This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TIP.2017.2713942, IEEE Transactions on Image Processing

Learning the image processing pipeline

Haomiao Jiang, Qiyuan Tian, Joyce Farrell, Brian Wandell
Department of Electrical Engineering, Stanford University
Psychology Department, Stanford University

***Abstract*—Many creative ideas are being proposed for image sensor designs, and these may be useful in applications ranging from consumer photography to computer vision. To understand and evaluate each new design, we must create a corresponding image processing pipeline that transforms the sensor data into a form that is appropriate for the application. The need to design and optimize these pipelines is time-consuming and costly. We explain a method that combines machine learning and image systems simulation that automates the pipeline design. The approach is based on a new way of thinking of the image processing pipeline as a large collection of local linear filters. We illustrate how the method has been used to design pipelines for novel sensor architectures in consumer photography applications.**

***Index Terms*—Local Linear Learned, Camera image processing pipeline, machine learning**

I. Introduction

ADVANCES in small, low-power CMOS image sensors and related optics have revolutionized consumer photography[1], [2], [3], [4], [5]. These technologies have improved dramatically the spatial resolution, dynamic range, and low light sensitivity of digital photography.

In addition to improving conventional photography, these technologies open up many possibilities for novel image systems architectures. The new optics and CMOS sensors capabilities have already motivated novel camera architectures that extend the original Bayer RGB design. For example, in recent years a new generation of architectures have been produced to increase spatial resolution[6], control depth of field through light field camera designs[7], [8], [9], extend dynamic range and sensitivity by the use of novel arrangements of color filters[10] and mixed pixel architectures [11], [12].

To develop these opportunities requires that we innovate on the third fundamental component of image systems, the image processing pipeline. The pipeline is the set of algorithms, including demosaicking, noise reduction, color management, and display nonlinear transforms (gamma curves), that convert the sensor data into a rendered image. Even modest changes to the camera architecture, such as more color pixels into the mosaic [13] or including near infrared detectors [14] can require substantial rethinking of the image processing pipeline. New image processing pipelines, specialized for the new types of cameras, are slow to develop. Consequently, the design of new imaging sensors is far outpacing the development of algorithms that can take advantage of these new designs[15], and the vast majority of image processing algorithms are still designed for sensors that use the classic single plane Bayer RGB spatial sampling mosaic.

In this paper, we describe a new framework that enables image systems engineers to rapidly design image processing pipelines for novel camera architectures that have not yet been built. The general idea of the framework was first proposed by Lansel et al. in 2011[16]. Here, we introduce the framework in the form of a set of software tools that use simulation and learning methods to design and optimize image processing pipelines for these new camera systems. We show how to create a high quality pipeline that can help assess the potential of new camera designs and serve as an excellent starting point to further optimize the camera once it is built.

This paper is organized into three sections that define our contributions. First, we explain the image processing architecture: the input data are grouped by their local features into one of a set of local classes, where locality refers to both position on the sensor array (space), pixel type (color) and response level. The optimal affine transform in each class is learned using camera simulation technology. We refer to this framework as L^3 to emphasize its key principles: Local, Linear and Learned. Second, we assess the performance of L^3 method by comparing the rendered quality with the ones from high-end modern digital cameras. We specifically show that such a collection of affine transforms accurately approximates the complex, nonlinear pipelines implemented in modern consumer photography systems. Third, we illustrate how L^3 method can learn image processing pipelines for new camera architectures.

Several papers incorporate system joint optimization and data-driven learning methods in designing camera image processing pipeline for existing cameras. The L^3 method generates pipelines for cameras that have not yet been built and thus relies on a significantly different approach. In Discussion, after introducing our framework, we describe the relationship between L^3 and these papers.

II. The L^3 Method

The L^3 method comprises two main steps: rendering and learning. The rendering step adaptively selects from a stored table of affine transformations to convert the raw camera sensor data into an image in the target color space (e.g. sRGB). The training step learns and stores the transformations used in rendering.

Conventional image processing pipelines often include nonlinear elements, including thresholding operations and 'gamma' transforms[17], [18]. The L^3 rendering algorithm uses a collection of affine transforms that are applied to data in the different classes. The accuracy of a collection of affine transforms for approximating the image processing pipeline, including nonlinearities such as the 'gamma', can be controlled by the number of classes; this is the conventional

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TIP.2017.2713942, IEEE Transactions on Image Processing

local linear approximation to continuous functions. The challenge of managing discrete transitions, such as the transition from the linear response region to saturation, represented by thresholds can be managed by selecting proper category boundaries. As a practical matter, there is rarely a strong need to specify a precise and sharp category boundary in natural image processing.

In the following, we explain these two steps of L^3 method in detail. We explain the method using the example of a camera sensor with RGBW (red, green, blue and white) color filter array. The method can be applied to other designs, which we describe in the Results section.

A. Rendering

The rendering pipeline begins with the sensor data in the spatial neighborhood of a pixel, $n(x, y, p)$. We illustrate the method for a 5×5 neighborhood, so that the neighborhood comprises 25 pixel responses (Figure 1). Each pixel is classified into one of many classes, c, based on its local features: the identity of the pixel, the mean response level of the local patch, the spatial variance of the neighborhood, and pixel saturation status, etc. The total number of classes can be estimated as the product of the number of categories for every feature. For example, suppose that there are 4 types of pixels, and we categorize the mean neighborhood intensity into 10 response levels. This produces 40 different classes. If we further classify each neighborhood into textured (high variance) or uniform, then there will be 80 classes ($1 \leq c \leq 80$). Additionally, pixel saturation is also a condition that requires careful management. In some designs, one type of pixels (e.g., the W pixel in an RGBW) can saturate while the RGB pixels will be well within their operating range. It is important to create separate classes that account for pixel saturation. Allowing for this increases the number of effective classes, typically by a factor of three for the RGBW case. The number of classes is a hyperparameter of the L^3 framework and can vary in different applications.

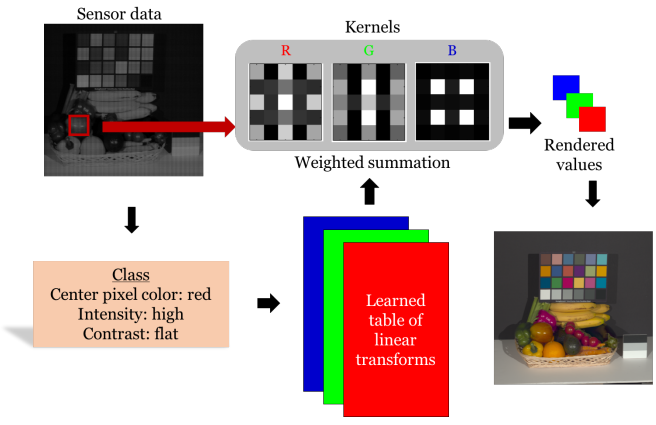

Fig. 1. **The L^3 rendering method.** In L^3 rendering, each pixel is classified by its local features into one of a few hundred classes (e.g. a red pixel, with high intensity, surrounded by a uniform field). Each class has a learned affine transform (stored in a table) that weights the sensor pixel values in order to calculate the rendered outputs. Hence the rendered output of each pixel is calculated as an affine transform of the pixel and its neighbors.

For each class c and output channel r, we retrieve an affine transform to map the 25 values into one rendered value. The affine transforms for each class and rendered output channel, $T(c, r)$, are pre-computed and stored. The rendered output, $o(x, y, r)$ is the inner product of the stored affine transform and the neighborhood data (augmented with a 1 to account for the affine term).

$$o(x, y, r) =< T(c, r), n(x, y, p) >$$

There are several practical considerations in L^3 implementation: the content and application dependent class definitions, the target color representations (monochrome, highly saturated) and even the fitting model in each class (affine or polynomial). These choices impact the algorithm efficiency and precision, and we describe experiments evaluating different choices in Results. These choices are part of the design process when implementing the L^3 method.

Finally, the computations for each pixel are independent, so that the architecture can be highly parallelized. We have performed preliminary tests of implementing these methods with a graphical processing unit (GPU) and other ISP hardware to shorten rendering time[19]. These preliminary analyses suggest that L^3 is capable of performing real-time rendering for high quality images on mobile devices that contain modern image processing hardware.

B. Learning

It is challenging to create algorithms for cameras that are being designed, rather than cameras that already exist, because of limitations in obtaining sensor data[20]. We solve this problem by using image systems simulation tools to model the proposed camera architecture and to create the training data[21], [22]. The Image Systems Engineering Toolbox (ISET) begins with a spectral representation of the scene and includes simulations of the optics and sensor. The simulator has been validated against data from several real devices [23], [24].

Once we have simulations of the critical data, we have a variety of options for how we select the transforms that map the sensor data into the rendered images. We describe our approach to simulation and transformation derivation in the next two sections.

1) Training data: Realistic camera training requires scene spectral radiance data sets that are representative of the likely application. We have obtained scene spectral radiance and accumulated examples from a number of public scene radiance data sets[1]. In addition, we have also used spectral computer graphics methods to simulate a variety of scene spectral radiance images. These simulations can produce scene spectral radiance examples for training that establish special spatial resolution and color requirements that extend what we would be likely to find by merely sampling a range of natural images.

A further advantage of the simulation method is that the desired output image and sensor data are precisely aligned, at the pixel level. For each input scene spectral radiance we calculate the calibrated color representation (e.g. CIE

[1]http://www.imageval.com/scene-database/

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TIP.2017.2713942, IEEE Transactions on Image Processing

XYZ) and the sensor response at each pixel (Figure 2). Such correspondence is very difficult or even impossible to obtain from empirical measurements.

Finally, because the training pairs are produced by simulation, we can produce many examples of scenes and measurement conditions. Through simulation we can control the properties of the ambient lighting, including its level and spectral power distribution. We can perform simulations with a wide range of optical parameters. Training can be performed for special content (e.g., faces, text, or outdoors scenes).

The pairs of images produced by the simulation methods can provide a virtually limitless collection of input data to the training system.

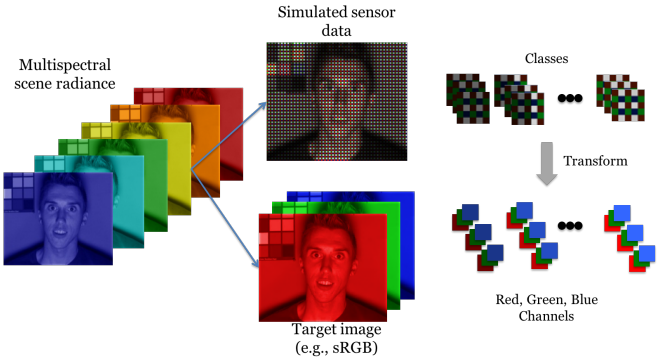

Fig. 2. **Preparing training data with image systems simulation.** Starting from scene spectral radiance data (left), we compute two aligned representations. Top: the sensor responses in a model camera system. Bottom: the target rendered image from scene radiance. The simulated sensor data at each pixel and its neighborhood (patch) is placed into one of several hundred pre-defined classes, based on properties such as the pixel type, response level, and local image contrast. For each class, the pairs of patch data and target output values are used to learn an affine transformation from the data and the target output value.

2) Transform optimization: The purpose of the training is to calculate a set of transforms that optimally map the sensor responses to the target output image and minimize the empirical risk. Stated formally, our task is to find for each class, C_i, a transformation, T_i such that

$$\underset{T_i}{\text{minimize}} \sum_{j \in C_i} \mathcal{L}(y_j, x_j T_i)$$

Here, x_j is a N-dimensional row vector containing the j-th patch data from the sensor; y_j are the three corresponding image values in the target space (e.g., R,G,B). $\mathcal{L}$ is the loss function (error) between the target image and the transformed sensor data. In consumer imaging applications, the visual difference measure CIE ΔE_{ab}[25] can be a good choice for the loss function. For other applications, regularized RMSE is widely used. In nearly all photography applications, the transformation from sensor to rendered data is globally non-linear. L^3 method approximates the global nonlinear transformation with a collection of affine transforms for appropriately defined classes C_i.

At first, we solve the transforms for each class independently. This problem can be expressed in the form of ordinary least-squares by stacking the corresponding x_j and y_j data into matrices, X and Y. To avoid noise magnification, we use ridge regression and regularize the kernel coefficients. That is

$$T_i = \underset{T_i}{\text{argmin}} ||Y - XT_i||^2 + \lambda ||T_i||^2$$

The data from each patch are placed in the rows of X; the regularization parameter is λ, and Y is the output in the target color space as an $N \times 3$ matrix. We have experimented using several target color spaces, including the XYZ, CIELAB and sRGB representations, and we can find satisfactory solutions in all cases. The closed-form solution for this problem is given as

$$T_i = (X^T X + \lambda I)^{-1} X^T Y$$

The computation of T_i can be further optimized by using singular vector decomposition (SVD) of X. That is, if we decompose $X = UDV^T$, we have

$$T_i = V \text{diag}(\frac{D_j}{D_j^2 + \lambda}) U^T Y$$

The regularization parameter (λ) is chosen to minimize the generalized cross-validation (GCV) error [26].

Once the transforms for each class are defined, it is possible to review them for properties that reflect our prior knowledge, such as continuity over the sensor input space, symmetry and uniformity (see Discussion). The software implementation includes methods to check for these conditions and to bring the transforms into alignment with this knowledge.

III. Results

In this section, we characterize the performance of the L^3 method and illustrate how it can be used to generate image processing pipelines for novel camera architectures. First, we analyze whether the L^3 rendering method based on many affine transforms is sufficient to approximate the performance of commercial image processing pipelines (Nikon and DxO). Second, we use L^3 to learn a collection of transforms for non-standard color sensor designs (RGBW and RGB-NIR).

A. The L^3 pipeline closely approximates high quality Bayer CFA algorithms

The L^3 pipeline is designed to be computationally efficient and to learn algorithms for novel arrays. Before applying the method to new designs, it is important to analyze whether the simple pipeline is capable of supporting high quality rendering expected from camera systems that have been optimized. To evaluate any performance limits, we compared how well the L^3 rendering algorithm can approximate the image processing pipeline embedded in a high quality commercial camera.

1) Accuracy: In one experiment, we used an image dataset of 22 well-aligned raw and JPEG natural images from a Nikon D200 camera. We used 11 randomly selected images for training the local linear transforms and the other half for testing (cross-validation). The L^3 pipeline parameters were set to use 50 luminance levels for the four pixel types (red, blue and two types of green), for a total of 200 classes. We analyzed

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TIP.2017.2713942, IEEE Transactions on Image Processing

the data with 5×5 local patches (affine transforms of 26 parameters). The effect of patch size is discussed later.

Figure 3 (upper left) shows a typical example of an image produced by the Nikon processing pipeline and the corresponding image produced by the L^3 method (lower left). By visual inspection the images are very similar; the largest visual differences are the blue sky and the bush in the lower left. We use a perceptual space-color visual difference metric S-CIELAB [27] to quantify the visual difference. Perceptual metrics require specifying the display and viewing distance, and for the S-CIELAB calculation we assumed the images are rendered and viewed on a calibrated LCD monitor with 96 dpi at viewing distance of one meter. For the 2592×3872 Nikon images, the horizontal field of view is 58.7 deg. The ΔE_{ab} error image (lower right) and histogram (upper right) are typical of the test data: the mean S-CIELAB ΔE_{ab} is 1.74 for the 11 test images (PSNR 40.36), which is a small visual difference.

These experiments show that the collection of L^3 transforms approximates the full commercial rendering produced by this Nikon D200 camera for this collection of outdoor images.

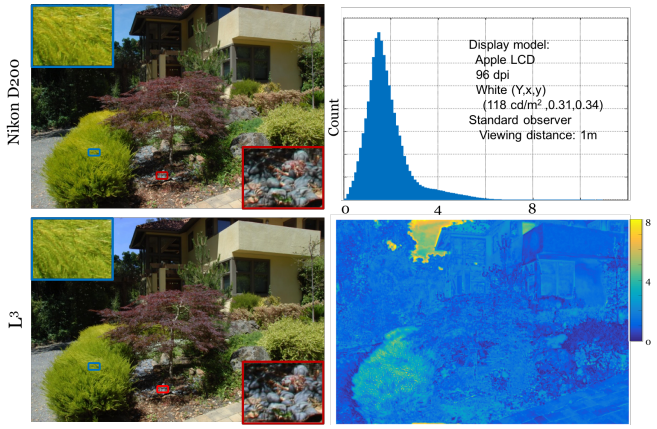

Fig. 3. **Approximating the Nikon pipeline with L^3 local linear transforms.** At the left, we compare camera RGB (upper left) with L^3 rendered image (lower left). The image at the lower right measures the perceptual difference between the two images using S-CIELAB ΔE_{ab} values for each pixel. The metric is calculated assuming that the images are displayed on a known, calibrated monitor (see inset text, upper right). The histogram of errors is shown on the upper right. The mean error is 1.84, the peak error is near 8, and the standard deviation of the ΔE_{ab} values is 0.9. The L^3 transforms were learned from one set of 11 images. This image is from an independent data set of 11 images. The errors reported for this image are typical for all the images in the independent test set.

We also applied the L^3 method to learn the local linear transforms that approximates a commercial image processing pipeline (DxO). In this experiment, 26 raw and RGB image pairs are analyzed. The RGB images are generated with DxO Optics Pro using parameters tuned by a professional photographer, Dave Cardinal. This dataset includes multiple types of cameras and the content spans various natural scenes, human portraits, and scenic vistas.

Each of the individual images can be well-approximated by the L^3 method. The images at the left and middle of Figure 4 show a typical example of the DxO image and the L^3 image. The image at the right is the S-CIELAB visual difference

Fig. 4. **Approximating DxO pipeline with L^3 local linear transforms.** We compare the image rendered by a DxO pipeline with parameters tuned by a professional photographer (Left) with one rendered with the L^3 method (Middle). The perceptual error is measured in S-CIELAB ΔE_{ab} image (Right). The error calculations are based on the same monitor as in Figure 3.

error for each pixel. The mean S-CIELAB ΔE_{ab} value for this image is 1.458 and the accuracy of the approximation is similar to what we achieved for the Nikon processing pipeline.

The experts settings vary significantly as the scene and camera types change; for example, in some scenes the expert chooses more sharpening and for others a softer focus is preferred. Hence, no single set of L^3 transforms applies to all of the images. The broad issue of selecting L^3 transforms for specific acquisition conditions or rendering aesthetics is further analyzed in Discussion.

The DxO and Nikon D200 experiments show that the L^3 kernel regression approach is sufficient to approximate the transforms embedded in commercial rendering products.

2) The transforms: Next, we examine the properties of the learned transforms that approximate the Nikon D200 processing pipeline. When learning the L^3 transforms, a few parameters must be selected: the number and distribution of response levels, and the size of the local patch.

The transforms change substantially with the response level (Figure 5). At low levels, the weights are relatively equal across the entire patch, and there is less selectivity for color channels. At high response levels the weights are concentrated on the appropriate pixel type. For example, when the center pixel is green and the output channel is also green, at high response levels the transform weights are concentrated on the central green pixel. At low light levels the transform weights at non-central green pixels are relatively high.

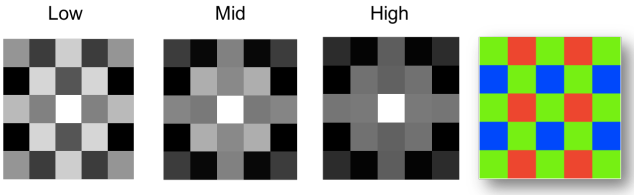

Fig. 5. **L^3 transforms depend on response level.** The three monochrome images show the relative weights of the transforms that convert data at a G pixel centered patch into the green output channel. The patch size is 5×5 and the three images show the weights that are learned for a class defined by Low (left), Mid (second left) and High (third left) response levels. The spatial distribution of the weights becomes more concentrated at the central pixel as the mean response level increases. The CFA pattern of the full 5×5 patch is shown at the right.

3) Response level spacing: The learned transform weights change more rapidly at the lower response levels compared to the higher levels. For this reason, it is efficient to use a

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TIP.2017.2713942, IEEE Transactions on Image Processing

logarithmic spacing of the mean response levels that define the classes; that is, we use more finely spaced classes at the low response levels than the high response levels.

Through simulation, we can evaluate the difference between linear and logarithmic spacing of the mean response levels. We analyzed the Nikon D200 data using different numbers of mean response levels (Figure 6). The levels were spaced either linearly or logarithmically. To achieve the same image quality (e.g. 2 ΔE), logarithmic spacing is equivalent to linear spacing with about 50% more number of classes. As the total number of classes becomes large, say 50 for this example, the performance of the two spacing methods is very similar. We expect that the specific parameter values, such as number of response levels, will differ slightly for different optics and sensor combinations. But the principle of using logarithmic spacing is likely to hold across conditions.

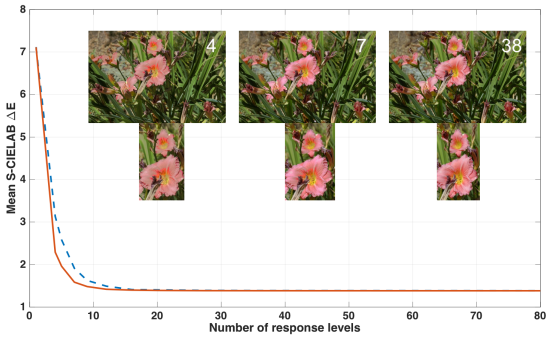

Fig. 6. **The effect of class selection on rendered image quality.** The graph shows how the color error (mean S-CIELAB ΔE) of the rendered image declines as the number of response level classes increases. The two curves show the effect for linear (blue dashed) and logarithmic (red solid) spacing of the mean response levels in the patch. This inset images are the renderings for 4, 7 and 38 mean response level classes (logarithmic spacing). The rendering of the flower (zoomed view) changes substantially as the number of levels increases, and the mean color reproduction error declines significantly as the number of mean response levels increases to about 15. There is only a very small advantage for the logarithmic spacing when using a small number of classes, and no advantage beyond about 12 levels.

4) Patch size selection: There is a significant computational cost to increasing patch size. Changing the patch size from a 5×5 to 7×7 (9×9) approximately doubles (triples) the number of coefficients and computational cost. Moreover, if the training dataset is limited, the risk of overfitting can increase with the patch size.

For the Nikon and DxO approximations, we found little improvement in the approximation as we changed the patch size beyond 5×5. Figure 7 shows the mean ΔE values of S-CIELAB on the 11 test images. The plotted data points show the mean error for individual test images, and the solid red line shows the average of all 11 images. The mean ΔE values decrease very slightly as the patch size increases from 5×5 to 7×7 and there is no further decrease as the patch size increases to 9×9. It might be more effective to create classes based on other features (e.g. nonlinear or global features) rather than increasing the patch size.

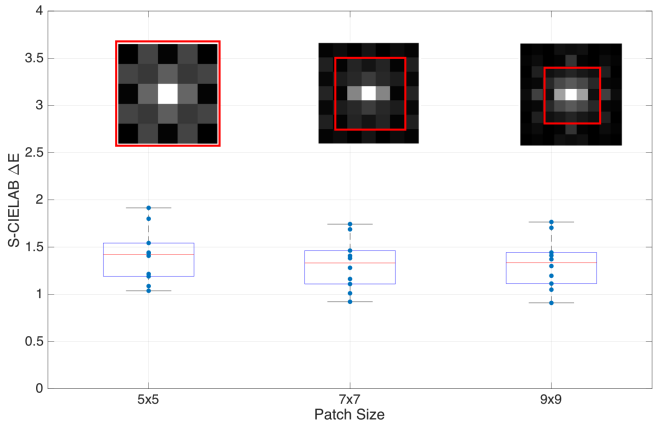

Fig. 7. **The effect of patch size on rendered image quality.** The boxplot shows the distribution of mean S-CIELAB ΔE errors for the 11 test images. The inset monochrome images are examples of the learned affine transforms. The red rectangle in each image denotes the center 5×5 region. The weights learned by the transform outside of the 5×5 patch are small, and increasing the patch size has little effect on the color accuracy.

B. L^3 pipelines for novel color filter arrays

The ability to automate the process of learning the rendering pipeline is an important objective of the L^3 method. In this section, we use L^3 to generate the image processing pipelines for two challenging CFA designs.

We first apply L^3 method to generate an image processing pipeline for camera with RGBW sensor. The CFA repeating pattern of RGBW sensor contains both RGB and clear (white) pixel. Adding a white pixel extends the operating range of the camera and makes the camera usable in low light imaging. The key challenge in designing a pipeline for this sensor is the large mismatch in the sensitivity between the W and RGB pixels[28]

We then consider a CFA that combines RGB channels with a near infrared (NIR) channel. There is a great deal of interest in adding an NIR channel to support applications of depth sensing. The NIR channel, which is invisible to the human eye, can be used to measure a projected NIR pattern for depth estimation. The NIR pixels do not contain significant information for image reproduction, so that this design reduces the pixel count significantly. We analyze concerns how to best render an image for the RGB-NIR design, and how this rendering depends on factors such as pixel size and optics.

1) RGBW: In this experiment, we simulated an RGBW camera with exactly one R, G, B and W in each CFA repeating pattern. The spectral transmittance of the color filters and other key sensor and lens parameters of the camera simulation are shown in Figure 8. The relative sensitivity of the W to the RGB pixels and the spatial arrangement of the four pixel types differs between vendors [29], [30], and this simulation represents one of a range of possible choices.

We apply simulation methods (Figure 2) to prepare the training data to learn the local linear transforms. Specifically, we first calculate the sensor response of multispectral scenes using the ISET camera simulator. Then, we compute the ideal (CIE XYZ) value at each pixel location. The local patches

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TIP.2017.2713942, IEEE Transactions on Image Processing

Optics	
f/#	4
focal length	3.9 mm
Sensor Array	
Rows, columns	[576, 706]
Exposure time	0.1 s
Height, Width	[0.8, 1] mm
DSNU	0.8 mV
PRNR	0.8%
Analog gain	1
Analog offset	0 V
Pixel	
Width, Height	[1.4, 1.4] um
Fill factor	0.45
Dark voltage	6.7 mV/s
Read noise	0.507 mV
Conversion gain	0.12 mV / e-
Voltage swing	0.64 V
Well capacity	5404 e-

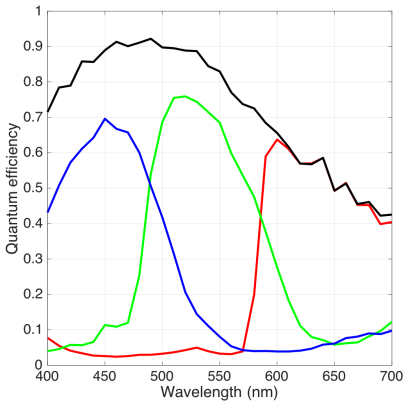

Fig. 8. **The simulated RGBW camera system.** The table lists key properties of the optics, pixel and sensor array used in the simulation. The curves show the spectral quantum efficiency of the four different pixel types. The inset shows the CFA pattern for a 5×5 patch centered on a green pixel.

are classified by mean response levels and the center pixel type. Affine transformations are learned for each class, using ridge regression with the regularization parameters set using a cross-validation error minimization.

Figure 9 shows examples of the learned transforms for this RGBW camera in four different classes, from low response levels to near saturation. The low mean response class transforms (Low) heavily weight data from the W pixel, presumably because the signal-to-noise ratio (SNR) of the W pixel is substantially higher at generally low response levels. As the response level increases (Mid), the W pixel SNR advantage is less important than the color information provided by the RGB pixels, and the weights redistribute to using more of the data from these pixels than the white pixel.

As the mean level increases further, the W pixel saturates (High) the L^3 transforms discounts the W responses. Notice that to estimate either the R or G responses, the central G pixel is heavily weighted. Even when the output is R, the value of the G pixel at the same position is very important. When the estimated output is G, the adjacent R pixels have low weights.

At the highest mean intensity levels (noted as Saturate), both the G and R pixels saturate in some cases. In this case the L^3 method reduces the weights on the G pixels and the largest weights are assigned to the B pixels.

By using many different response levels, not just the four shown here, we obtain a smooth transition from the W-dominated to the RGB-dominated domain to the R/B dominated.

Notice that for the Mid and High response levels, the red output depends significantly on the the G center pixel response. The algorithm learns that there is a strong spatial and color correlation between the G center pixel value and output red channel[31]. This confirms the previous observation that the value at the G center pixel is useful in predicting the red output value, and the linear transform quantitatively estimates the proper amount that G pixel should contribute to the red channel output. However, as the center G pixel starts to saturate, the transforms assign them lower weights. At this mean intensity level, R pixel may also saturate and L^3 learns to assign more weights to B pixels.

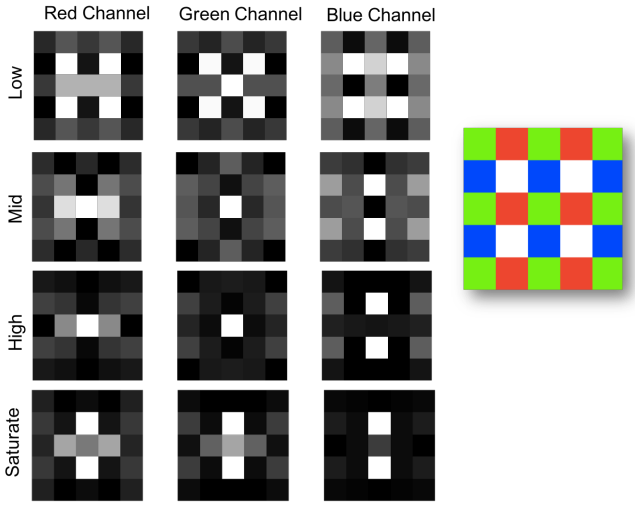

Fig. 9. **The transforms learned for the RGBW sensor.** The transforms learned for Low, Mid, High and Saturate response levels are shown in the rows. The columns show the red, green and blue output channels. The color filter array is shown at the right. At Low response levels, the highest weights are on the W pixels. As the response level increases, the weights of the RGB pixels become larger. When most W and some of the G pixels saturate, the R and B weights are the largest.

We analyzed two important aspects of the L^3 performance: color accuracy and image resolution. We performed this analysis by training on a standard data set and then testing performance on targets designed to analyze color and resolution.

To assess color accuracy, we compute the CIELAB ΔE values between the L^3 rendered image and ideal values of a standard Macbeth color checker. The L^3 pipeline for the RGBW sensor achieves a mean ΔE of 1.7, which is very accurate. We then replaced the W pixel with a green pixel to form a traditional Bayer CFA pattern. The mean ΔE difference is similar (ΔE= 1.65). Hence, the L^3 method learns how to incorporate the W pixels to achieve an accurate color reproduction.

To assess resolution, we calculated the spatial frequency response (SFR) using the ISO 12233 slanted bar method[32]. This method measures the image intensity in the region near the edge of a slanted bar. The intensity measurements are converted from a spatial representation into a modulation transfer function (MTF). The metric is defined by the spatial frequency at which the MTF drops to half of its peak value (MTF50); higher MTF50 values imply higher spatial resolution. The value of the MTF50 depends on a number of system features, including the optics and the pixel size. For the system described in Figure 8 (f/#=4, pixel size 1.4 um), the MTF50 is 154.80 cycles/mm, which is close to the upper bound imposed by the optics 186.70 cycles/mm.

We assessed the spatial resolution for different lens (diffraction-limited) and sensor combinations (Figure 10a). The radius of the blur circle grows proportionally with the f/#, blurring the optical irradiance at the sensor. When the pixel size is small and the f/# is large, the spatial resolution, assessed by MTF50, is limited by the optics. When the pixel

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TIP.2017.2713942, IEEE Transactions on Image Processing

size is large and the f/# is small, the spatial resolution is limited by the pixel size.

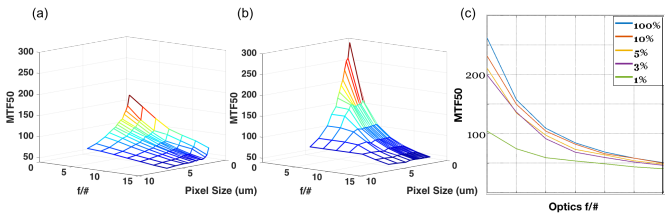

Fig. 10. **Dependence of spatial resolution (MTF50) on system and acquisition conditions.** (a) The MTF50 depends on both the f/# of the diffraction-limited optics and the pixel size. The mesh shows the MTF50 for a range of these parameters. The transforms are learned from images of human face images. (b) MTF50 of the transform as function of f/# and pixel size when trained with a spatial resolution test chart. (c) MTF50 as a function of f/# plotted separately for classes at different percentages of the maximum response level.

The exact value of the MTF50 also depends on the training data. In Figure 10a, the L^3 transforms are trained with a collection of human faces which do not have sharp edges. To evaluate the upper limit of spatial resolution, we trained with a scene containing only a spatial resolution chart and again used the MTF50 metric to evaluate spatial resolution (Figure 10b). In this case, the L^3 transforms achieve almost double the spatial resolution in the optimal region (small pixel size, small f/#). In other regions, the benefits of using spatial resolution target is much smaller.

In addition to the optics and pixel size, scene illumination level and exposure conditions also matter (Figure 10c). We evaluated the MTF50 for a fixed pixel size using over a range of response levels and f/#s. For small f/# when the resolution can be very high, the L^3 transforms differ between the response levels. At low response levels, the transforms reduce noise by placing significant weights on most of the pixels in the patch. Thus the MTF50 is relatively low. When the response level is high, the MTF50 is much higher. For large f/# the loss is dominated by the lens and thus the difference in the MTF50 between low and high response levels is minimal.

2) RGB-NIR: Next, we use the L^3 method to design an image processing pipeline for an RGB near infrared (NIR) sensor. The main application for including an NIR channel is to acquire extra information that is used in combination with an IR projector to estimate depth [33], [34]. There are several ways to implement NIR sensors. One approach removes one of the two green filters and the IR cutoff filter that is normally placed on the sensor surface. Modern color filters pass significant amounts of IR, so this approach allows NIR photons to enter the same pixels that are used by the visible light[35]. The image processing pipeline must estimate and remove these correlated IR signals, which introduces noise and reduces sensor dynamic range.

An alternative approach, recently implemented by Panasonic, selectively blocks IR photons in the RGB channels placing metal within the pixel[36]. In this approach, each CFA block contains a pixel of each type, and the RGB channels are protected from absorbing NIR photons by an infrared cut filter layer which includes a stack of silicon oxide and titanium oxide films.

The Panasonic RGB-NIR differs significantly from RGBW because the NIR channel captures very little information in the visible range. However, there may be useful image reproduction information in the NIR channel, and in any case the pipeline must run effectively even if the imaging component only uses the RGB channels.

In this example, we simulate the Panasonic design and the key parameters are shown in Figure 11a. Learning the L^3 pipeline for this sensor requires a hyperspectral scene radiance dataset that extends into the NIR. We used a dataset of 12 scenes that were measured from 415 to 950 nm. The data set includes calibration targets, fruits, buildings and natural scenes.

Optics	
f/#	4
focal length	3.9 mm
Sensor Array	
Rows, columns	[72, 88]
Exposure time	Auto exposure
Height, Width	[0.8, 1] mm
DSNU	0 V
PRNR	0
Analog gain	1
Analog offset	0 V
Pixel	
Width, Height	[2.8, 2.8] um
Fill factor	0.75
Dark voltage	1 mV/s
Read noise	1 mV
Conversion gain	0.1 mV / e-
Voltage swing	1 V
Well capacity	10000 e-

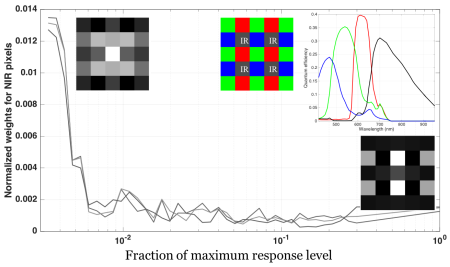

Fig. 11. **Key parameters of RGB-NIR camera systems (left) and the normalized weights for NIR pixels as a function of response levels (right).** The three curves show the normalized weights for NIR pixels towards three output channels as a function of response levels. The two inset gray images are the learned transforms for a G centered patch towards blue output under low and high response range. In low light, the NIR pixels are slighted used while in high response levels, there is almost no contribution from NIR pixels in the RGB rendered image. The CFA pattern and effective quanta efficiency of the pixels (CFA transmittance included) are shown as the inset image on the upper right.

We used the L^3 method to learn a set of local linear transforms for the RGB-NIR sensor. Figure 11 shows the normalized weights of NIR pixels for G centered patches at different response levels. For low response levels, the L^3 transforms assign significant weight on the NIR pixels; the weights on these pixels become very small at high response levels.

We evaluated the color accuracy for the RGB-NIR sensor as we did for the RGBW case. The cross-validated median CIELAB ΔE value is 2.87. If we black out the IR pixel and solve for the transforms again, the cross-validated median ΔE increases slightly to 3.08. This shows that the NIR data can slightly help estimate color. If we replace the IR sensor with a green pixel, forming a conventional Bayer pattern, the median ΔE value is 2.69. Hence, the RGB-NIR acquires some information in the invisible range at a cost of slightly worse RGB image. The simulations and L^3 algorithm quantify how to take advantage of this information in natural indoor images.

We also evaluated the spatial resolution of the RGB-NIR design using the MTF50 measure. For the Panasonic camera, the MTF50 value is 137 cycles/mm (optics f/#=4, 2.75 um pixel size with proper exposure). Replacing the NIR pixel with

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TIP.2017.2713942, IEEE Transactions on Image Processing

a G pixel, increases the MTF50 to 151 cycles/mm. This shows that the RGB-NIR design has slightly lower spatial resolution than the matched Bayer design.

IV. Discussion

We quantified how accurately the L^3 architecture renders images. We used a perceptual error metric to compare the L^3 rendering with two different commercial rendering methods (cf. Nikon and DxO). The L^3 architecture, based on kernel regression approximation, produces images that are about 2 ΔE (spatial CIELAB) from the original.

We also showed that the L^3 architecture can automatically generate rendering pipelines that are optimized for novel sensor designs. We implemented and evaluated image rendering pipelines for a sensor designed to extend the dynamic range by including a clear pixel (RGB-W), and also for a sensor that includes a set of pixels capable of measuring near infrared patterns projected into the scene to estimate depth (RGB-NIR).

In this section, we review the relationship between L^3 and other new ideas in image processing. Then, we discuss some design choices of the L^3 method that arise in practice. First, we discuss the need to create multiple tables of transforms that should be applied in different acquisition conditions (color balancing; exposure duration; imaging conditions). Second, we describe how we account for knowledge about the transformations that should be incorporated to improve the learned transforms. Third, we consider the choice of a target space for the rendering. Fourth, we discuss how L^3 might be extended to address additional modules in the rendering pipeline.

A. Related work

There are several themes in the literature that share common elements with the L^3 method. At the most general level, Milanfar et al. proposed using the multidimensional kernel regression and non-parametric learning methods in image processing and reconstruction [37], [38]. That work generalizes several image processing methods, including bilateral filtering, and denoising algorithms, under kernel regression schema. The general principles of kernel regression - classifying local data and interpolating measurements - can be applied to a range of imaging problems, such as learning super-resolution kernels [39].

The proposal that is closest to our work comes from Khabashi et al. [20] Similar to our work, they describe a nonparametric regression tree models, together with Gaussian conditional random fields, to demosaic raw sensor response. They classify the data near each pixel into one of a large number of classes; the class is based on the color filter type of the pixel and a measurement of the local edge direction in the 5×5 neighborhood of the pixel. For each class they use example camera data to find a quadratic transform that maps pixel data to the rendered value.

In addition to these similarities in the approach, there are several significant differences. First, Khabashi et al. perform their training by processing mosaicked sensor data from existing cameras to estimate the ground truth, full resolution. In contrast, L^3 makes extensive use of image systems simulation technology to create sensor data for training. The use of simulations enables L^3 to support analyses for cameras that do not yet exist. Second, Khabashi et al. classify the sensor data based on pixel type and spatial orientation of the data in the of 5×5 patches within the sensor mosaic. L^3 classifies the sensor data based on response level and local contrast. Relying on response level is important because the different levels have very different noise characteristics, and the optimal transform differs significantly between low and high response levels (Figure 9). Finally, Khabashi et al. focus on the demosaicking stage of the process, while L^3 training replaces additional pipeline components, including denoising and color transforms that map the sensor data directly into the target color space.

Another related set of ideas concerns the development of image processing pipelines that are based on joint optimization across optics, sensor, and display. An example is from Stork and Robinson [40] who offered a theoretical foundation for jointly optimizing the design and analysis of the optics, detector, and digital image processing for imaging systems. They optimized the image processing pipeline for different lenses, assuming a monochrome sensor. The L^3 method incorporates lens properties into the simulation, so that the table of transforms accounts for the specific lens properties. Different tables are generated as the lens properties (e.g., aperture, f/#) are varied. Hence, the L^3 method is also a co-design approach in the sense that the learned rendering parameters depend on the whole system, including the optics and sensor.

Heide et al. also conceive of the image processing pipeline as a single, integrated computation[41]. They suggest a framework (FlexISP) in which they model the relationship between the sensor data and a latent image that represents the fully sampled sensor data prior to optical defocus. They propose to estimate the latent image from the sensor data by solving an optimization problem; the optimization accounts for both the data and a set of natural image priors. Hence, a key difference is that Heide et al. calculate a solution separately for each sensor acquisition, while L^3 pre-computes a fixed table of transforms and applies this table to all images. Another difference is that Heide et al., like Khabashi et al., begin their calculations with the sensor data. In contrast, L^3 simulates a camera system beginning with scene radiance, accounting for properties of the optics, pixel, and sensor. By working from scene spectral radiance data, L^3 can be used to create pipelines at the earliest stages of the design process, when no hardware implementation yet exists. The simulations also make it possible to optimize L^3 parameters for different types of scenes, some of which may be difficult to create in a laboratory environment.

Convolutional sparse coding (CSC) methods share some features of the L^3 method. CSC representations begin with a full image representation and decompose the image into a linear sum of component images [42]. Each component is the convolution of a single, usually small, kernel with a sparse feature map (most entries are zero). The CSC learns local features from the input training images, and the core calculations are linear. However, the CSC learning methods and target applications of the differ significantly from L^3. First, CSC learns kernels and feature maps that decompose

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TIP.2017.2713942, IEEE Transactions on Image Processing

an image into separate components. L^3 performs the reverse computation; it starts with partial sensor data and creates a complete image. Second, the learning methods are different. The CSC kernels are learned through advanced bi-convex optimization methods that require substantial computational power. The affine (or simple polynomial) transforms learned by L^3 use prior knowledge of the camera and training about the camera design but very simple optimization methods. In summary, L^3 is an architecture for designing new image processing pipelines and efficient rendering; CSC is a technique for feature extraction and applications to learning image features for machine vision applications and computational photography, such as inpainting.

B. Multiple transform tables

Training L^3 for the range of settings (optics, sensor properties) of a single camera, leads to different transform tables (Figure 10). For mobile devices, the camera settings do not vary extensively. When only a few number of possible settings are available, the best solution might be to pre-compute and store a table of transforms for each setting.

In addition to hardware settings, there is the question of whether the L^3 training would produce different tables as we change the scene characteristics. We have explored important case: how the tables depend on the spectral power distribution of the illumination Germain. In this case we found that the tables learned for different illuminant are similar enough so that we can render the image with a single table and then apply a 3×3 color transform to render a color-balanced image.

There are many other different scene characteristics that remain to be explored. The optimal table of transforms may depend on factors such as image motion, image content, optics parameters such as depth of field and focus. It is possible that multiple tables will be required or that a single set of tables followed by simple transforms will suffice for most conditions.

C. Applying prior knowledge to the transform table

L^3 training depends on the specific training data. This can be used to our advantage, say if we know we want to optimize the rendering for a particular condition and target (e.g., human faces, outdoors). The reliance on specific samples produces transforms that may differ in some small way from our expectations. For example, in many cases we expect the transforms to be left-right symmetric. Further, we expect that transforms at nearby response levels will be similar to one another. We developed functions that can be applied to the learned table of transforms to enforce these expectations (prior knowledge).

- **Symmetry**
  When the underlying color filter array of each patch is symmetric in some way (up-down, left-right, transpose, circular), we also expect the learned transform to be symmetric. Imposing symmetry helps avoid over-fitting to the training data. We transform general transforms into symmetric transforms by creating symmetric versions of the learned transform and then using the average.
- **Smoothing and interpolation**
  The L^3 coefficients change relatively smoothly as the response level increases. We smooth the transforms by fitting a spline to each of the coefficients and then replacing the coefficients with the value of the smooth spline. We use the same method to interpolate for transforms in classes that have a small amount or insufficient training data.
- **Uniformity**
  A uniform scene should be rendered as a uniform image. This requirement, unlike the previous two, requires operating on the transforms from different pixel types. Specifically, the sum of the transform weights of each pixel type must be equated between the classes of different center pixels.

D. Choosing the target space for rendering

We emphasized L^3 consumer photography applications: the sensor data are transformed to a rendered image. Even for consumer photography applications, there are multiple choices for the target rendering space. We have trained L^3 instances to transform into various colorimetric spaces (e.g., CIE-XYZ), and we have also trained to transform into nonlinear representations (e.g., sRGB, CIELAB). Because the global L^3 transformation is nonlinear (though locally linear), the input data can be effectively transformed to most representations that are smoothly related to colorimetry representations. Choosing the target space is equivalent to choosing the error function. For example, rendering to CIELAB space minimizes the point-by-point color error.

E. Space-varying

The current L^3 formulation does not account for the position of the center pixel within the sensor. Thus, the algorithm is effectively shift-invariant. There are two important aspects of the rendering pipeline that are space-varying. First, lens shading produces an uneven illumination level from the center to the periphery of the sensor. Second, geometric distortion of the image (e.g., barrell distortion) varies the relationship between the position of the pixel within the sensor and its appropriate position in the output image.

The pixel type and response level are used to identify a class, and we do not include the position of the pixel within the sensor. Hence, the L^3 method is fundamentally space-invariant. Correcting for the shift-varying components (lens and geometric distortion) are shift-varying. The parameters of these features are determined by the main taking lens and are independent of the image processing pipeline. Hence, like illuminant correction, for the moment we think it is best to perform these steps separately rather than extending the number of classes and setting L^3 the task of accounting for the position-dependent factors.

F. Reproducibility

The data and methods necessary to reproduce the figures are available from the Stanford Digital Repository[2].

[2]http://purl.stanford.edu/bk962py0458

V. Conclusion

We introduce a methodology to automate the design of image processing pipelines. The image-processing pipeline is approximated as a locally linear operation in which sensor data are grouped into various classes, and the data from a class are rendered by a linear transform into the rendered image. We illustrate that the local transforms can produce high quality rendered images. Then, we use image systems simulation to create the table of affine transforms for novel camera designs, including sensors with clear or near infrared pixels. We evaluate the performance of these tables using color metrics (S-CIELAB), spatial resolution metrics (MTF50), and simulations of captured images. Hence, this paper combines image systems simulation technology and modern computational methods into a methodology that creates image processing pipelines.

VI. Acknowledgements

Steve Lansel. Olympus Trisha. Dave Cardinal. Qualcomm.

References

[1] Gamal, Abbas EI., *Trends in CMOS image sensor technology and design.*, Electron Devices Meeting. IEDM'02: 805-808, 2002.

[2] Blanco-Filgueira, Beatriz, Paula Lopez Martinez, and Juan Bautista Roldan Aranda., *A Review of CMOS Photodiode Modeling and the Role of the Lateral Photoresponse.*, 2016.

[3] Fossum, Eric R, and Donald B Hondongwa. , *A review of the pinned photodiode for CCD and CMOS image sensors.*, IEEE J. Electron Devices Soc 2.3 (2014): 33-43.

[4] Chang, Jen-Tsorng. , *Compact camera module with lens array.*, US 8289409 B2, 2012

[5] Reshidko, Dmitry and Sasian, Jos. , *Current trends in miniature camera lens technology*, SPIE Newsroom 02/2016; DOI: 10.1117/2.1201602.006327

[6] Longoni, Antonio et al., *The transverse field detector (TFD): a novel color-sensitive CMOS device.*, Electron Device Letters, IEEE 29.12 (2008): 1306-1308.

[7] Georgiev, Todor et al., *Lytro camera technology: theory, algorithms, performance analysis.*, Proc. SPIE 7 Mar. 2013: 86671J.

[8] Bishop, Tom E, and Paolo Favaro., *The light field camera: Extended depth of field, aliasing, and superresolution.*, Pattern Analysis and Machine Intelligence, IEEE Transactions on 34.5 (2012): 972-986.

[9] Marwah, Kshitij et al., *Compressive light field photography using overcomplete dictionaries and optimized projections.*, ACM Transactions on Graphics (TOG) 32.4 (2013): 46.

[10] Baranov, Pavel, and Olesya Drak. *A new color filter array with high light sensitivity and high resolution properties.*, Young Researchers in Electrical and Electronic Engineering Conference (EIConRusNW), 2015 IEEE NW Russia 2 Feb. 2015: 17-23.

[11] Nayar et al. *Generalized Assorted Pixel Camera Systems and Methods.*, US 2015/0070562 A1, March 12, 2015.

[12] Yasuma, Fumihito et al. *Generalized assorted pixel camera: postcapture control of resolution, dynamic range, and spectrum.*, Image Processing, IEEE Transactions on 19.9 (2010): 2241-2253.

[13] Monno, Yusuke, Masayuki Tanaka, and Masatoshi Okutomi. *Multispectral demosaicking using guided filter.*, IS&T/SPIE Electronic Imaging, 2012

[14] Tang, Huixuan et al. *High Resolution Photography with an RGB-Infrared Camera.*, Computational Photography (ICCP), 2015 IEEE International Conference on 24 Apr. 2015: 1-10.

[15] Nayar, Shree K. *Computational cameras: Redefining the image.*, IEEE Computer 39.8 (2006): 30-38.

[16] Lansel, Steven Paul. *Local Linear Learned Method for Image and Reflectance Estimation.*, Stanford PhD Dissertation, 2011.

[17] Poynton, Charles A. *SMPTE Tutorial:Gamma and its Disguises: The Nonlinear Mappings of Intensity in Perception, CRTs, Film, and Video.*, SMPTE journal 102.12 (1993): 1099-1108.

[18] Guo, Hongwei, Haitao He, and Mingyi Chen. *Gamma correction for digital fringe projection profilometry.*, Applied Optics 43.14 (2004): 2906-2914.

[19] Tian, Qiyuan, and Haomiao Jiang. *Accelerating a learningbased image processing pipeline for digital cameras.*, IS&T/SPIE Electronic Imaging 2015.

[20] Khashabi, Daniel et al. *Joint Demosaicing and Denoising via Learned Nonparametric Random Fields.*, Image Processing, IEEE Transactions on 23.12 (2014): 4968-4981.

[21] Farrell, Joyce E et al. *A simulation tool for evaluating digital camera image quality.*, Electronic Imaging 2004 19 Dec. 2003: 124-131.

[22] Farrell, Joyce E, Peter B Catrysse, and Brian A Wandell. *Digital camera simulation.*, Applied optics 51.4 (2012): A80-A90.

[23] Farrell, Joyce, Michael Okincha, and Manu Parmar. *Sensor calibration and simulation.*, Electronic Imaging 2008 14 Feb. 2008: 68170R-68170R-9.

[24] Chen, Junqing et al. *Digital camera imaging system simulation.*, Electron Devices, IEEE Transactions on 56.11 (2009): 2496-2505.

[25] Luo, M Ronnier, Guihua Cui, and B Rigg. *The development of the CIE 2000 colourdifference formula: CIEDE2000.*, Color Research & Application 26.5 (2001): 340-350.

[26] Golub, Gene H, Michael Heath, and Grace Wahba. *Generalized cross-validation as a method for choosing a good ridge parameter.*, Technometrics 21.2 (1979): 215-223.

[27] Zhang, Xuemei, and Brian A Wandell. *A spatial extension of CIELAB for digital colorimage reproduction.*, Journal of the Society for Information Display 5.1 (1997): 61-63.

[28] Parmar, Manu, and Brian A Wandell. *Interleaved imaging: an imaging system design inspired by rod-cone vision.*, IS&T/SPIE Electronic Imaging 18 Jan. 2009: 725008-725008-8.

[29] Kumar, Mrityunjay et al. *New digital camera sensor architecture for low light imaging.*, Image Processing (ICIP), 2009 16th IEEE International Conference on 7 Nov. 2009: 2681-2684.

[30] Wang, Jue, Chao Zhang, and Pengwei Hao. *New color filter arrays of high light sensitivity and high demosaicking performance.*, Image Processing (ICIP), 2011 18th IEEE International Conference on 11 Sep. 2011: 3153-3156.

[31] Hel-Or, Yacov. *The canonical correlations of color images and their use for demosaicing.*, HP Laboratories Israel, Tech. Rep. HPL-2003-164R1 (2004).

[32] ISO 12233:2014 *Photography – Electronic still picture imaging – Resolution and spatial frequency responses*, International Organization for Standardization (2014).

[33] Jeong, Jae-chan et al. *High-quality stereo depth map generation using infrared pattern projection.*, ETRI Journal 35.6 (2013): 1011-1020.

[34] Smisek, Jan, Michal Jancosek, and Tomas Pajdla. *3D with Kinect.*, Consumer Depth Cameras for Computer Vision (2013): 3-25.

[35] Fredembach, C. and Lu, Y. and Susstrunk, S. *Camera design for the simultaneous capture of near-infrared and visible images*, 2013, US 8462238 B2

[36] Watanabe, H. *Solid-state imaging device and camera module*, 2015, US Patent App. 14/426,810

[37] Takeda, Hiroyuki, Sina Farsiu, and Peyman Milanfar. *Kernel regression for image processing and reconstruction.*, Image Processing, IEEE Transactions on 16.2 (2007): 349-366.

[38] Milanfar, Peyman. *A tour of modern image filtering.*, IEEE Signal Processing Magazine 2 (2011).

[39] Zhang, Kaibing et al. *Learning multiple linear mappings for efficient single image super-resolution.*, Image Processing, IEEE Transactions on 24.3 (2015): 846-861.

[40] Stork, David G, and M Dirk Robinson. *Theoretical foundations for joint digital-optical analysis of electro-optical imaging systems.*, Applied Optics 47.10 (2008): B64-B75.

[41] Heide, Felix et al. *FlexISP: a flexible camera image processing framework.*, ACM Transactions on Graphics (TOG) 33.6 (2014): 231.

[42] Heide, Felix, Wolfgang Heidrich, and Gordon Wetzstein. *Fast and flexible convolutional sparse coding.*, Computer Vision and Pattern Recognition (CVPR), 2015 IEEE Conference on 7 Jun. 2015: 5135-5143.

[43] Germain, Francois G et al. *Efficient illuminant correction in the local, linear, learned (L3) method.*, IS&T/SPIE Electronic Imaging 27 Feb. 2015: 940404-940404-7