable devices such as mobile phones are degraded by noise and blur due to the small size of optical sensors in these devices. A wide range of image restoration methods exists, yet feasibility of these methods in portable platforms is not guaranteed due to limited hardware resources on such platforms. The paper addresses this problem by focusing on denoising algorithms. We have chosen two representatives of denoising methods with state-of-the-art performance, and propose different parallel implementations and algorithmic simplifications suitable for mobile phones. In addition, an extension to resolution enhancement is presented including both visual and quantitative comparisons. Analysis of the algorithms is carried out with respect to the computation time, power consumption and output image quality.

Keywords Image restoration · Denoising · Super-resolution · Numerical optimization · Portable devices

1 Introduction

Enhancement of image and video quality by restoration methods is one of the most important challenges of mobile imaging since noise and blur are always present in real world images. Appropriate algorithms able to diminish noise and sharpen images significantly increase resulting image quality and make the portable device camera even more attractive for customers. However, these algorithms can be computationally expensive and often the most power consuming parts in applications.

To meet this challenge, it is nowadays common to integrate several different computing devices in a single chip each accelerating specific algorithms. Specialized image signal processors (ISPs) have been shown to bring forth performance and power efficiency gains. Typically, industrial ISPs power consumption range from 150 to 250 mW depending on the resolution and frame rate [3]. This sets the commercial target for the power consumption level of algorithms running in the camera preview mode.

✉ Jan Kamenický
kamenik@utia.cas.cz

[1] Czech Academy of Sciences, Institute of Information Theory and Automation, Pod Vodárenskou věží 4, CZ-182 08, Prague 8, Czech Republic

[2] Visidon Ltd, Teknologiantie 2, 90590, Oulu, Finland

Designing, implementing, and tuning new algorithms for ISPs has remained a high-cost exercise. Therefore, a more general purpose solution such as mobile graphics processing units (GPUs) that handle image processing tasks is desired. Here, scalability of the solution using mobile GPUs helps to achieve improved energy efficiency and low power dissipation which is needed when dealing with battery powered devices. Therefore, parallel implementation using for example the OpenCL framework allows to cut down time-to-market and related costs.

In this context, analysis of the algorithm efficiency with respect to the achievable enhancement quality and usage of different mobile SoC (system on chip) computing units was realized on the multi-core ARM CPU, ARM NEON, and GPU platforms. This includes finding the optimal algoritmization, design patterns considering a good trade-off between the performance/throughput, energy efficiency and reuse via programmability. From the implementation point of view, all targets should be supported from the same C/C++/OpenCL application source code, and the processing time and the power consumption must be predictable, when the algorithm is integrated to end-users applications.

In this paper image restoration is the subject of such algorithm analysis. Image denoising is one of the fundamental digital image processing challenges, it has been studied for decades, yet it is still a valid research topic as the theoretical limits of this type of restoration are not well understood [5, 14]. The key idea of denoising is to average pixels with

similar intensities that ideally differ only by noise. By averaging similar pixels the noise variance decreases without further blurring the image. The key problem of denoising is to determine which pixels to average. There are generally two ways to address this problem: spatial averaging and temporal averaging. The first exploits self-similarity in images and searches for similar patches within a single image. The latter takes multiple images of the same scene and searches for similar pixels by spatially aligning the images.

Spatial averaging methods range from simple and fast algorithms, such as basic filtering using Gaussian mask or wavelet thresholding, to state-of-the-art and complex algorithms based on *non-local means* principles (NLMS) [2]. The idea of current methods is to cluster similar patches within a neighbourhood window and denoise them simultaneously. The patches to be denoised are accumulated in a buffer and pixel-wise normalized with respect to the number of overlapping patches. The baseline for high-quality image denoising using patches remains block matching with 3D filtering (BM3D) [6]. Later, non-local Bayes approach was proposed [13], which directly solves for the most likely patches by matrix inversion, which outperforms BM3D for vector images. A wide class of patch-based methods makes use of dictionaries for sparse representation of the patches [7]. Recently, many methods using neural networks appeared [4].

The second category - temporal averaging - has an advantage over spatial averaging. It combines a set of images and therefore the latent image is recovered with less blurring compare to spatial averaging. Averaging multiple images however requires accurate spatial alignement (image registration), otherwise blurring due to misregistration tends to appear. To achieve accurate registration, a four-step procedure [20] is typically applied: feature detection, feature matching, transform model estimation, and final image resampling and transformation. If the image viewpoints are close to each other and geometric transformations are subtle then it is more convenient to register via *optical flow* [9]. In addition, temporal averaging is naturally extendable to resolution enhancement, i.e. *super-resolution* [18]. Images registered with sub-pixel accuracy are warped on a reference high-resolution grid, averaged, and the final high-resolution image is deconvolved with a sensor blur estimate [8].

From each category one denoising method was chosen. These methods were not previously implemented on portable platforms due to their complexity. To tackle the problem of complexity, we propose appropriate algorithmic simplifications and different parallel implementations.

The paper is organized as follows. Section 2 provides the mathematical formulation of denoising and discusses in detail the selected spatial and temporal methods. Section 3 shows different implementations viable on mobile platforms. Section 4 covers experiments comparing the quality of denoised images, computational time and power consumption. The final Section 5 concludes the paper.

2 Image Denoising

The objective of image denoising algorithms is to reduce noise while preserving details in the image. The general model for an image degradation caused by additive noise and blurring is

$$g(x, y) = (h * f)(x, y) + n(x, y), \tag{1}$$

where $g(x, y)$ is an observed noisy image, $f(x, y)$ is an original image, $h(x, y)$ is the sensor PSF (Point Spread Function), and $n(x, y)$ is additive noise. As mentioned earlier there are two general categories of denoising methods: spatial averaging and temporal averaging. From each category we have selected one representative, which we analyze from the algorithmic and implementation perspective.

In the category of spatial averaging, the non-local means algorithm (NLMS) [2] is one of the most well-known methods for image denoising. The implementation of the patch based version (PNLMS) of this algorithm for portable devices is a computationally cheaper alternative than, e.g., the BM3D method [6]. Therefore, PNLMS has been considered for portable device implementation here.

In the case of temporal averaging, we extend the single-image model (1) to multiple acquisitions of K images g_k, $k = 1, \ldots, K$, as

$$g_k(x, y) = D(h * W_k f)(x, y) + n_k(x, y)\,, \tag{2}$$

where W_k denotes a geometric transformation (warping) of the k-th image to the reference grid and D is a sampling operator that models ideal sampling of the camera sensor. The reference grid is typically aligned with one image in the set, let us denote it g_r, and then W_r is the identity operator. The sampling operator is defined as multiplication by a sum of delta functions placed on a grid. Denoising by temporal averaging requires accurate estimation of W_k's. We assume images captured in the camera burst mode which results in small misalignment among images and we propose registration via patch-wise rigid optical flow (PROF). To further improve the image quality, we apply super-resolution (SR). In the model (2) this can be seen as inverting the effect of sampling operator D and sensor blur h.

2.1 Spatial Averaging – PNLMS Algorithm

In contrast to filters considering only neighboring pixels of the reference pixel, the NLMS algorithm [2] compares non-local pixel patches to each other. Typically fixed size patches such as 7×7 pixels are used. Non-locality of

the algorithm comes from the fact that the patches can in theory locate anywhere in the image. In practice, the patch locations are limited inside a local search window such as 21×21 pixels in order to reduce computations.

The patch wise algorithm weights pixel values for each square patch B_k centered at the location k in the image I as follows

$$\hat{B}_k = \sum_{l \in I} g(B_l) w(B_k, B_l) \tag{3}$$

where $g(B_l)$ is the unfiltered image patch centered at the location l. The weighting factor W_k for the patch k is

$$W_k = \sum_{l \in I} w(B_k, B_l) \tag{4}$$

and $w(B_k, B_l)$ is the weighting function which is a Gaussian

$$w(B_k, B_l) = e^{-\frac{\|g(B_k) - g(B_l)\|_2^2}{\sigma^2}}. \tag{5}$$

This function computes the sum of squared differences between neighboring patches and the reference patch. Therefore, each denoised output patch is obtained by weighted averaging of neighboring patches. In general, larger weights are given to patches that are similar to the reference patch. Parameter σ defines the strength for filtering.

After applying this procedure for all patches in the image, we average all the estimates to construct the final denoised image. For all pixels i in the image, we aggregate all overlapping patches as follows

$$\hat{f}(i) = \frac{\sum \hat{B}_k}{\sum W_k} \tag{6}$$

where $\sum \hat{B}_k$ is the sum of weighted patches contributing to pixel i, $\sum W_k$ is the sum of patch weights contributing to pixel i and $\hat{f}(i)$ is the denoised pixel value at the location i in the image. Using overlapping image patches, we can avoid block effects in the patch boundaries. However, it is also possible to use non-overlapping patches when targeting computationally fast implementations. In this case, we also avoid synchronizing memory writes to the final output image if multi-threading is utilized.

2.2 Temporal Averaging – PROF-SR Algorithm

The proposed temporal averaging method consists of two main parts; in the first step the images are geometrically registered by means of the patch-wise rigid optical flow (PROF) and in the second step the enhancement itself is realized using the super-resolution (SR) approach.

2.2.1 PROF Registration

Computing dense optical flow is time consuming on mobile devices. Therefore, we propose to restrict ourselves to rigid transformations and perform calculations patch-wise. The described method is inspired by [15] with several simplifications and modifications. It is possible to use homography (projective transformation) with optical flow leading to linear equations when restricting to small homography deformations only. Let the reference frame be the r-th image g_r from the set, and we register the other frames g_k's to the reference one. The optical flow constraint at the location i is

$$g^x(i)\Delta x(i) + g^y(i)\Delta y(i) + g^t(i) = 0\,. \tag{7}$$

where g^x and g^y denote derivatives of the reference image g_r with respect to spatial coordinates, and $g^t = g_k - g_r$. To simplify the notation, we will further omit the pixel location index i. Δx and Δy are the shifts of corresponding pixels between the two images. We assume that in every image patch the shifts are parametrized by a homography P and this leads to

$$[g^x, g^y, g^t - xg^x - yg^y]\mathbf{P}[x, y, 1]^T = 0\,, \tag{8}$$

where the column vector $[x, y, 1]^T$ denotes homogeneous coordinates of the location i and $\mathbf{P}$ is the 3×3 homography transformation matrix. The same equation exists for every pixel in the image patch and the only unknowns are the nine homography parameters of $\mathbf{P}$. Let $\mathbf{p}$ denote these unknown parameters as a column vector. Rewriting the equation to extract the unknowns, we obtain

$$[xg^x, yg^x, g^x, xg^y, yg^y, g^y, xg', yg', g']\mathbf{p} = 0\,, \tag{9}$$

where $g' = g^t - xg^x - yg^y$. Stacking the equations for every pixel, we construct a "tall" matrix with 9 columns for homography parameters and the number of rows equal to the number of pixels in the patch. We apply singular value decomposition and the right singular vector corresponding to the smallest singular value is the solution.

Patch-wise estimated homographies are then used to construct warping operators W_k in Eq. 2. Note that here we assume that the order of convolution and warping is interchangeable, which is not precisely true. For linear transformations the assumption is correct. For non-linear transformations, which includes homography, interchanging operators transfers the original convolution to space-variant convolution, however the PSF variations are subtle and can be ignored in practice. The denoised image $\hat{f}$ is estimated as the mean of registered images, i.e.,

$$\hat{f}(i) = \frac{1}{K} \sum_{k=1}^{K} W_k^{-1} g_k(i)\,. \tag{10}$$

Instead of the mean one can calculate the median to have a more robust estimator yet at a cost of higher memory consumption (we need all warped images to calculate the median).

2.2.2 Non-iterative SR Algorithm

Super-resolution is an inverse problem for the model (2). Sampling operator D is ill-conditioned, and for a single image ($K = 1$), the inverse problem is ill posed. If the number of images is large enough so that the number of unknowns (size of f) is smaller than the number of equations in Eq. 2 then the inverse problem is overcomplete and thus better posed. Full SR methods try to solve such inverse problems. For mobile platforms this is however not feasible, since the inverse problems are complex and require iterative optimization methods that are computationally demanding. Instead, we apply an approximative method which is simple and not iterative with the quality performance similar to full SR methods.

When W_k's are estimated, the model (2) is approximately given by

$$\frac{1}{K}\sum_{k=1}^{K} W_k^{-1} D^T g_k = h * f\,, \tag{11}$$

where D^T is an upsampling operator and $W_k^{-1} D^T$ performs warping of g_k on a high-resolution grid. Again, the mean can be replaced by the median to increase robustness to outliers which are produced, e.g., by incorrect registration or saturated pixels.

Finally to estimate the original high-resolution image f, a deconvolution step, which in our case is the Wiener filter, is applied to remove the effect of the sensor PSF h.

3 Implementation

In this section we discuss implementation details of the PNLMS and PROF-SR algorithms.

3.1 PNLMS Implementation

Three implementations of the PNLMS algorithm were made for different targets: multi-threaded C/C++ for CPU, parallelized ASM with NEON extensions, and OpenCL for GPU. Both C/C++ and ASM implementations are able to run on ARM CPU cores. The GPU version was implemented using OpenCL API.

3.1.1 Mobile Multi-core CPU

Mobile processors are designed to consume less power and dissipate less heat than desktop processors, while using a smaller silicon size. To preserve battery life, mobile processors can work with different power levels and clock frequencies. The operating system can set cores on-line or off-line depending on computational load and thermal status. For example, if the device gets heated for a long time, CPU/GPU cores are set to a lower frequency by the system.

Processors used in mobile devices are mostly based on the ARM architecture which describes a family of processors designed in accordance with RISC principles. A VFP (Vector Floating Point) extension is included to provide for low-cost floating point computations although later versions of the architecture have abandoned it in favor of a more complete NEON SIMD extension.

The particularities of ARM processors enable C code optimizations to achieve higher performance. We have implemented an ARM optimized version of the PNLMS algorithm that avoids conditional branching and utilizes the built-in ARM registers to reduce the number of memory accesses.

Most of the newest devices include processors with several cores. All the cores usually have the NEON extension access. In Android devices, different tasks can be assigned to the cores by using several APIs. The processor cores can share data with different techniques such as shared caches.

3.1.2 Mobile Multi-core CPU with NEON Extension

Many computationally intensive algorithms requiring high performance cannot be carried out on the mobile application processor alone in real-time. For this purpose, a wide range of accelerators have been included as specific arithmetic units or co-processors. Many ARM based mobile processors provide signal processing acceleration by including a SIMD instruction set known as NEON, which shares floating point registers with the VFP. NEON supports 8-, 16-, 32- and 64-bit integer and single-precision (32-bit) floating point data and SIMD operations.

Bordallo et al. [1] reported that the use of NEON increases the power consumption. However, a better performance over power can be achieved, with about a 40% gain in performance for only a 20% increase in the power consumption. Considering energy consumption, the highest contributing factor is usually the execution time for similar CPU utilization.

NEON can be accessed by using, for example, inline assembly instructions or NEON intrinsic. Grasso et al. [11] claim that the high performance is achieved only through manual code optimization and tuning. Our experiments support this claim as it was impossible for us to get optimized solution using intrinsic or auto vectorization functionality. In our design, we try to use NEON registers as efficiently as possible. The shortcoming is that the code is

not easily portable and the designer should be careful when integrating the inline assembler code to new projects.

3.1.3 Mobile GPU

A mobile GPU is especially useful for executing tasks that can be parallelized. Its resources are most conveniently and portably utilized with a standard API. A number of projects have targeted real-time implementations for the desktop GPU acceleration using CUDA [10, 16, 17]. On a mobile device platform the choice was earlier limited to OpenGL ES but now the OpenCL framework offers flexibility similar to vendor specific solutions designed for desktop computers. All major mobile GPU vendors such as ARM's Mali, Qualcomm's Adreno, Imaginations PowerVR and Vivante, support OpenCL in one way or another.

Our OpenCL kernel, which computes the PNLMS algorithm performs image denoising for non-overlapping 8×8 image blocks. Then, the memory synchronization is not needed. If SIMD instructions are supported on the GPU, the implementation can benefit from hand-tuned vectorization of the code. In the sum of squared difference (SSD) computing, we adopt the vector data type of *short8* to load one row from the 8×8 pixel block simultaneously. With some mobile GPUs this leads to considerably faster code. Also, the designer should favor built-in fast integer math operations, which can improve the performance significantly.

Considering the use of local memory, the proper workgroup size reduces the redundancy in global memory accesses. However, a larger workgroup size needs more local memory. Since the local memory can be limited to 8 kbytes in some Adreno devices, this practically limits the possible workgroup combination that we can handle with the current generation of hardware.

3.2 PROF-SR Implementation

For registration we use rigid homography optical flow described in Section 2.2.1. However, for the optical flow to work efficiently it needs to be performed on multiple scales. We begin by downsampling the images several times by a factor of 2 and then on each scale perform the registration, update the transformation matrix, go one scale up and repeat. This multiscale approach is necessary to be able to cover transformations where pixel positions differ more than 2-3 pixels.

It is often the case that images do not differ strictly by a homography, e.g. a rolling shutter effect introduces slight differences that smoothly increase throughout the image or the scene is not planar and the camera is not only rotating along its optical center. For this reason we implemented a patch-wise registration. The images are divided into several patches, e.g. an 8×8 grid covering the whole image area. Then registration is performed per patch leading to potentially slightly different transformation matrices. Final warping is computed by homography matrices for each pixel and the matrices are defined by bilinear interpolation of transformations in neighbouring patches. This leads to a smooth and "elastic"-like transformation which registers the images more precisely. Yet, care must be taken to have large enough patches to ensure correct registration, or detect failed registration and deal with it appropriately. The Wiener filter is computed in the Fourier domain. The FFT algorithm implemented in the FFTW library was used for converting images to the frequency domain. We use the FFTW port specifically tuned for multi-core ARM CPUs with SIMD instruction set (NEON). The final algorithm with PROF multi-scale registration and SR is summarized in Algorithm 1.

Input images are in YUV420 color space. That means we have a gray-scale image Y and two UV color components with $4\times$ smaller resolution. Most information is contained in the gray-scale part and therefore we use only Y in the registration step. Color information from the UV components is incorporated during the final warping using the same transformation matrices.

Algorithm 1 Mobile PROF-SR algorithm

```
for all input images except the reference one do
    divide image into patches
    for each patch do
        downsample maxLevel times with factor 2
        P ← identity transformation matrix
        for each pyramid level do
            upsample P, warp image according to P
            estimate P′ using OF registration described in Section 2.2.1
            P ← PP′
        end for
    end for
    interpolate image on a high-resolution grid using patch-wise estimated H
end for
combine all images by calculating the mean or median
Wiener filter
```

4 Experiments

The experimental section is divided into three parts. The first part provides visual assessment of implemented denoising algorithms on real data taken by a mobile phone. In the second part we compare computation time and power consumption of different implementations of the PNLMS

algorithm, which represents spatial-averaging denoising. The third part quantitatively evaluates resolution improvement of the PROF-SR algorithm, which belongs to the category of denoising methods based on temporal averaging.

4.1 Visual Assessment

We show results of denoising methods on real data captured by LG Nexus 5 mobile phone camera (8 MP, f/2.4, 30 mm, 1/3.2", 1.4 μm), see Fig. 1. The raw image as returned by the camera API is in (a). The default denoising method in the mobile phone which stores the image as JPEG is shown in (b). Notice that the level of noise in the JPEG image is much lower than in the original image, yet at a cost of severe loss of details. Results of the denoising methods PNLMS and PROF that we implemented in the phone are in (c) and (d), respectively. The last row compares results for the SR factor of 2 achieved by: (e) proposed non-iterative PROF-SR method and (f) full iterative SR method [19]. In the case of temporal averaging PROF and both SR methods, the results were computed from ten input images. It is therefore logical that they outperform the spatial-averaging PNLMS method (c) which works with only one image. Notice that the deconvolution step in SR methods further improves the final image, however the additional benefit of the complex SR method (f), which solves the inverse problem precisely, is negligible in this case.

4.2 Computational and Power Performance

This section reports and discusses the results obtained with different implementations on the Qualcomm's Snapdragon

(a) input raw image (b) default JPEG produced by the camera

(c) spatial averaging – PNLMS (d) temporal averaging – PROF

(e) proposed non-iterative PROF-SR (f) full iterative SR

Figure 1 Denoising of images captured by a mobile phone camera.

Table 1 The computation time results when running implementations of the noise reduction algorithm for different image sizes.

Implementation	2MP: 1920 × 1080	8MP: 3840 × 2160	16MP: 5312 × 2988
Plain C, CPU with 1 core	430 ms	1615 ms	4641 ms
Multi-threaded C, CPU with 4 cores	174 ms	690 ms	1953 ms
Mixed multi-threaded C/ASM, CPU with 1 core + NEON	238 ms	986 ms	2790 ms
Mixed multi-threaded C/ASM, CPU with 4 cores + NEON	99 ms	402 ms	1170 ms
OpenCL, GPU	22 ms	51 ms	94 ms

820 mobile platform. The CPU in this platform is a quad-core Kryo@2.2GHz and the GPU is an Adreno 530. The OS in the test device is Android 7.0 "Nougat". The Adreno 530 GPU has clock frequency choices 510, 624, and 650 MHz. In our tests, the maximum frequency of 650 MHz was used. The OpenCL API version is 2.0 Full profile.

4.2.1 Computation Time

The objective for this test is to evaluate different implementations of the PNLMS denoising algorithm on the target platform. Totally, we tested 6 implementations with three image resolutions 2MP, 8MP, and 16MP. The performance was measured on the Qualcomm Snapdragon development board including the MSM8996 chipset, which is a newer version of the MSM8974 chipset included in Nexus 5. Both of these chipsets belong to Snapdragon 800 processor series.

Algorithm parameters were set differently for the preview mode (real-time requirement) and still image capturing. In the preview mode, the patch size was 8×8 pixels, the search window size was 9×9 pixels with 3 pixels step in each direction (totally 9 neighbors considered), and the sliding window step was 8 pixels. In the still image capture (hiqh quality mode), the patch size was 8×8 pixels, the search window size was 11×11 pixels with 1 pixels step in each direction (totally 121 neighbors considered), and the sliding window step was 7 pixels. These parameters used in testing were chosen to guarantee a good enough quality of results and at the same time minimize computational cost.

The results are summarized in Table 1 and it can be seen that the OpenCL based GPU implementation achieves the best performance with all image resolutions. The FullHD (2MP) test case provides the lowest speedup (19X) compared to the baseline plain C implementation while the best speedup is achived with 16MP images. This is an expected result because the GPU can parallelize larger tasks better.

In our earlier work on this topic, transferring the image data to the GPU took 65 ms and writing the output image took 15 ms [12]. In the proposed implementation, this is not a problem anymore and delay of transferring the image data to the GPU was smaller (10 ms) due to improved GPU hardware and newer driver software.

These earlier tests also showed that the implementation for GPU (Adreno 330) did not give significant improvement for the processing time compared to the multithreaded NEON optimized version. However, the new results clearly show the advantage of using the GPU for image denoising. The most significant reason for improved results was the new GPU hardware (Adreno 530) with increased data transfer and processing capability.

4.2.2 Power Consumption

Power consumption was measured as the total system power on the Qualcomm Snapdragon 820 development board. We used the National Instruments NI 4065 measurement device for measuring the electric current. The measurement device was connected to between the battery connector and battery of the target device and then an averaged period of current measurements were captured.

First, the baseline system power without the algorithm running was measured in order to determine the actual power consumption of the algorithm. Table 2 summarizes the measured power consumption on the target platform using the GPU version of the algorithm.

Figure 2 shows current [mA] measurement for 30fps FullHD (1920 × 1080) stream, and for comparison Fig. 3 shows similar measurements for CPU+NEON version of the algorithm. It should be noted that only the GPU implementation can achieve the real-time frame rate (30fps). The fastest CPU implementation could only achieve 10fps.

The development board uses ~ 12 V power supply, and with easy POWER = VOLTAGE × CURRENT

Table 2 The results for the power consumption test for the GPU implementation of the algorithm.

Image resolution	Current [mA]	Power consumption [mW]
2MP	91	1092
8MP	140	1680

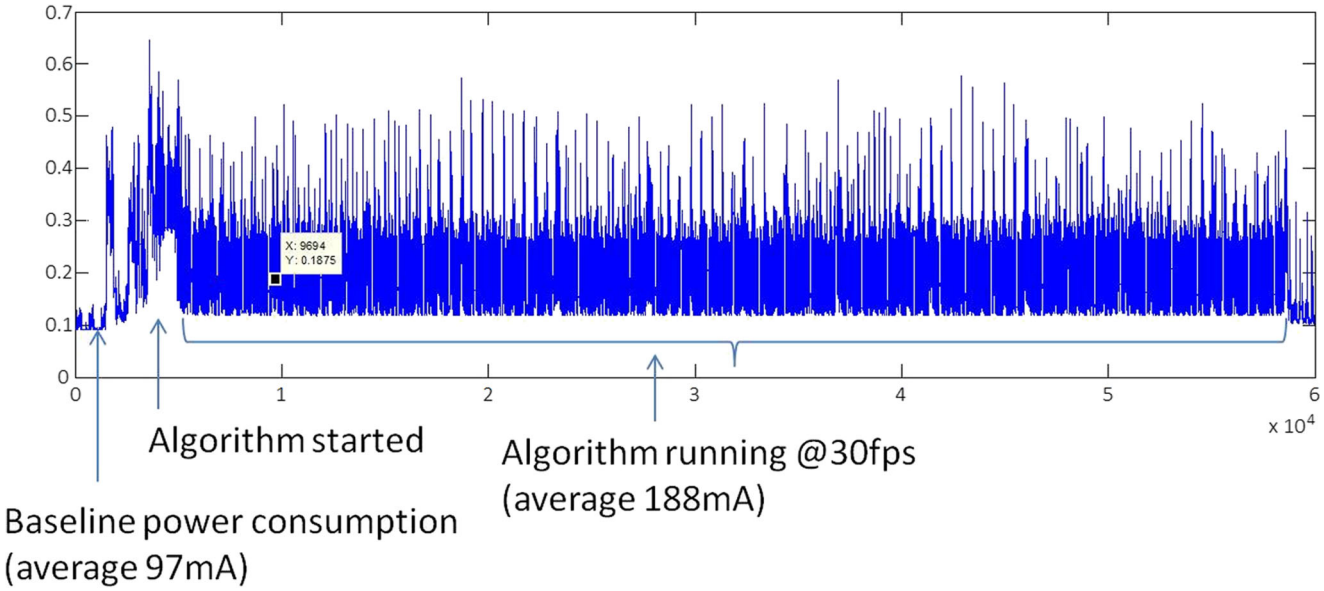

Figure 2 Power consumption measurement (mA) when running noise reduction for FullHD (1920 × 1080) size frames at 30 frames per second with GPU (OpenCL) implementation. X-axis shows time and y-axis shows current in the range between 0 and 700 mA.

computation it can be approximately calculated that the power consumption is 12 V × 0.091 A = 1.092 W = 1092 mW. This is clearly higher than expected. For comparison, the fastest CPU implementation using only 10fps consumes 12 V × 0.433 A = 5.196 W = 5196 mW.

Because the typical target is much less than the minimum achieved power consumption of 1092 mW, we wanted to find the minimum power consumption of target hardware when the GPU is activated with the OpenCL framework. For this experiment, we only load FullHD size images from

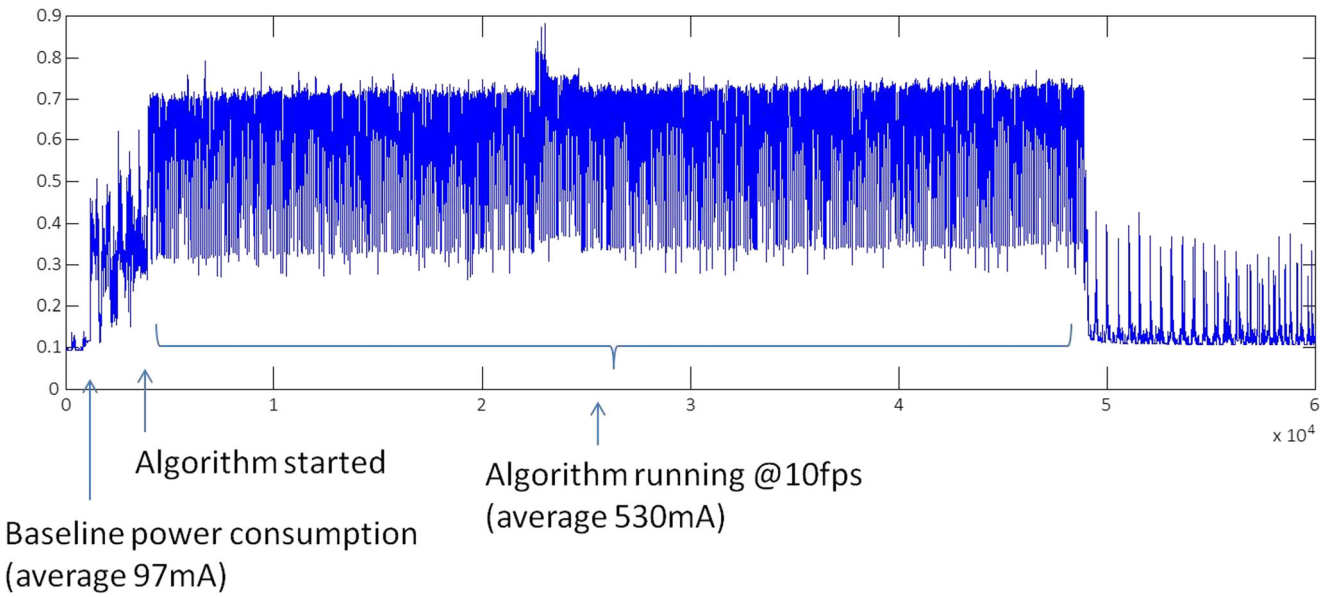

Figure 3 Power consumption measurement (mA) when running noise reduction for FullHD (1920 × 1080) size frames at 10 frames per second with CPU (4 thread + ARM NEON) implementation. X-axis shows time and y-axis shows current in the range between 0 and 900 mA.

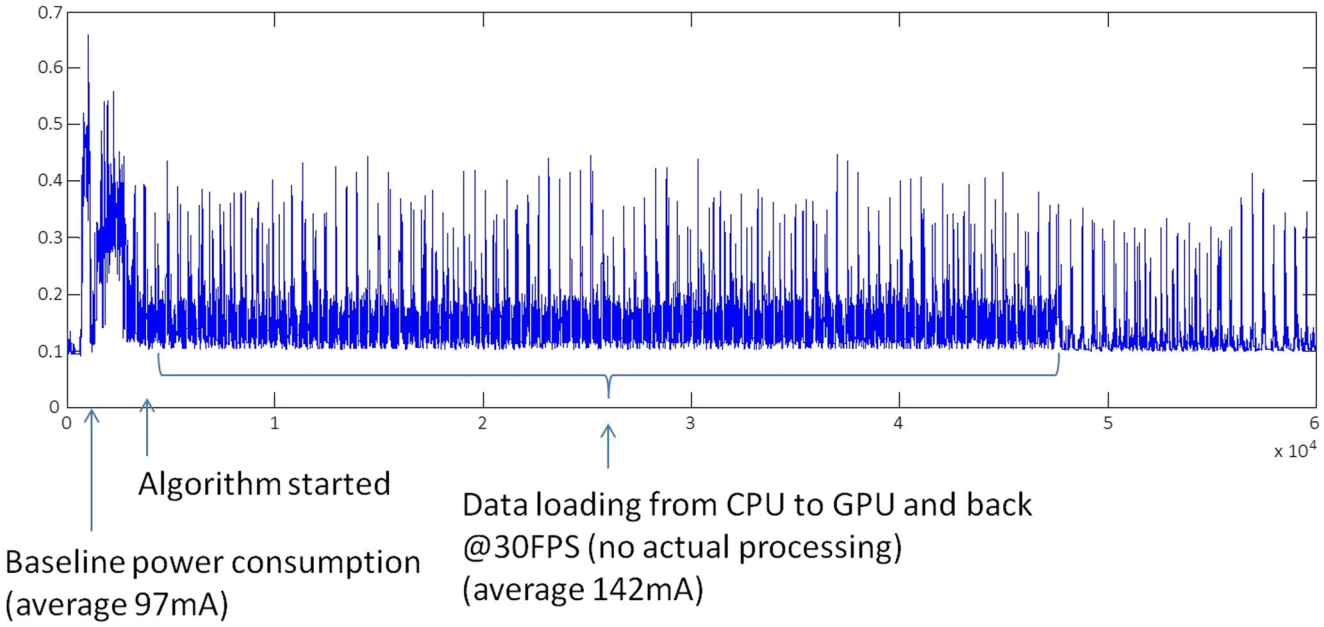

Figure 4 Power consumption measurement (mA) when loading FullHD (1920 × 1080) size frames at 30 frames per second from CPU to GPU and reading back. X-axis shows time and y-axis shows current in the range between 0 and 700 mA.

CPU to GPU and read it back from GPU to CPU at the rate of 30fps. Figure 4 shows the result of this experiment. It can be calculated that the power consumption of such process is 12 V × 0.045 A = 0.54 W = 540 mW. Based on this observation it is worthwhile to say that it is impossible to achieve power consumption level of ISPs with the Qualcomm Snapdragon 820 when the GPU is used to process FullHD size images at 30fps.

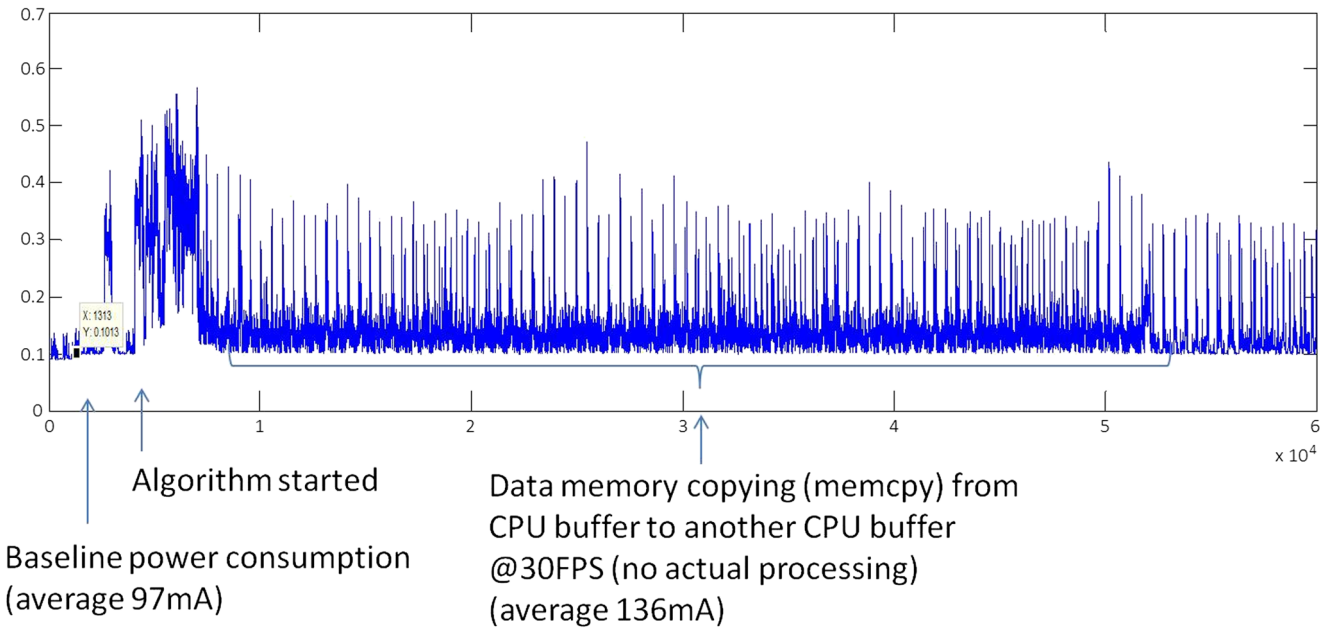

Figure 5 Power consumption measurement (mA) when copying FullHD (1920 × 1080) size frames at 30 frames per second from CPU buffer to another CPU buffer (memcpy). X-axis shows time and y-axis shows current in the range between 0 and 700 mA.

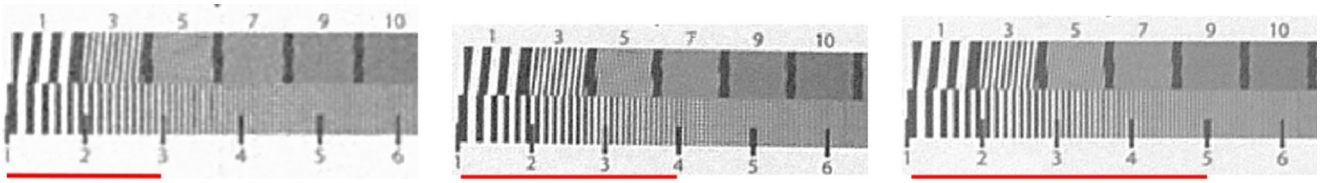

Figure 6 Performance of SR depends, to a certain extent, on the number of input images. Notice moiré patterns that appear in the area of higher line frequencies, which is an indication of image insufficient resolution. The SR reconstruction extends the spectral region (red line), in which details are correctly reproduced. The image quality stops improving beyond ten inputs.

In addition to GPU framework power consumption testing, we wanted to determine the power consumption when only CPU is used and very simple buffer copying (memcpy) is done at 30fps rate. Figure 5 shows the result of this experiment. It can be calculated that the power consumption of such process is 12 V $\times$ 0.039 A $=$ 0.468 W $=$ 468 mW. Therefore it can be assumed that all memory intensive computation, such as processing large images at video frame rate would consume more power than 500 mW, and the power consumption level of the ISP is not possible to achieve with the current hardware.

4.3 Resolution Enhancement Performance

In this section we measure image-quality performance of the proposed PROF-SR algorithm with respect to different criteria.

4.3.1 Number of Input Images

We took several images of the test chart ISO 12233 with the mobile camera and the SR image quality with respect to the number of images is compared in Fig. 6. The resolution quality is measured as the number of line widths (cycles) per picture height [lw/ph]. For this experiment setting, one step in the test chart is approximately 300 lw/ph. As the number of input images increases the high-frequency information is better restored. Note that the time complexity of the temporal averaging increases linearly with the increasing number of input images. With five images we can improve recognition from the original 800 lw/ph (left image) to 1100 lw/ph (middle image) for this particular mobile device (Nexus 5). With ten images we get to the resolving power of 1400 lw/ph (right image). Because of model inconsistencies we were not able to improve beyond this point and more input images only slow down the computation.

4.3.2 Comparison with the Full Iterative SR

The proposed PROF-SR algorithm is a simplified version of an iterative SR method [19] that solves the full inverse problem. This simplification allowed us to circumvent a time-complex iterative numerical solution that is difficult to implement on embedded devices such as smartphones. It is important to quantitatively evaluate to what degree the proposed simplified algorithm lacks behind the complex iterative one. We have performed the following experiment to provide this comparison. We took with the mobile camera ten images of the test chart "Siemens star" (see Fig. 7), in which the high-frequency content gradually increases towards the center. The moiré pattern appears close to the center of the star where the lines approach each other, which indicates the high-frequency information is not correctly represented in the input images.

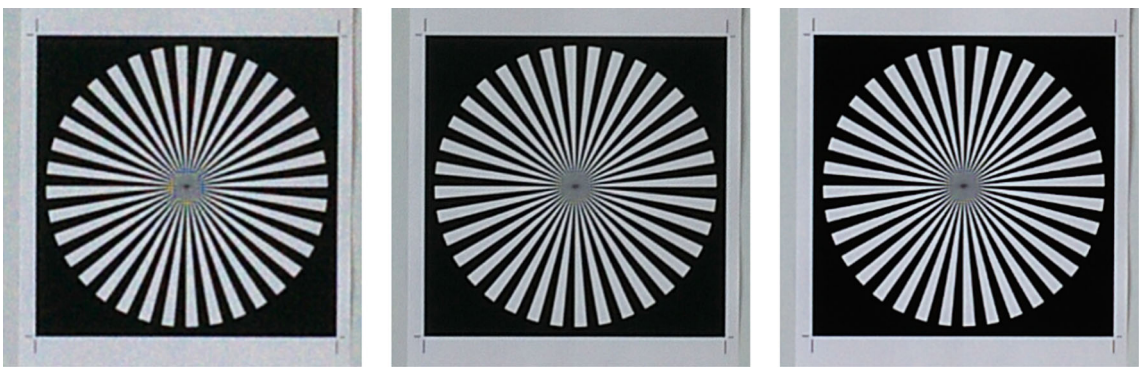

Figure 7 Comparing performance of the proposed PROF-SR algorithm with the full SR method.

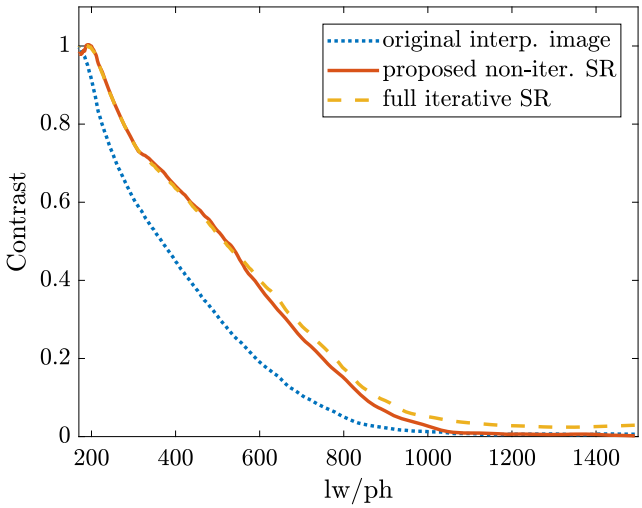

Figure 8 Contrast versus line widths per picture height (lw/ph) for three resolution enhancement methods: (blue dotted line) linear interpolation of a single image, (yellow dashed line) full iterative SR method, (solid red line) proposed non-iterative PROF-SR method. Both SR methods were applied to ten captured photos.

We can estimate how well different spatial frequencies are represented by calculating the contrast along circular profiles of the star. The diameter of circular profiles is inversely proportional to the spatial frequency, which we measure again as lw/ph. The graph in Fig. 8 plots the contrast versus lw/ph of the three images from Fig. 7. Both SR methods improve contrast over the linear interpolation significantly in the range between 400 and 1000 lw/ph. The most important conclusion is however that in this case the complex iterative SR method performs almost identically to the proposed approximative SR.

5 Conclusion

We have analyzed the feasibility of implementing two denoising methods based on spatial and temporal averaging on mobile platforms. Both methods provide images with less noise and superior image quality compared to the default denoising algorithm for JPEG photos implemented in mobile phones. Different implementations and algorithmic simplifications were considered. Algorithm analysis was carried out with respect to the computation time, power consumption and output image quality. In the case of temporal averaging, we also implemented and tested a non-iterative super-resolution algorithm optimized for mobile platforms. The resulting algorithm is fully functional with a negligible image-quality loss compare to the more complex version of iterative super-resolution. In the case of spatial averaging and mobile GPU implementation, we have achieved real-time performance on FullHD videos. In addition, GPU implementation with OpenCL offers improved flexibility and portability compared to optimized CPU specific implementations using, for example, the NEON intrinsics. Albeit the power consumption is much higher than the current ISPs, which renders the algorithm not fully suitable for being used in the camera preview mode by default, the on-demand use of the algorithm by end users is viable as it delivers a perceivable increase of image quality.

Acknowledgments This work was supported by ARTEMIS JU project 621439 (ALMARVI) and partially also by the Czech Science Foundation project GA18-05360S.

References

1. Bordallo López, M., Nieto, A., Boutellier, J., Hannuksela, J., Silvén, O. (2014). Evaluation of real-time lbp computing in multiple architectures. *Journal of Real-Time Image Processing*, 1–22.
2. Buades, A., & Coll, B. (2005). A non-local algorithm for image denoising. In *Computer vision and pattern recognition (CVPR)* (pp. 60–65).
3. Buckler, M., Jayasuriya, S., Sampson, A. (2017). Reconfiguring the imaging pipeline for computer vision. In *IEEE International conference on computer vision (ICCV)* (pp. 975–984).
4. Burger, H.C., Schuler, C.J., Harmeling, S. (2012). Image denoising: can plain neural networks compete with BM3D? In *Computer vision and pattern recognition (CVPR)* (pp. 2392–2399).
5. Chatterjee, P., & Milanfar, P. (2010). Is denoising dead? *IEEE Transactions on Image Processing*, *19*(4), 895–911.
6. Dabov, K., Foi, A., Katkovnik, V., Egiazarian, K. (2007). Image denoising by sparse 3-D transform-domain collaborative filtering. *IEEE Transactions on Image Processing*, *16*(8), 2080–2095.
7. Elad, M., & Aharon, M. (2006). Image denoising via sparse and redundant representations over learned dictionaries. *IEEE Transactions on Image Processing*, *15*(12), 3736–3745. https://doi.org/10.1109/TIP.2006.881969.
8. Farsiu, S., Robinson, M., Elad, M., Milanfar, P. (2004). Fast and robust multiframe super resolution. *IEEE Transactions on Image Processing*, *13*(10), 1327–1344.
9. Fortun, D., Bouthemy, P., Kervrann, C. (2015). Optical flow modeling and computation: a survey. *Computer Vision and Image Understanding*, *134*, 1–21.
10. Goossens, B., Luong, H., Aelterman, J., Pižurica, A., Philips, W. (2010). *A GPU-accelerated real-time NLMeans algorithm for denoising color video sequences*, (pp. 46–57). Berlin: Springer.
11. Grasso, I., Radojkovic, P., Rajovic, N., Gelado, I., Ramirez, A. (2014). Energy efficient hpc on embedded socs: optimization techniques for Mali gpu. In *2014 IEEE 28th International parallel and distributed processing symposium* (pp. 123–132).
12. Hannuksela, J., Niskanen, M., Turtinen, M. (2015). Performance evaluation of image noise reduction computing on a mobile platform. In *2015 International conference on embedded computer systems: architectures, modeling, and simulation (SAMOS)* (pp. 332–337).
13. Lebrun, M., Buades, A., Morel, J.M. (2013). A nonlocal Bayesian image denoising algorithm. *SIAM Journal on Imaging Sciences*, *6*(3), 1665–1688.

14. Levin, A., & Nadler, B. (2011). Natural image denoising: optimality and inherent bounds. In *Computer vision and pattern recognition (CVPR)* (pp. 2833–2840).
15. Liu, C. (2009). Beyond pixels: exploring new representations and applications for motion analysis. Ph.D. thesis, Massachusetts Institute of Technology.
16. Márques, A., & Pardo, A. (2013). *Implementation of non local means filter in GPUs*, (pp. 407–414). Berlin: Springer.
17. Palma, G., Comerci, M., Alfano, B., Cuomo, S., Michele, P.D., Piccialli, F., Borrelli, P. (2013). 3d non-local means denoising via multi-gpu. In *2013 Federated conference on computer science and information systems* (pp. 495–498).
18. Šroubek, F., Cristobal, G., Flusser, J. (2007). A unified approach to superresolution and multichannel blind deconvolution. *IEEE Transactions on Image Processing*, *16*(9), 2322–2332.
19. Šroubek, F., Flusser, J., Cristóbal, G. (2009). Super-resolution and blind deconvolution for rational factors with an application to color images. *The Computer Journal*, *52*, 142–152.
20. Zitová, B., & Flusser, J. (2003). Image registration methods: a survey. *Image and Vision Computing*, *21*, 977–1000.

Jan Kamenický recieved his Ph.D. degree in software engineering from the Czech Technical University in Prague, Czech Republic, in 2011. He is currently with the Department of Image Processing at the Institute of Information Theory and Automation, Czech Academy of Sciences, Prague. His research interests are in the areas of image deconvolution and super-resolution, image registration, dense optical flow, medical image processing, and image/video processing in security applications.

Filip Šroubek received the MS degree in computer science from the Czech Technical University, Prague, Czech Republic in 1998 and the PhD degree in computer science from Charles University, Prague, Czech Republic in 2003. From 2004 to 2006, he was on a postdoctoral position in the Instituto de Optica, CSIC, Madrid, Spain. In 2010 and 2011, he was the Fulbright Visiting Scholar at the University of California, Santa Cruz. He is currently with the Institute of Information Theory and Automation, the Czech Academy of Sciences, and teaches at Charles University. Filip Sroubek is an author of eight book chapters and over 80∼journal and conference papers on image fusion, blind deconvolution, super-resolution and related topics.

Barbara Zitová received her Ph.D degree in software systems from the Charles University, Prague, Czech Republic, in 2000. She is a head of Department of Image Processing at the Institute of Information Theory and Automation, Czech Academy of Sciences, Prague. She teaches courses on Digital Image Processing and Wavelets in Image Processing. Her research interests include geometric invariants, image enhancement, image registration, image fusion, medical image processing, and applications in cultural heritage. She has authored/coauthored more than 70 research publications in these areas, including the monographs Moments and Moment Invariants in Pattern Recognition (Wiley, 2009) and 2D and 3D Image Analysis by Moments (Wiley, 2016). In 2003 Barbara Zitová received the Josef Hlavka Student Prize, the Otto Wichterle Premium of the Czech Academy of Sciences for young scientists in 2006, and in 2010 she was awarded by the SCOPUS 1000 Award for receiving more than 1000 citations of a single paper.

Jari Hannuksela received his M.Sc. and Ph.D. (Tech.) degrees from the Department of Electrical and Information Engineering at the University of Oulu (Finland) in 2003 and 2009, respectively. After a brief post-doc in the Center for Machine Vision Research, he was an acting professor of signal processing systems in the Department of Computer Science and Engineering at University of Oulu from January 2010 to June 2015. He is currently a R&D director at Visidon, a company that he co-founded in 2006. He is also an adjunct professor of embedded computer vision at University of Oulu.

Markus Turtinen received his Ph.D degree in computer engineering from the Univerisy of Oulu in 2007. His research was related to machine vision and machine learning. In 2006 he co-founded Visidon Oy, a technology company specialized in mobile imaging solutions and since then he has been working as a CEO for this company and has successfully built company to become one of the leading vendors in global mobile imaging business. Until today, Visidon imaging products have been integrated over 1 billion devices around the world.