Evaluation of color error and noise on simulated images

Clémence Mornet, Jérôme Vaillant, Thomas Decroux, Didier Hérault, Isabelle Schanen

Evaluation of Color Error and Noise on Simulated Images

Clémence Mornet[a], Jérôme Vaillant[a], Thomas Decroux[a], Didier Herault[a] and Isabelle Schanen[b]

[a]STMicroelectronics, 850 rue Jean Monnet, 38926 Crolles Cedex, France;
[b]IMEP-LAHC, 3 parvis Louis Néel BP257, 38016 Grenoble Cedex 1, France

ABSTRACT

The evaluation of CMOS sensors performance in terms of color accuracy and noise is a big challenge for camera phone manufacturers. On this paper, we present a tool developed with Matlab at STMicroelectronics which allows quality parameters to be evaluated on simulated images. These images are computed based on measured or predicted Quantum Efficiency (QE) curves and noise model. By setting the parameters of integration time and illumination, the tool optimizes the color correction matrix (CCM) and calculates the color error, color saturation and signal-to-noise ratio (SNR). After this color correction optimization step, a Graphics User Interface (GUI) has been designed to display a simulated image at a chosen illumination level, with all the characteristics of a real image taken by the sensor with the previous color correction. Simulated images can be a synthetic Macbeth ColorChecker, for which reflectance of each patch is known, or a multi-spectral image, described by the reflectance spectrum of each pixel or an image taken at high-light level. A validation of the results has been performed with ST under development sensors. Finally we present two applications one based on the trade-offs between color saturation and noise by optimizing the CCM and the other based on demosaicking SNR trade-offs.

Keywords: CMOS image sensor, image processing, image quality, color, saturation, noise

1. INTRODUCTION

Over the last few years, the demand of higher resolution for camera phones has increased. Keeping a fixed sensor area while increasing the resolution leads to smaller pixels and less incoming light. However, consumers expect at least the same image quality over generations. This is the reason why the evaluation of CMOS sensor's performance especially in terms of color accuracy and noise is a big challenge for camera phone manufacturers.[1] Since developing a new pixel is very expensive and time-consuming, it will be very useful for sensor developers to evaluate the image quality on simulated images as early as possible in this process. Digital camera imaging simulation tools already exist for the evaluation of image quality[2].[3] The purpose of this study was to developed our own tool focused on sensor simulation and image processing for the development of our new pixel generations.

The Image Quality Evaluation tool (IQE) developed with Matlab as a Graphics User Interface (GUI) includes a module which optimizes the color correction matrix (CCM) and a module which displays a chosen simulated image. Simulated images can be a Macbeth synthetic ColorChecker, a multi-spectral image or a real image of the evaluated sensor at high-light level. These images can be computed with measured or predicted Quantum Efficiency (QE) curves and an illumination and noise model. The tool also allows the extraction of relevant quality metrics such as signal-to-noise ratio (SNR), color error in CIE-L*a*b* space and color saturation.

This paper is organized as follows: in section 2 we will present the image processing pipe used in the tool and in section 2.5, the GUI will be described. Section 3 will explain the validation of the processing by evaluating the noise level on measured and simulated images. Finally in section 4, two studies using the Image Quality Evaluation tool will be highlighted: the trade-offs between color saturation and noise by analysing simulated images with different CCM and the effect of interpolation on SNR and color artefacts.

2. IMAGE PROCESSING PIPE

The proposed image sensor simulation tool includes a color correction matrix optimization unit, an image quality evaluation unit and a simulated image unit which allows images to be displayed on the interface.

2.1 From scene to sensor signal and noise

The incoming parameters which have to be set in the parameters unit of the GUI (Fig. 1) are classified in three main categories: sensor, operating conditions and scene parameters. First of all, the sensor has to be described. Measured or predicted QE curves are downloaded as *.csv file for all color channels. For a given sensor, it is also necessary to set the pixel pitch, the noise model (such as noise floor, PRNU and saturation) and finally the IR filter spectrum of the sensor.

These previous parameters are set for a given sensor, but operating conditions can take different values like the camera lens parameters (aperture denoted $f\sharp$ and optical transmission of the lens denoted T). The integration time τ in s is set as a framerate in fps on the interface.

Finally, the scene is described by choosing the illuminant spectrum $I(\lambda)$ (*e.g.* D65 or the temperature of the blackbody), the scene illuminance E in lux, and the scene reflectance. For the last parameter, either multi-spectral image or real image taken at high-light level can be used. In the first case, a multi-spectral image can be downloaded as a *.mat file (size *e.g.* 33x1019x1337) containing for each pixel a scene reflectance (*e.g.* sample from 400 to 720nm every 10nm *i.e.* 33 points). A synthetic Macbeth ColorChecker (24 or 96 patches) is a particular case of a multispectral image because only the 24 or 96 patches reflectance spectra are needed to reconstruct the Macbeth Chart (and not the reflectance spectrum of every pixel of the chart). In this study, several multispectral images of natural scenes were used.[4] Sensitivity s in $e^-/(lux \cdot s)$ on color channels for each image pixel (i, j) is calculated from the reflectance spectra $R(i, j, \lambda)$ with the following formula, given for a red pixel:

$$s_{Red_{e^-/(lux\cdot s)}}(i,j) = \frac{a^2}{hc} \cdot \frac{T}{4f\sharp^2} \cdot \frac{\int_\lambda R(i,j,\lambda).I(\lambda).QE_{Red}(\lambda).\lambda \mathrm{d}\lambda}{\int_\lambda I(\lambda).V_{Obs}(\lambda)\mathrm{d}\lambda} \tag{1}$$

where a is the pixel pitch (in m), h the Planck's constant (in $J \cdot s$), c the speed of light (in m/s), I the scene illuminant spectrum (in W/m^3), QE the sensor Quantum Efficiency, λ the wavelength (in m) and V_{Obs} the photopic luminous efficiency.

A real image taken by the sensor at high-light level can also be downloaded in order to compute simulated images at different illumination levels. The image must be a *.raw *i.e.* the level of signal in electrons of every pixel on each color plane. This type of input image has been used for the validation of the tool in Section 3. In this second case, the reflectance spectrum of each scene pixel $R(i, j, \lambda)$ isn't known but the reflectance of each scene pixel on color channels (denoted $R_{Red}(i, j)$ for the Red channel) can be deduced from highlight image data. The hypothesis is made that there is a white object in the scene *i.e.* the maximum reflectance is scaled to 100%. This hypothesis leads to the following formula:

$$R_{Red}(i,j) = \frac{s^H_{Red}(i,j)}{M} \tag{2}$$

with

$$M = \max_{i,j}(s^H_{Red_{e-/(lux\cdot s)}}(i,j), s^H_{Green_{e-/(lux\cdot s)}}(i,j), s^H_{Blue_{e-/(lux\cdot s)}}(i,j)) \tag{3}$$

where $s^H_{Red}(i, j)$ is the sensitivity in $e^-/(lux \cdot s)$ of a Red pixel (i, j) of the image taken by the sensor at high-light level. The sensitivities at selected illumination levels for each pixel (i, j) can then be deduced from the calculated reflectance on each pixel.

$$s_{Red_{e-/(lux\cdot s)}}(i,j) = R_{Red}(i,j) \cdot s^{R=1}_{Red_{e-/(lux\cdot s)}} \tag{4}$$

where $s^{R=1}_{Red}$ is the sensitivity in $e-/(lux \cdot s)$ of the Red channel of the sensor calculated in (1) for $R = 1$.

Finally, the signal S in electrons can be deduced with the scene illuminance E in lux, the integration time τ in s, the optical transmission T and the aperture $f\sharp$ with the following equation:

$$S_{Red_{(e^-)}}(i,j) = E_{lux} \cdot \tau \cdot s_{Red_{(e^-/(lux\cdot s))}}(i,j) \tag{5}$$

Noise standard deviation in electrons on color channels is then calculated for each pixel with the calculated signal in electrons and the noise model which takes three standard deviations into account: the noise floor

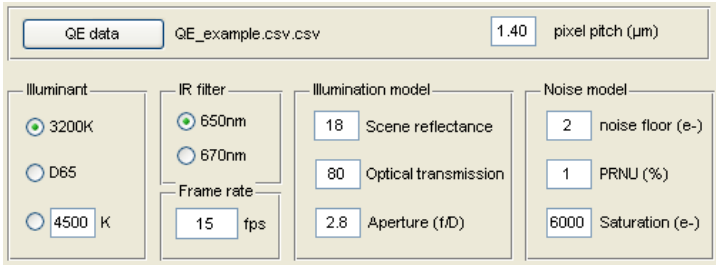


Figure 1: Illustration of the interface of the IQE tool for sensor, illumination and noise parameters

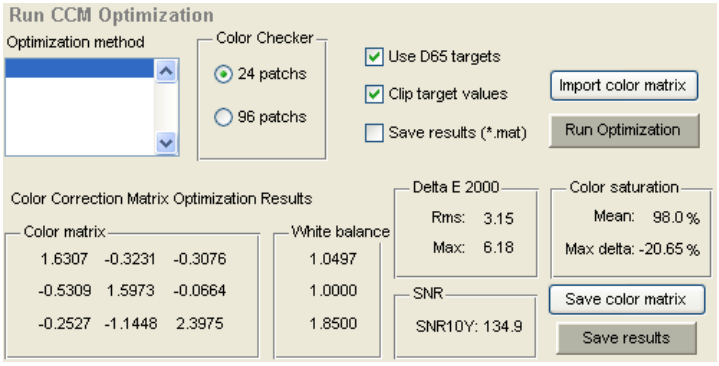


Figure 2: Illustration of the CCM unit

(independant of the signal), the PRNU (Pixel Response Non-Uniformity), proportional to the signal and the shot noise, which is proportionnal to the square root of the signal (6).

$$\sigma_N = \sqrt{(noisefloor)^2 + (PRNU \cdot S)^2 + (\sqrt{S})^2} \tag{6}$$

2.2 Color Correction Matrix Optimization Unit

An important part of the color imaging processing chain to improve image quality is the color correction used to map the captured sensor data to the desired output color space response. The model of full color reconstruction transformation for a standard RGB sensor is expressed in (7):

$$\begin{bmatrix} R \\ G \\ B \end{bmatrix}_{\text{out}} = \overbrace{\begin{bmatrix} c_{RR} & c_{GR} & c_{BR} \\ c_{RG} & c_{GG} & c_{BG} \\ c_{RB} & c_{GB} & c_{BB} \end{bmatrix}}^{CCM} \overbrace{\begin{bmatrix} W_R & 0 & 0 \\ 0 & W_G & 0 \\ 0 & 0 & W_B \end{bmatrix}}^{WB} \begin{bmatrix} R \\ G \\ B \end{bmatrix}_{\text{in}} \tag{7}$$

where the incoming signal is the raw sensor data with the offsets removed to ensure a correct black level. A white balance (WB) correction denoted $\boldsymbol{W}$ is applied to ensure the neutral tone. Finally the color correction matrix (CCM) denoted $\boldsymbol{C}$ is applied to compensate the way the image sensor measures the spectral distributions of the incoming light, and improve color accuracy.

For instance, a simple way to optimize the CCM is the linear squares method (LLSQ). In this case, the CCM is calculated by minimizing the Euclidean distance in RGB output space using linear least-squares fitting. Another well-known method is the non linear least squares method (NLLSQ), which minimizes the Euclidean distance in the non linear CIE-L*a*b* space. It can be performed in the IQE tool using the *fmincon* function of the Matlab Optimization Toolbox. Others methods are available in the GUI to optimize only color errors or to find a compromise between color errors and noise. New methods could be easily added in the future. Section 4 will show the effect of the trade-offs between color accuracy, color saturation and noise on images simulated by the IQE tool with different optimized CCM. It is also possible to import a CCM as a *.csv file if no optimization is required. The CCM unit of the GUI is illustrated in Fig. 2.

2.3 Image Quality Evaluation unit

Once the CCM has been calculated, the tool evaluates parameters in order to determine the performances of the color correction in terms of color error, color saturation and noise. Color errors are evaluated with ΔE_{rms} (8), the Root-Mean-Square error (RMS) of the color error in CIE-L*a*b* space over the N patches of the Macbeth ColorChecker (N=24 or 96).

$$\Delta E_{\text{rms}} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}\Delta E_i^2} \tag{8}$$

Color saturation are evaluated with *Chroma*, the mean of the saturation degradation (SD) over non-grey patches of the Macbeth ColorChecker (9).

$$\text{SD} = 100 \cdot \frac{\sqrt{\left(a^*_{corr}\right)^2 + \left(b^*_{corr}\right)^2}}{\sqrt{\left(a^*_{target}\right)^2 + \left(b^*_{target}\right)^2}} \tag{9}$$

Noise is evaluated by SNR_Y. SNR_Y is the signal-to-noise ratio calculated after color correction but without interpolation of the Bayer pattern. The SNR_Y is computed for a grey color, *i.e.* $W_R S_R = W_G S_G = W_B S_B = S_W$, with S the raw signal and W the white balance for each channel. The signal after color correction and white balance, denoted $S^{'}$, for a grey color is calculated as follows on channel $i = \text{R,G,B}$:

$$\begin{aligned} S^{'}_i &= c_{Ri}W_R S_R + c_{Gi}W_G S_G + c_{Bi}W_B S_B \\ &= (c_{Ri} + c_{Gi} + c_{Bi})W_i S_i \\ &= S_W \end{aligned} \tag{10}$$

The raw noise, denoted N, can be deduced on each channel from the noise model (6). The noise variance terms add quadratically, and so the noise on channel i is calculated as:

$$N^{'}_i = \sqrt{\left(c_{\text{R}i}W_\text{R}N_\text{R}\right)^2 + \left(c_{\text{G}i}W_\text{G}N_\text{G}\right)^2 + \left(c_{\text{B}i}W_\text{B}N_\text{B}\right)^2} \tag{11}$$

The luminance channel is calculated from RGB signals with the NTSC (National television Standards Commitee) standard under illuminant C as in[1] which gives the following equation:

$$\text{SNR}_\text{Y}^{\text{after color correction}} = \frac{S^{'}_Y}{N^{'}_Y} = \frac{\alpha S^{'}_R + \beta S^{'}_G + \gamma S^{'}_B}{\sqrt{(\alpha N^{'}_R)^2 + (\beta N^{'}_G)^2 + (\gamma N^{'}_B)^2}} \tag{12}$$

where $\alpha = 0.299$, $\beta = 0.587$ and $\gamma = 0.114$.

If the bilinear interpolation is used for the Bayer pattern, as it will be the case in this section, the analytic calculus of the SNR_Y after interpolation and color correction (*i.e.* level of noise on displayed corrected image) is possible. Indeed, the standard deviation of the mean of a set of N measurement is $\sqrt{\text{N}}$. During bilinear interpolation, noise of one color is divided by $\sqrt{2}$ or $\sqrt{4}$, depending on the position of the pixel inside the Bayer pattern as illustrated in Fig. 5. This gives the following equation:

$$\begin{cases} N_{R_{\text{interp}}} &= \sqrt{\frac{N_R^2 + \left(\frac{N_R}{2}\right)^2 + 2\left(\frac{N_R}{\sqrt{2}}\right)^2}{4}} \\ N_{G_{\text{interp}}} &= \sqrt{\frac{2N_G^2 + 2\left(\frac{N_G}{2}\right)^2}{4}} \\ N_{B_{\text{interp}}} &= \sqrt{\frac{N_B^2 + \left(\frac{N_B}{2}\right)^2 + 2\left(\frac{N_B}{\sqrt{2}}\right)^2}{4}} \end{cases} \tag{13}$$

The interpolated noise after color correction on Y channel is then calculated as in (11). The SNR_Y after interpolation and after color correction can then be calculated as in (12). A SNR chart of an under development STMicrelectronics pixel is plot on Fig. 3 with SNR_Y after color correction and SNR_Y after interpolation and color correction.

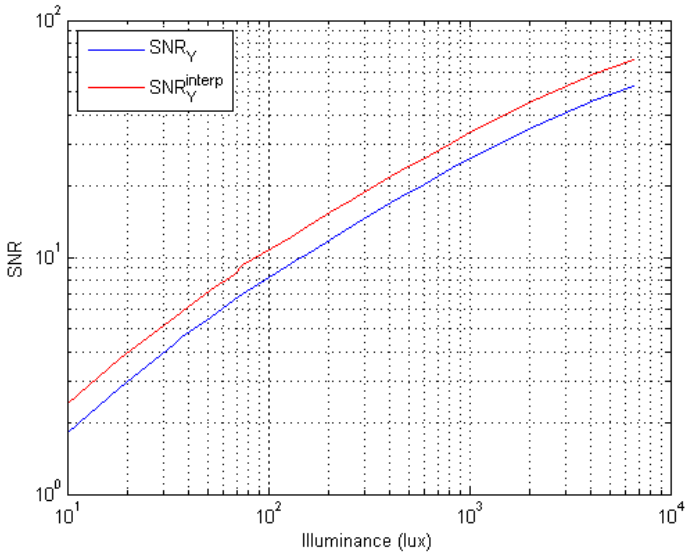


Figure 3: Theorical SNR on Y channel after color correction with and without bilinear interpolation

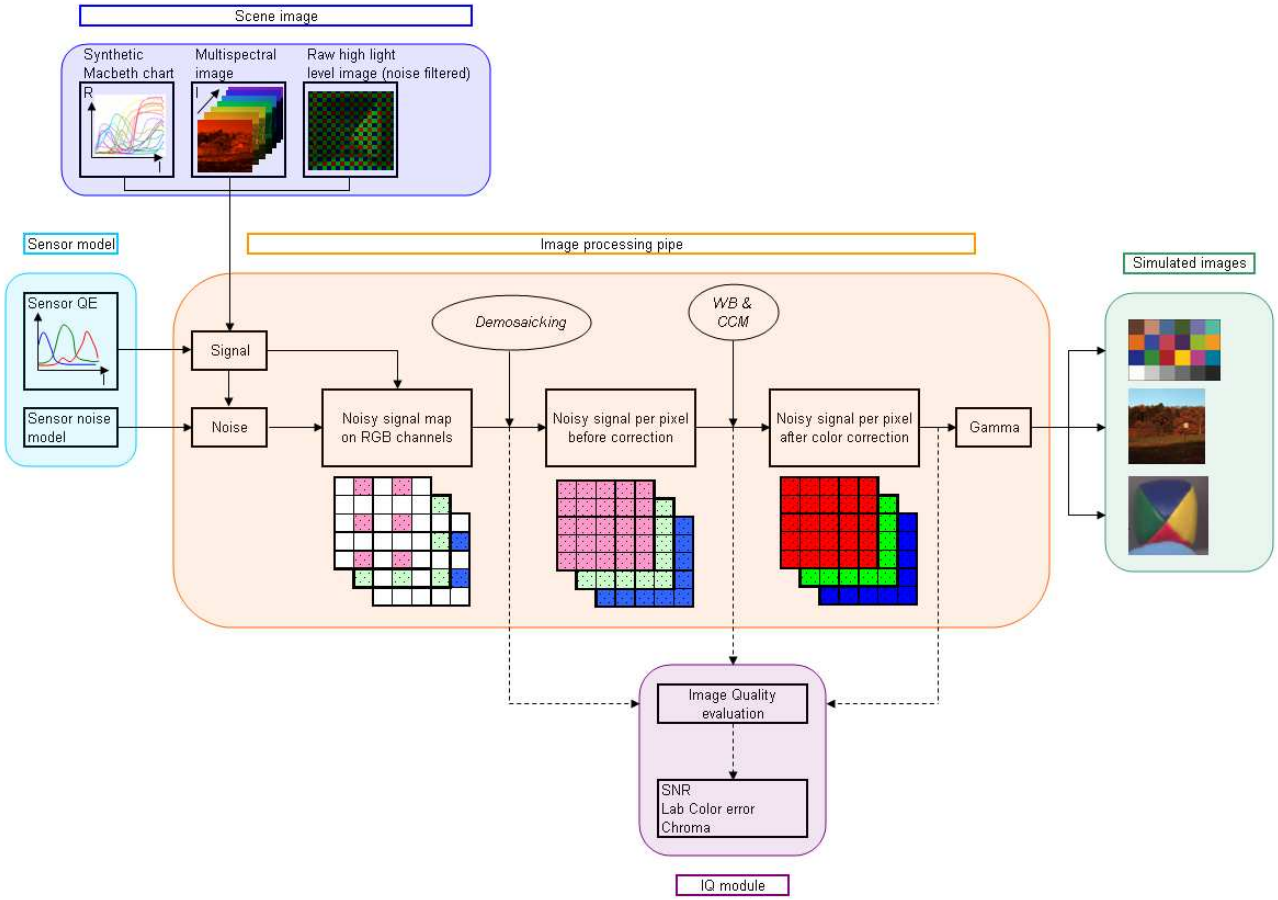
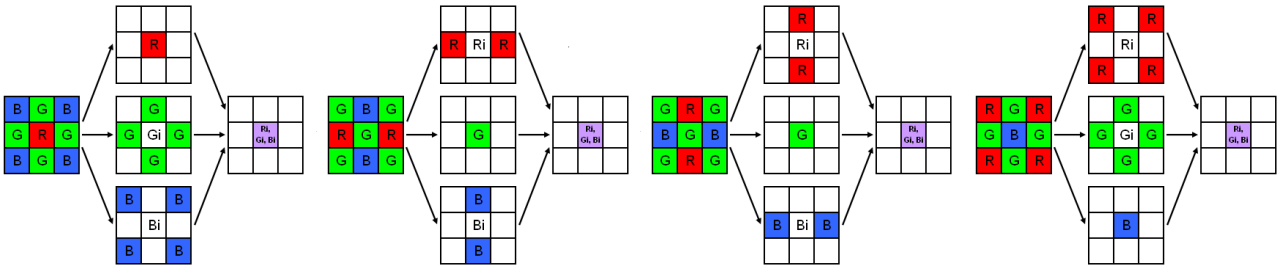


Figure 4: Illustration of the tools workflow



(a) (b) (c) (d)

Figure 5: Illustration of the bilinear interpolation of the Bayer pattern for a (a) Red pixel, (b) Green-Red pixel, (c) Green-Blue pixel, (d) Blue pixel.

2.4 From raw data to RGB image

Once the signal and noise standard deviation per pixel is known for a specified scene/sensor couple (Section 2.1) and once the CCM is chosen (Section 2.2), the processing of the simulated image can start. The full processing pipe is illustrated in Fig. 4 for a sensor with a Bayer pattern. The noise distribution on the matrix of pixels is assumed to be gaussian according to the central limit theorem. As there are other types of noise coming from extrinsic population (RTS noise, white pixels...), this assumption is a limitation of the tool. With this hypothesis, a noisy signal map on color channels is built. This step gives a raw image with simulated noise as would have been a raw image taken with the sensor at the chosen level of illumination.

Then, the usual color processing pipe is applied. The user can choose the demosaicking method among the existing ones described by Malvar[5] and Guntrunk.[6] The more simple is the bilinear interpolation method using the following kernels:

$$F_{R,B} = \frac{1}{4}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix} \text{ and } F_G = \frac{1}{4}\begin{bmatrix} 0 & 1 & 0 \\ 1 & 4 & 1 \\ 0 & 1 & 0 \end{bmatrix} \tag{14}$$

Fig. 5 illustrates the bilinear interpolation for an RGB sensor with a Bayer pattern. In the future, new interpolation methods could be easily added in the IQE tool. After interpolation, the white balance and color correction matrix chosen by the user in the CCM unit is applied. Finally, after scaling, clipping and gamma correction, the simulated image can be displayed on the Graphics User Interface.

2.5 GUI overview

The whole Graphics User Interface is given in Fig. 6. Sensor, operating conditions and scene image parameters are in the blue unit, CCM optimization unit and Image Quality evaluation unit is in red and the image simulation unit is in green. The GUI also displays the target image, the noisy simulated image before color correction, the noisy simulated image after color correction and a target *versus* simulated image mixing both images.

The GUI includes a tool which allows the $\mathrm{SNR_Y}$ to be computed on image. The user selects the two corners of the Region Of Interest (ROI) as illustrated in Fig. 7 and the SNR is computed with the following formula given for a RGB sensor:

$$\mathrm{SNR_Y} = \frac{\alpha\bar{R} + \beta\bar{G} + \gamma\bar{B}}{\sqrt{(\alpha\sigma_R)^2 + (\beta\sigma_G)^2 + (\gamma\sigma_B)^2}} \tag{15}$$

where $\bar{R}$, $\bar{G}$ and $\bar{B}$ are the mean of signal for each channel, σ_R, σ_G and σ_B the standard deviation on the ROI for each channel and with $\alpha = 0.299$, $\beta = 0.587$ and $\gamma = 0.114$.

The GUI also includes tools to set the parameters of displayed image like gamma or to calculate the scaling factor (*e.g.* to set the white patch to 255/255/255 for an 3x8 bits image). The last parameter to be set is the choice of the interpolation method. The effect of the interpolation method on simulated image will be described in Section 4.

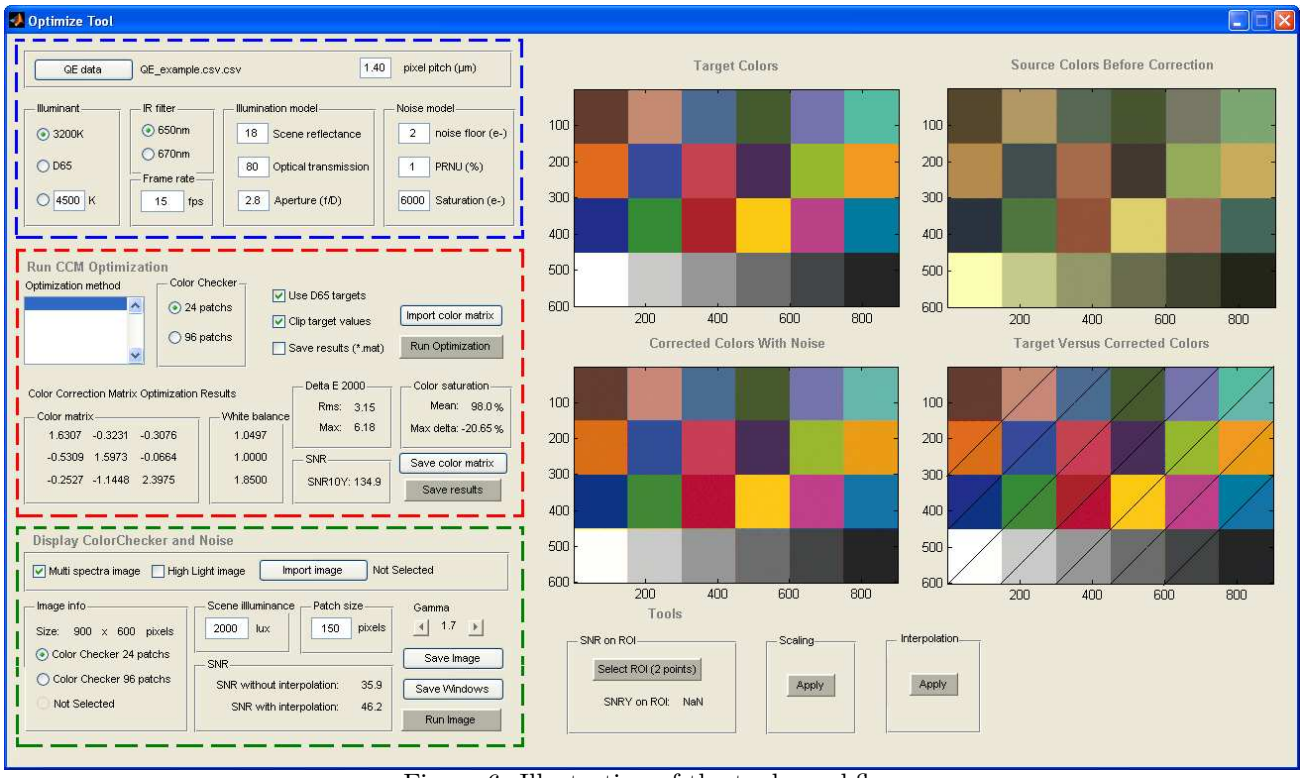


Figure 6: Illustration of the tools workflow

Figure 7: Selection of the ROI on image

3. VALIDATION

In this section, the noise level on real images, synthetic Macbeth ColorChecker, and simulated images from highlight image will be compared to the theory described in Section 2.3. To compare the theory with measured images and simulated images, the SNR tool described in Section 2.5 will be used in Sections 3.1 and 3.2.

3.1 Measurement

Images of a Macbeth ColorChecker were taken with the STMicroelectronics sensor used in Section 2.3 in order to compare noise levels calculated from theory and simulated images. The laboratory has halogen light (3200K) of 2000lux with three neutral densities to adjust the sensor illuminance ($ND = 0.6, 0.6$ and 0.9), which allow to have six levels of illuminance. The set-up is described in Fig. 8. SNR_Y after interpolation and after color correction is computed on image on grey patches n°20, 21, 22, and 23 of the Macbeth ColorChecker. The reflected luminance of grey patches has been measured with a chromameter. The ratio of the reflected luminance over the scene illuminance in the area around the patch (2000lux) gives the reflection coefficient. For grey patches n°20 to 23, measurement gives $R = 59\%, 38\%, 20\%$ and 10%. The SNR_Y after interpolation is usually computed for a reflection coefficient of 18% and a framerate of 15fps. The measured points are corrected to set $R = 18\%$ and $1/\tau = 15fps$. We collected 24 points for the validation. The SNR is computed on the four grey patches of 6 images with different illumination levels with the SNR tool described in Section 2.5. Results are shown in Fig. 9.

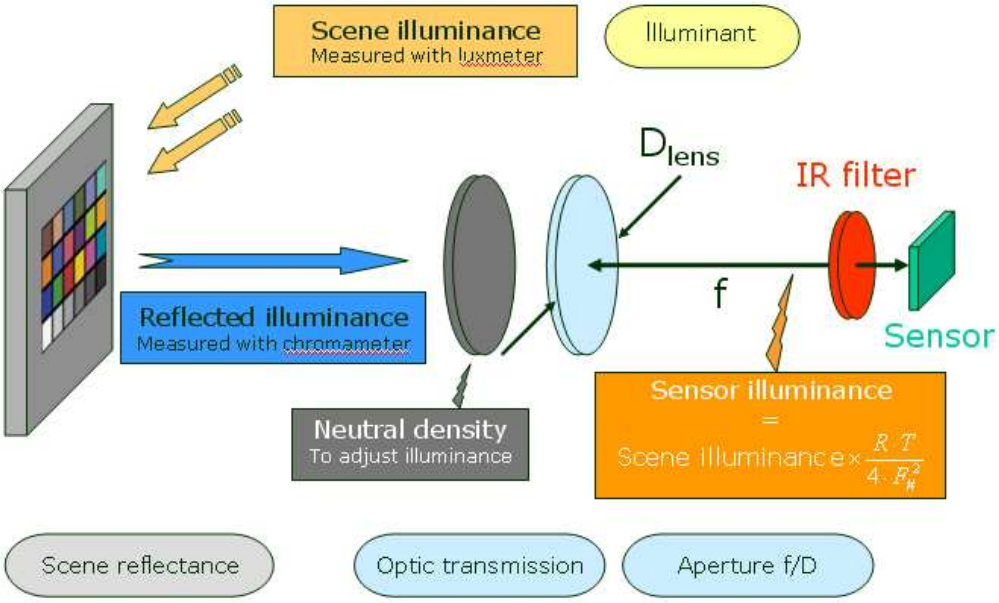


Figure 8: Measurement set-up

3.2 Simulation results

Two kind of images are used for validation: a particular case of multispectral image with the synthetic Macbeth ColorChecker and a high-light image taken by the sensor which is assumed to be noise-free. For the second case, several images have been averaged at highlight level (2000lux) in order to remove temporal noise (temporal average). This raw image (*i.e.* before interpolation and color correction) is processed with the simulation tool at the same scene illuminances than measurements. These scene illumination levels have been chosen with framerate directly set to 15fps in the GUI in order to simulated synthetic Macbeth 24 ColorChecker.

The SNR_Y is computed on both measured and simulated images at each illumination level and on the four grey patches n°20 to 23. Results are shown in Fig. 9. A set of measured and simulated images at lowlight level (30lux) is shown in Fig.10. Zoom on grey patch n°21 shows that the noise appearance is very similar in the three cases.

Fig. 9 shows that all SNR calculations (theoric and on image) are consistant. There are only little discrepancies at lowlight and highlight levels which can be easily explained. At lowlight illumination level, SNR computed on measured and simulated images is a little bit higher than theory. This effect is caused by the gamut clipping of colors which are outside the output color space. At highlight illumination level, SNR on simulated highlight image is below measurement because the highlight image used to simulate others illumination levels images isn't perfect: there is still noise on image due for instance to PRNU. To conclude, there is a good correlation between SNR computed on simulated images and measured images, and with the theory. The only limitation is about the distribution of noise which is assumed to be gaussian whereas others type of noise could appear on image. This effect could be visible at very lowlight illumination conditions.

4. APPLICATIONS

Two direct applications of the IQE tool are the performance of a Color Correction Matrix and the evaluation of the performance of a demosaicking method.

4.1 Influence of Color Correction Matrix Optimization

As described in section 2.2, the IQE tool includes a CCM Optimization unit. Usually, the color reconstruction corrects the mixing of colors caused by the sensor sensitivities. For RGB sensors, the off-diagonal elements of the

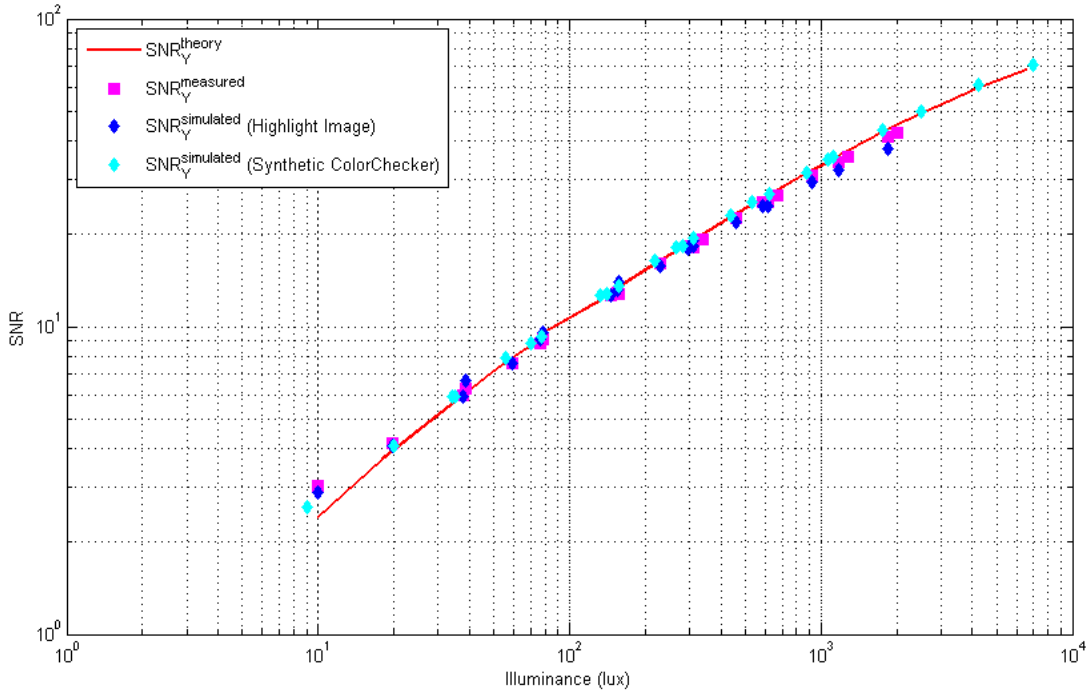
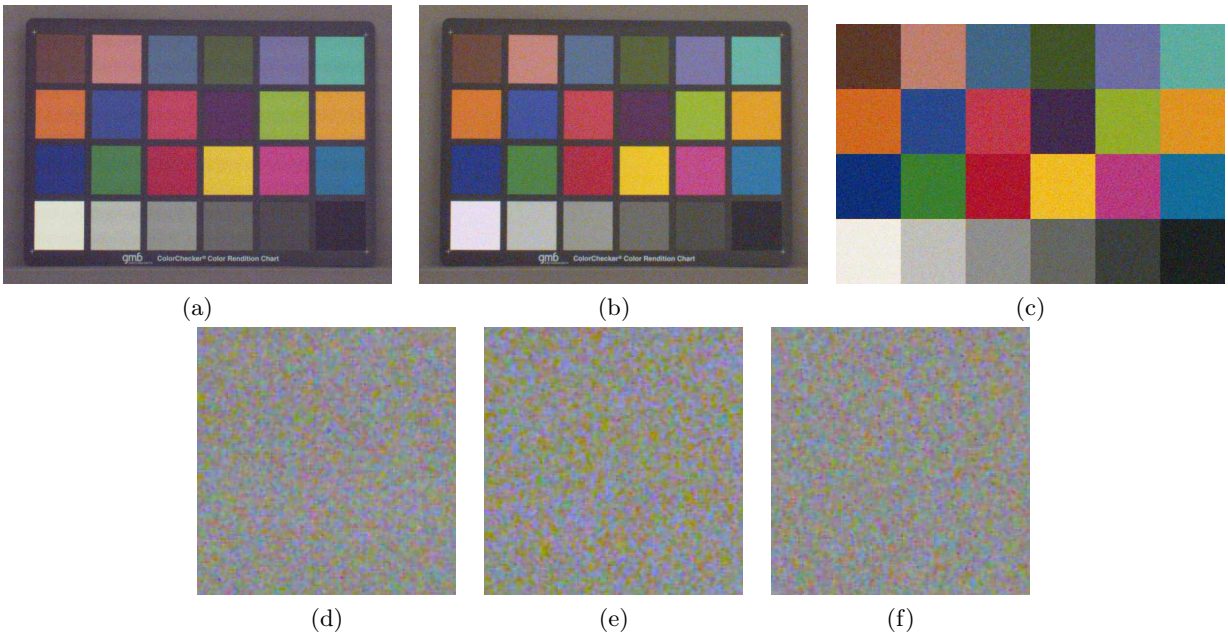


Figure 9: SNR of Y channel versus scene illumination level



(a) (b) (c)

(d) (e) (f)

Figure 10: Validation of noise level on low-light image (at 30lux) (a) Measured image. (b) Simulated image from highlight image. (c) Simulated image from synthetic Macbeth ColorChecker. (d) Zoom of measured grey patch. (e) Zoom of simulated grey patch from highlight image. (f) Zoom of simulated grey patch from synthetic Macbeth ColorChecker.

CCM are most often negative while the diagonal coefficients will be larger that one in order to achieve accurate colors. However, substracting one signal from another leads to a degradation of SNR. If little color correction is needed, the off-diagonal coefficients of the CCM are small and the noise degradation of the color reconstruction is limited. But this is possible at the cost of the degradation of the color fidelity.[7] The user can also choose a method to optimize CCM available in the GUI in order to improve color accuracy and color saturation above all or to find a better compromize with noise level.

A simulation is done for an under development STMicroelectronics pixel under D65 illuminant. Two CCM have been calculated: the first one in order to optimize only color accuracy and the second one to optimize SNR by allowing higher color error and a degradation of color saturation (16).

$$CCM_1 = \begin{bmatrix} 1.84 & -0.65 & -0.19 \\ -0.36 & 1.54 & -0.18 \\ -0.08 & -0.59 & 1.67 \end{bmatrix} \text{ and } CCM_2 = \begin{bmatrix} 1.44 & -0.33 & -0.11 \\ -0.26 & 1.07 & 0.20 \\ -0.20 & -1.33 & 2.52 \end{bmatrix} \tag{16}$$

Simulated synthetic Macbeth 24 ColorChecker have been generated on the GUI at two illumination levels under D65 illuminant. Simulated images are shown in Fig.11. At highlight illumination level, we observe that the first correction matrix is more efficient for color accuracy but at lowlight level the impact of noise is improved on image corrected with the second CCM. The tool can also help the user to choose the CCM, depending on his own constraints or preferences.

4.2 Influence of demosaicking method

Several patterns exist for the filter array. The most common array is the Bayer CFA, which is used in this section. Artificial CZP 1000x1000 test image (Fig.12) is used to emulate the whole processing of the developed tool, including demosaicking step. We examine the performance of three demosaicking methods with the IQE tool. The first demosaicking method is the bilinear interpolation described in sections 2.4 and 2.3. The second one is a bilinear interpolation method but with an interpolation of the Green pixel at the green pixel location with its neighbours, according to the following kernels:

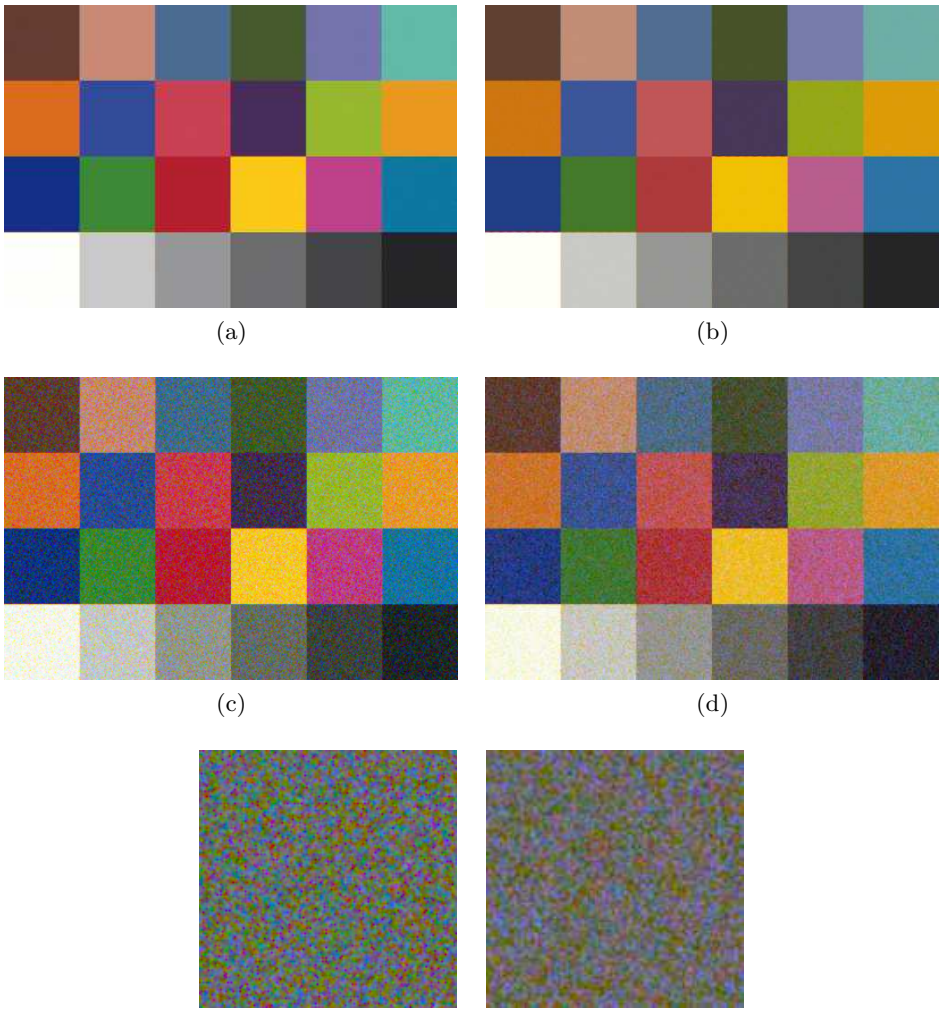
$$F_{R,B} = \frac{1}{4}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix} \text{ and } F_G = \frac{1}{8}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix} \tag{17}$$

The last interpolation method is a high quality linear interpolation described by Malvar.[5] Result images are illustrated in Fig.13 (only one-fourth area with the center of the CZP located at the lower right corner). Interpolation methods are also evaluated in term of noise by simulated images of a synthetic grey patch (Fig. 14) and with the SNR calculation tool available on the GUI (Tab. 1). Fig. 12 shows that the bilinear interpolation generates aliasing effects and color artefacts (color moiré). The modified bilinear generates even more aliasing effects than the bilinear method but the impact of noise on image is reduced whereas with a sophisticated interpolation like Malvar interpolation, aliasing can be significantly reduced with a noise level improved compared to bilinear interpolation (but still with higher noise than the modified bilinear method).

The simulation tool allows the evaluation of demosaicking methods on artificial test images but also on "natural" images thanks to a multispectral image database.

	15lux	30lux	50lux	100lux
Bilinear	5.62	9.08	12.79	19.18
Bilinear modified	7.24	11.50	16.07	23.97
Malvar	6.23	10.10	14.21	21.21

Table 1: Comparison of SNR values for a simulated synthetic grey patch at different illumination levels for three interpolation methods



(a) (b) (c) (d) (e) (f)

Figure 11: Synthetic Macbeth Colorchecker simulated images for different CCM. (a) Image at highlight level with CCM_1. (b) Image at highlight level with CCM_2. (c) Image at lowlight level with CCM_1. (d) Image at lowlight level with CCM_2. (e) Zoom of grey patch at lowlight level with CCM_1. (f) Zoom of grey patch at lowlight level with CCM_2.

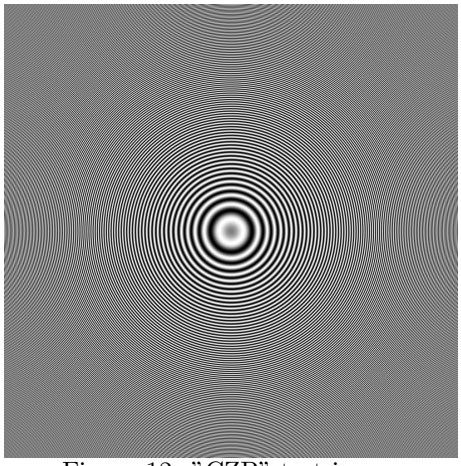
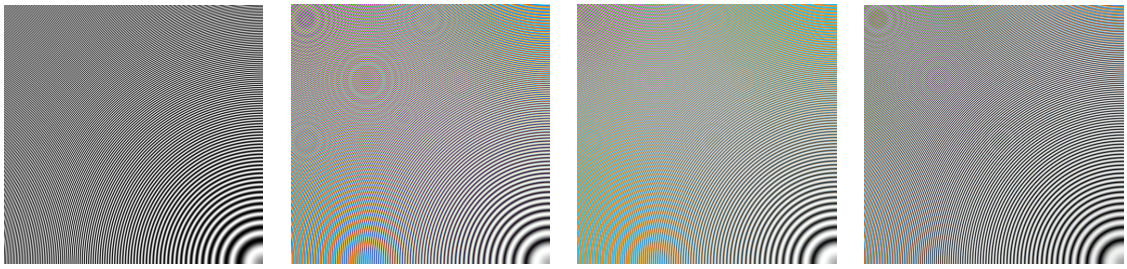
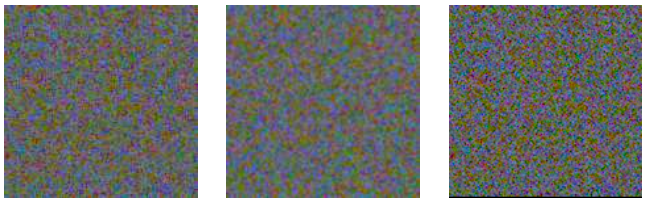


Figure 12: "CZP" test image



(a) (b) (c) (d)

Figure 13: Result images for the "CZP" test image. (a) Original image. (b) Bilinear interpolation. (c) Modified Bilinear interpolation. (d) Malvar high quality linear interpolation



(a) (b) (c)

Figure 14: Result images for a synthetic grey patch at 15lux. (a) Bilinear interpolation. (b) Modified Bilinear interpolation. (c) Malvar high quality linear interpolation

5. CONCLUSION

This paper presented the design, validation, performance evaluation and two applications of our Image Quality Evaluation tool developed under Matlab. As the validation shows that the simulation tool is performing well, it could be very helpful for the development of new pixels which is a very expensive and time-consuming process. It could be also useful for the characterization of current pixel as the extraction of quality metrics (SNR, color error...) from QE data is very simple. The tool will be in constant evolution by adding for instance new CCM optimization methods or new demosaicking methods, allowing to test new color sampling pattern like WRGB.[8] The multispectral database will also become larger in order to evaluate images with a large panel of natural and indoor multispectral images. The next step of the development will be to add QE off axis to have a map with the characteristics of the whole matrix of pixels. In the future, the camera lens could be simulated (OTF) in order to have a complete simulation of the camera module.

REFERENCES

[1] Alakarhu, J., "Image sensors and image quality in mobile phones," in [*Proc. Int. Image Sens. Workshop*], (2007).

[2] Chen, J., Venkataraman, K., Bakin, D., Rodricks, B., Gravelle, R., Rao, P., and Ni, Y., "Digital camera imaging system simulation," *IEEE Trans. Electron Devices* **56**, 2496–2505 (2009).

[3] Farrell, J., Xiao, F., Catrysse, P. B., and Wandell, B., "A simulation tool for evaluating digital camera image quality," in [*Proc. of IS&T/SPIE Electron. Imaging*], **5294**, 124–131 (2004).

[4] Nascimento, S., Ferreira, F., and Foster, D., "Statistics of spatial cone-excitation ratios in natural scenes," *J. Opt. Soc. America* **19**, 1484–1490 (2002).

[5] Malvar, H., He, L.-w., and Cutler, R., "High-quality linear interpolation for demosaicing of bayer-patterned color images," *IEEE Int. Conf. on Acoustics, Speech, and Signal Process. (ICASSP)* **3**, 485–488 (2004).

[6] Gunturk, B., Glotzbach, J., Altunbasak, Y., Schafer, R., and R.M., M., "Demosaiking: color filter array interpolation," *IEEE Signal Process. Magazine* **22**, 44–54 (January 2005).

[7] Vora, P. and Herley, C., "Trade-offs between color saturation and noise sensitivity in image sensors," in [*Proc. Int. Conf. Image Process.*], **1**, 196–200 (1998).

[8] Honda, H., Iida, Y., Egawa, Y., and Seki, H., "A color CMOS imager with 4x4 white-RGB color filter array for increased low-illumination signal-to-noise ratio," *IEEE Trans. Electron Devices* **56**, 2398–2402 (2009).