# Optimized Construction of ICC Profiles by Lattice Regression

*Eric Garcia, Maya Gupta; Dept. of Electrical Engineering, University of Washington; Seattle, WA*

## Abstract

*We focus on a recently proposed regression framework termed* lattice regression*, as applied to the construction of multi-dimensional color management look-up tables from empirical measurements. The key idea of lattice regression is that the construction of a look-up table should take into account the interpolation function used in its final implementation. Lattice regression solves for the look-up table (LUT) that minimizes the error of interpolating the empirical measurements (training samples) and regularization is added to promote smoothness and enable extrapolation. The main contribution of this paper is the proposal and analysis of using the thin-plate regularizer for lattice regression to produce smooth and accurate color transformations. Experiments with a consumer inkjet and laser printer show that the proposed regularizer obtains similar accuracy to the previously-proposed (and more complicated) combination of Laplacian and global-bias regularizers, and that both can create significantly more accurate and smoother results than a state-of-the-art locally linear approach.*

## Estimation for Color Transformations

It is common to transform colors between colorspaces and devices by first characterizing the transformation with a look-up table (LUT), and then interpolating the LUT to transform new color values. The LUT can be specified in an International Color Consortium (ICC) profile, which is part of a standard color management workflow. In this paper we consider the estimation question of how to best specify a LUT to characterize a color transformation.

Recently, we proposed a new approach called *lattice regression* for learning a LUT from sample pairs of input and output colors that takes into account that the LUT is interpolated at runtime [6, 7], but that preliminary work required a complicated regularizer in order to perform well in practice. In this paper, we show that similar results can be achieved using the more elegant thin-plate regularizer [8].

For simplicity, and because it is a common case, we refer throughout to the input color space as CIELAB and the output color space as RGB, but everything applies to transformations between any two colorspaces. We call the LUT defined over the input color space the lattice $a$, which has vertices $\{a_1, a_2, \ldots, a_m\}$, where each vertex $a_i \in$ CIELAB. Then the goal is to learn the output RGB color for each $a_i$. We treat each of the output color planes separately so that the output color corresponding to vertex $a_i$ is the scalar color value $b_i \in [0, 255]$.

Printing a calibration target results in data pairs $\{x_i, y_i\}$, where $x_i \in$ CIELAB, $y_i \in [0, 255]$ and $i = 1...n$, where $n$ is the number of color patches in the calibration target.

The standard approach to learning a color transformation is to first fit a function to the data pairs $\{x_i, y_i\}$ to produce an estimated function $\hat{f}$ that maps CIELAB to $[0, 255]$. Then, the estimated function $\hat{f}$ is evaluated at the the vertices of the lattice. That is, the estimated output value for $a_i$ is $\hat{b}_i = \hat{f}(a_i)$. Mathematically, the estimated function $\hat{f}$ is chosen to minimize the sum of some loss function, usually squared error as shown here:

$$\hat{f} = \arg\min_{f \in \mathscr{F}} \sum_{i=1}^{n} (f(x_i) - y_i)^2, \tag{1}$$

where $\mathscr{F}$ is a set of allowed functions. For example, standard least-squares linear regression is (1) with $\mathscr{F}$ being the set of all linear functions $f(x) = \beta^T x + \beta_0$. Neural nets, decision trees, and support vector machine regression and other standard approaches to regression can be written as (1) for different choices of $\mathscr{F}$, sometimes with a different loss function than squared error, and sometimes with additional regularization terms that are independent of the data but add a preference for smoother functions in the function class $\mathscr{F}$. Many such regression methods have been compared for learning ICC profiles [1, 2, 11]. Bala's experiments showed that the best accuracy was obtained with a local linear regression [2]. Roughly 20% improvement in median accuracy was achieved over local linear regression using the enclosing neighborhood definition and ridge regularization [9], and further accuracy improvements were shown if the ridge regularizer was replaced by a Tikhonov regularizer [10].

Another metric in the creation of ICC profiles is whether smooth images input to the LUT appear smooth after the LUT interpolation [14, 15].

## Lattice Regression

The problem with the standard approach described by (1) is that the LUT will be interpolated at run-time to actually estimate output color values for an image, and this interpolation step is not taken into account in fitting the function by (1). For example, if one used the LUT to estimate the appropriate RGB color for a training sample $x_i$, the output of the lookup table is not $\hat{f}(x_i)$, rather it is the interpolation of $x_i$ from the vertex-output pairs $\{(a_i, \hat{b}_i)\}$ that surround $x_i$. Thus, the error being minimized in (1) does not represent the true error produced by the LUT.

Lattice regression takes into account the LUT interpolation, and directly chooses the $m \times 1$ vector of output values $b$ such that the interpolated training CIELAB colors will be close to their corresponding RGB output values. A training point $x_i$ in CIELAB falls in a cell of the lattice with eight vertices; the $j$th vertex in the lattice is given a linear interpolation weight $w_{ij} \geq 0$, where $w_{ij} = 0$ if $a_j$ is not a vertex of the cell containing $x_i$, and otherwise $w_{ij}$ is set so that linear interpolation equations hold: $\sum_j w_{ij} a_j = x_i$, and $\sum_j w_{ij} = 1$ (see [6] for more on calculating linear interpolation weights). The LUT then interpolates the input CIELAB value $x_i$ as $\hat{y}_i = \sum_j w_{ij} b_j$.

Lattice regression minimizes the post-interpolation error on the training data. That is, it chooses the $m \times 1$ vector of output

values $\hat{b}$ that solve,

$$\hat{b} = \underset{b\in[0,255]^m}{\arg\min} \sum_{i=1}^{n} (\hat{y}_i - y_i)^2 .$$
$$= \underset{b\in[0,255]^m}{\arg\min} \sum_{i=1}^{n} \left( \left( \sum_j w_{ij} b_j \right) - y_i \right)^2 . \quad (2)$$

To write (2) more compactly, let $W \in [0,1]^{n\times m}$ denote the matrix with $i$th-$j$th element $w_{ij}$. Then given $n$ training inputs $\{x_i\}$ and their corresponding linear interpolation weights $W$, the LUT interpolates a vector of $n$ output values $\hat{y} = Wb$. If we let $y$ be the $n \times 1$ vector of corresponding output values, then the lattice regression objective (2) can be succinctly expressed,

$$\hat{b} = \underset{b\in[0,255]^m}{\arg\min} \|\hat{y} - y\|_2^2.$$
$$= \underset{b\in[0,255]^m}{\arg\min} \|Wb - y\|_2^2. \quad (3)$$

Conveniently, (3) has a closed-form solution:

$$\hat{b} = (W^T W)^{-1} W^T y, \quad (4)$$

and because $W$ is sparse, the matrix inversion can be computed efficiently by sparse Cholesky factorization (for instance the `ldivide` command in Matlab).

In related work by Bala [3], a LUT is learned and then adjusted to correct for the interpolation error using an iterative technique. A similar iterative method was proposed by Tobler for a similar regression problem in geospatial analysis [16]. More recently, Monga and Bala [13] proposed a joint optimization of lattice node location as well as node output values and the optimization for node output values is as given in (4). However, our contribution is the proposal and analysis of regularization terms that can be added to solve (3) when it is underdetermined.

### Incorporate Additional Constraints

A key advantage of this approach is that because the lattice output values are being directly estimated, it is straightforward to add constraints to the LUT that are appropriate for color management. For printers it is common to want saturated bright input color values to print as pure ink colors, for example, one may desire that the CIELAB input $[100,0,0]$ be mapped to the pure white RGB printer input $[255,255,255]$, and that the CIELAB input $[80,0,100]$ be mapped to the pure yellow RGB printer input $[255,255,0]$. This kind of constraint is easily incorporated into the lattice regression objective by treating these as training pairs and setting a large relative weight for these particular points within the outer summation of (2). Another type of constraint is to require that a subset of the nodes share the same output value, for instance one could force all inputs with $L^* < 10$ to produce the same RGB value, in order to reduce color noise in dark areas of images. For artistic purposes, one can also constrain the allowed set of output RGB values, for example, to create sepia-toned images one can allow the learned output values to be restricted to the sepia-ramp for that printer.

### Regularization Needed

The minimization of (2) is not, in general, a well-posed problem. For a LUT node belonging to a cell that do not contain training samples, any output value will result in the same objective function cost. Mathematically, this problem is *underdetermined*. Prior knowledge about the nature of the underlying color transformation should be added to (3) in order to set the value for such nodes. Specifically, one expects the color transformation to be somewhat smooth, and we encode this information as a regularization term that prefers to give close LUT nodes output values that are also close (in some sense). Adding such a regularization term will also reduce the probability of overfitting any noise in the training data measurements, and will lead to an overall smoother transformation.

First, we review our previous regularization approach to enforce smoothness which required two regularization terms [6, 7], then in the following subsection we propose using the thin-plate regularizer instead and explain why it is a more elegant solution.

### Laplacian and Global Bias Regularizers

The lattice formed by the LUT can be represented as a graph with edges connecting adjacent nodes in the lattice. A standard approach to enforcing smoothness on the nodes of a graph is to minimize the graph Laplacian [12], which can be expressed as the sum of squared differences between the values at adjacent nodes in a graph, that is:

$$J_L(b) = \sum_{\text{adjacent } a_i, a_j} (b_i - b_j)^2$$
$$= b^T L b, \quad (5)$$

where the graph Laplacian $L$ is the diagonal degree matrix of the graph minus the adjacency matrix of the graph.

Solving for a lattice that minimizes the empirical risk (2) and also minimizes the Laplacian given by (5) forces the values chosen for adjacent nodes in the lattice to be close, and is expressed as:

$$\hat{b} = \underset{b\in[0,255]^m}{\arg\min} \|Wb - y\|_2^2 + \lambda J_L(b), \quad (6)$$

where the regularization parameter $\lambda > 0$ trades-off the two goals. Just as with (4), the solution to (6) has a closed form that can be efficiently computed via sparse Cholesky factorization.

The top plot of Fig. 1 shows an example of training samples (blue dots) and the function estimated by interpolating a lattice learned by (6). In this example, the regularization parameter was (optimistically) chosen to minimize the test error on a set of test samples (not shown) that were drawn independent from the training samples. For LUT nodes that comprise cells that do not have any training data, the Laplacian regularizer pushes their output values to be the mean of nearby training points. This effect is seen as the flat parts at the edge of the interpolated function shown in Fig. 1. For learning color transformations, we found that this could be problematic near the edge of the gamut. Also, the Laplacian regularizer pushes all the lattice output values towards the mean value of the training samples, which can lead to a slight decrease in contrast.

To lessen these problems, we proposed adding a global bias term that would help the lattice regression extrapolate [6, 7]. Specifically, the global bias term regularizes the lattice regression estimate of the lattice outputs $b$ towards some other estimate of the color transformation $\hat{f}$. For example, solve (1) for some function class $\mathscr{F}$ to produce an estimate $\hat{f}$. To be a useful global

bias term, $\mathscr{F}$ needs to be an extrapolating function class, but it can be a rather poor estimate, for example, one might fit the best linear function to the training data. With this global bias, and letting $\hat{f}(a)$ be the $n \times 1$ vector with $i$th element $\hat{f}(a_i)$, the lattice regression problem is expressed:

$$\hat{b} = \underset{b\in[0,255]^n}{\arg\min} \|Wb - y\|_2^2 + \lambda J_L(b) + \lambda_g \|\hat{f}(a) - b\|_2^2, \qquad (7)$$

where $\lambda_g > 0$ is the regularization parameter on the global bias term. With the addition of these two regularizers, the optimization given in (7) still has a closed-form solution; for details see [6]. In our previous work and in the comparisons in this paper, we used a trilinear function for the global bias; again, for details see [6].

If only a little weight $\lambda_g$ is put on the global bias term, then it will have a weak effect on most vertices, but it will help the function extrapolate for vertices that are far from the training data. If a large weight $\lambda_g$ is put on the global bias term, then the fitted lattice will be close to the $\hat{f}$, and the lattice regression acts more like a correction to $\hat{f}$, observing that the applied lattice will be interpolated.

The performance of (7) is impressive [6], but the dual-regularization is awkward in that both regularization parameters must be chosen, and a separate function $\hat{f}$ must be estimated.

## Using the Thin-plate Regularizer

The main contribution of this paper is in showing that the thin-plate regularizer can be used with lattice regression as a replacement for the dual-action of the previously proposed regularization terms. In addition to removing one parameter from the model, the new regularization approach employs a more rigorous mathematical notion of smoothness as well.

Before introducing the new regularizer, it is helpful to further analyze the net effect of the two previously proposed regularizers. First, the Laplacian term (5) penalizes the squared difference of *neighboring* vertices only, thus imposing a *first-order* sense of smoothness. This first-order concept of smoothness considers only constant functions to be perfectly smooth (that is, $J_L(b) = 0$ only when $b_i = b_j$ for all $i, j$) and any linear trend in the fitted model is penalized. The global bias term corrects for this often-undesirable penalty by encouraging the fitted function to also not stray far from the global trilinear trend of the data. Combined, the two terms approximate a *second-order* sense of smoothness in an ad-hoc way by first penalizing all variation (including linear variation) and then decreasing the penalty on those linear variations that align to the global trend of the data. The proposed thin-plate regularization, however, encodes a second-order sense of smoothness directly with a single term.

The thin-plate regularizer [8], adapted from the literature on multi-dimensional splines, provides a convenient and mathematically-precise notion of second-order smoothness that fits into the existing structure of lattice regression. While the conceptual notion of the thin-plate regularizer is simple to grasp, the form used in computation is difficult to arrive at. Therefore, we will begin by focusing on the conceptual definition and provide the implementation details primarily for reference.

In physical terms, the thin-plate regularizer $J_K(f)$ measures the amount of energy required to bend a (differentially) thin plate into the form of the function $f$, where $f$ is the result of interpolation from the lattice. Importantly, linear trends of $f$ go unpenalized by this energy. The second-order nature of this regularizer is more directly seen in the integral definition, which shows that one is regularizing the total second-order derivative of $f$ in all directions:

$$J_K(f) = \int_{L^*}\int_{a^*}\int_{b^*} \Big(\frac{\partial^2}{\partial L^2}f^2 + 2\frac{\partial^2}{\partial L \partial a}f^2 + 2\frac{\partial^2}{\partial L \partial b}f^2 + \\ + \frac{\partial^2}{\partial a^2}f^2 + 2\frac{\partial^2}{\partial a \partial b}f^2 + \frac{\partial^2}{\partial b^2}f^2\Big) db^* da^* dL^*. \qquad (8)$$

Next, we will show that for interpolation that is linear in the lattice outputs, the thin-plate regularizer can be concisely written as

$$J_K(b) = b^T K b \qquad (9)$$

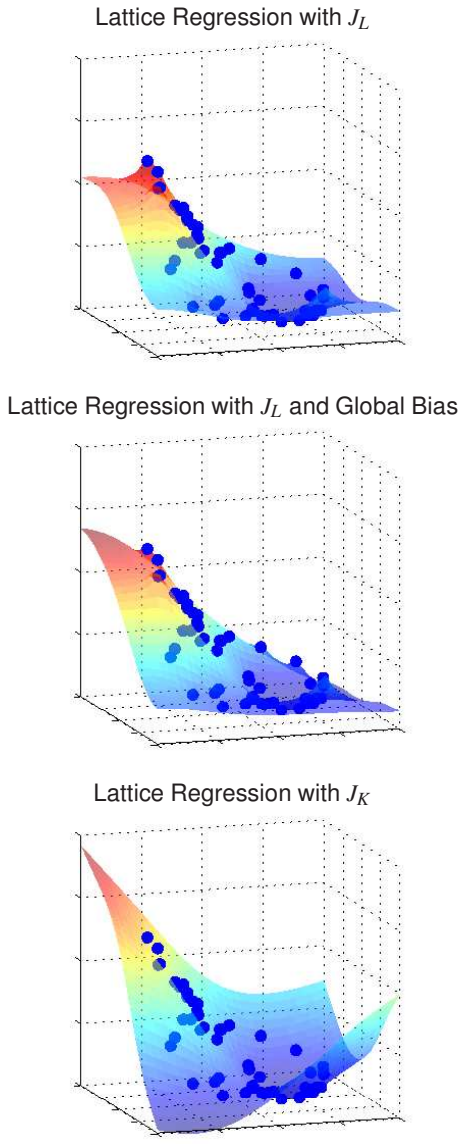


***Figure 1.*** *Figure illustrates the extrapolation effect of the different regularizers when combined with the lattice regression empirical risk minimization. The training data (blue dots) used to fit each of the models was the same in all cases, and was drawn from a Gaussian Mixture Model.*

for a particular $m \times m$ matrix $K$ that depends only on the type of interpolation basis function and the lattice dimensions. Importantly, $K$ it needs to be computed only once for a given lattice structure (for example, once for all $17 \times 17 \times 17$ regular rectangular lattices), as it does not depend on any data values.

Combining (9) with (3), we arrive at the proposed objective:

$$\hat{b} = \underset{b \in [0,255]^m}{\arg\min} \; \|Wb - y\|_2^2 + \lambda J_K(b). \tag{10}$$

Again, this can be solved in closed form and the sparsity of the involved matrices lends an efficient solution.

In Fig. 1 , the effect of the thin-plate regularizer is compared to the previous Laplacian and global-bias regularizers for a simulated dataset drawn from a Gaussian mixture model. We see that the Laplacian regularization $J_L$ alone (top plot) has the effect of forcing the function values toward the mean when extrapolating from the training samples. This is corrected for somewhat by the addition of the bias term (middle plot), but using $J_K$ (bottom plot), one sees entirely different behavior with more linear extrapolations.

### *Computing $K$ for the Thin-Plate Regularizer*

This subsection details how to compute the matrix $K$ in (9). First, note that both linear interpolation and cubic interpolation of a point $x \in$ CIELAB given the lattice $\{a_j, b_j\}_{j=1}^m$ can be expressed as

$$f(x; \{a_j, b_j\}_{j=1}^m) = \sum_{j=1}^m k(x, a_j) b_j,$$

where $k : \text{CIELAB} \times \text{CIELAB} \to \mathbb{R}$ is known as the basis function associated with the function class of $f$ (the $k$ for linear interpolation and cubic interpolation differ). We would like to directly apply the thin-plate penalty to linear interpolation, but, due to discontinuities, this does not produce a twice-differentiable function and thus the thin-plate regularizer (8) can not be computed for it. Instead, we compute the thin-plate regularizer for the cubic interpolation kernel. The resulting $J_K(b)$ regularizer will prefer lattice outputs $b$ that are smooth in the thin-plate sense (8) if cubic-interpolated, but even under linear interpolation we expect the effective function to have a similar shape.

For $x \in$ CIELAB, the kernel for tricubic interpolation can be written as

$$k(x, a_j) = \prod_{\nu \in \{L^*, a^*, b^*\}} \delta_{\nu j}\Big((x)_\nu\Big)$$

where $(\cdot)_\nu$ extracts the $\nu \in \{L^*, a^*, b^*\}$ coordinate of a vector and with

$$\delta_{\nu j}(u) = \tilde{\delta}\left(\frac{\big|u - (a_j)_\nu\big|}{h_\nu}\right),$$

for $u \in \mathbb{R}$ where $h_\nu \in \mathbb{R}$ is the spacing of the lattice in the $\nu$ coordinate. This expression can be interpreted as centering and scaling the following one-dimensional cubic interpolation kernel about the $j$-th lattice node for $u \in \mathbb{R}$:

$$\tilde{\delta}(u) = \begin{cases} 1.5u^3 - 2.5u^2 + 1 & \text{if } 0 \le u < 1 \\ -.5u^3 + 2.5u^2 - 4u + 2 & \text{if } 1 \le u < 2 \\ 0 & \text{otherwise.} \end{cases}$$

The kernel matrix $K$ has the (i,j)-entry

$$K_{i,j} = \sum_{s=0}^{2} \sum_{t=0}^{2-s} \omega(s,t)\, D_{L_{i,j}}^{(s)} D_{a_{i,j}}^{(t)} D_{b_{i,j}}^{(2-s-t)},$$

where

$$\omega(s,t) = \begin{cases} 2 & \text{if } s = 1 \text{ or } t = 1 \\ 1 & \text{otherwise} \end{cases}$$

weights the cross-components with the value 2 and where

$$D_{\nu_{i,j}}^{(s)} = \int \delta_{\nu i}^{(s)}(\nu') \delta_{\nu j}^{(s)}(\nu')\, d\nu' \tag{11}$$

with $f^{(s)}$ denoting the $s$-th derivative of the function $f$.

Because $\delta$ is piecewise-polynomial, (11) can be computed algebraically. Furthermore, the computation of $K$ is decoupled from the lattice values $b$, and can thus be pre-computed once for a particular lattice size and reused indefinitely.

## Experiments

Accuracy and smoothness experiments were performed to compare lattice regression with $J_K$ regularization to lattice regression with $J_L$ + global-bias regularization, and to Tikhonov-regularized local linear regression [9, 10].

### *Experimental Details*

Each of the three methods has a regularization parameter that must be trained to determine how much the regularizer is weighted compared to the empirical error. Choices for the cross-validation parameters were based on previously published papers and preliminary experiments, and are detailed in Table 1.

**Table 1. Cross-validation Parameter Choices**

| | $\lambda$ | $\lambda_g$ |
|---|---|---|
| Lattice $J_K$ | {1e-8,1e-7,...,1e0} | n/a |
| Lattice $J_L$+bias | {1e-6,1e-5,...,1e2} | × {1e-8,1e-7,...,1e0} |
| Tikhonov | {1e-4,1e-3,...,1e4} | n/a |

All lattices were $17 \times 17 \times 17$ ranging from $L \in [0, 100]$, $a \in [-100, 100]$, and $b \in [-100, 100]$. Training data was produced for each printer by printing the Gretag MacBeth TC9.18 RGB target, which consists of 918 color patches, of which 789 form a $9 \times 9 \times 9$ grid in the RGB space, and the remaining 189 samples are neutral ramps and bright saturated colors. All prints were measured with an X-Rite iSis spectrophotometer, using D50 illuminant at a $2^o$ observer angle and UV filter. Preliminary experiments of the stability of the spectrophotometer showed that the variance of repeated measurements averaged over 918 color patches was less than .1 $\Delta E_{2000}$.

All the 3D LUTS were preceded by the 1D gray-calibration LUTS for each color channel, as described by Bala [2]. The 1D LUTS were the same for all experiments for each printer, and were estimated using Tikhonov-regularized regression method [9, 10] with a regularization parameter $\lambda = 1$.

For each printer, a randomly generated set of RGB values was printed and measured, forming an independent set of RGB $\to$ CIELAB pairs. This set provided a dual purpose in our experiments. The first role is as a test set; since the CIELAB values are

guaranteed to be within the gamut of each printer, we use the set to assess the color accuracy of the constructed LUTs. That is, the LUT is applied to the CIELAB values, producing $\widehat{\text{RGB}}$ which is sent to the printer and measured, producing $\widehat{\text{CIELAB}}$ which can be compared in $\Delta E_{2000}$ to the original CIELAB values. Second, these set of values provided an independent set on which the RGB error of the LUT built by an algorithm could be compared across different parameter settings. Since cross-validation of the parameters for each algorithm in terms of the actual CIELAB error is intractable in a realistic workflow, the absolute RGB error was used as a proxy in order to find an appropriate parameter setting for the specific device. Table 1 shows the compared parameter settings and those that were chosen as optimal in terms of absolute RGB error are shown in Table 2.

**Table 2. RGB-error Cross-validated Parameters**

| | | $\lambda$ | $\lambda_g$ |
|---|---|---|---|
| Brother | Lattice ($J_K$) | 1e-5 | n/a |
| | Lattice ($J_L$+bias) | 1e0 | 1e-2 |
| | Tikhonov | 1e2 | n/a |
| HP | Lattice ($J_K$) | 1e-5 | n/a |
| | Lattice ($J_L$+bias) | 1e0 | 1e-2 |
| | Tikhonov | 1e1 | n/a |

### *Subjective Smoothness Experiments*

Starting with the parameters in Table 2, a range centered at these values was logarithmically swept by two values in each direction in order to compare how the smoothness and accuracy change with respect to the parameters. As described above, accuracy was measured in the $\Delta E_{2000}$ error between the desired and measured CIELAB values. Additionally, the smoothness of each algorithm was assessed by perceptual evaluation of a target consisting of circular ramps on a neutral $(L^* = 50, a^* = 0, b^* = 0)$ background. Examples of the smoothness target are shown in Fig. 2, which shows that discontinuities in the color show up as circular artifacts. We found in previous experiments that such circular artifacts are generally more noticeable than the banded artifacts produced by analogous rectangular color ramps. All of the CIELAB ramps were adjusted to lie within the gamut of the intended printer and this gamut was estimated by the alpha-shape [4] (with $\alpha = 2000$) applied to the measured values from the TC9.18 chart for that printer. For seven of the ramps, a constant chroma was fixed at values lying at a constant radius in $a^*b^*$ (including one at $a^* = 0, b^* = 0$) and the $L^*$ value was swept within the range of the gamut at this chroma. The remaining two ramps were the lines (in $[L^*, a^*, b^*]$) {[100,-100,-100],...,[0,100,100]} and {[100,-100,100],...,[0,100,-100]} clipped to the gamut and these provide an example of smoothness in both lightness and hue.

Twenty adults with normal color vision subjectively ranked the smoothness of the printed targets. Each person ranked fifteen targets separately for the inkjet and laser printer, with ties not allowed. The lighting was produced by two 4700K Sollux lamps from Tailored Lighting approximating D50 illumination.

The ranked data was treated as pairwise preference information, and Thurstone's Law (Case V) was applied for analysis [5], producing a smoothness scale value for each print along with a increment $\Delta_s$ by which values can be judged significantly different.

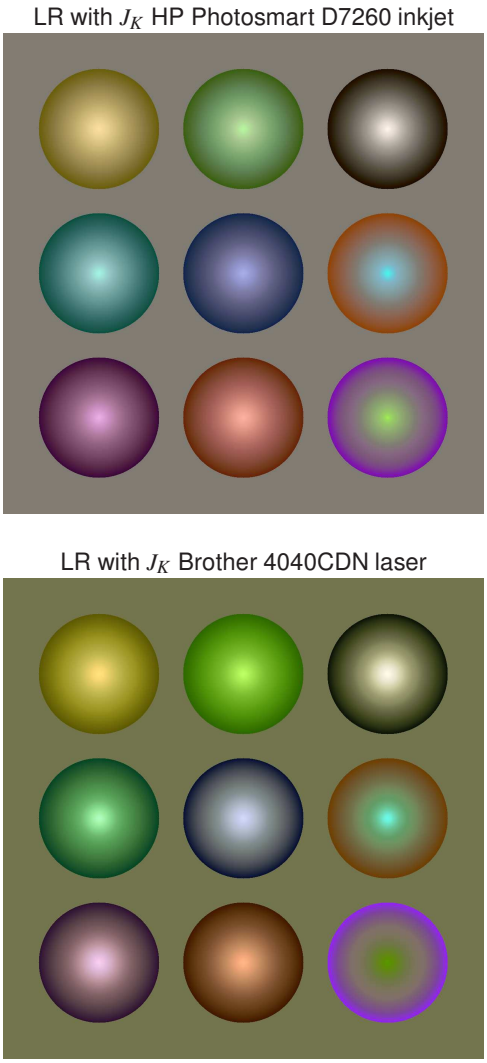


***Figure 2.*** *Figure shows examples of the color-managed prints that subjects ranked to evaluate smoothness. Since the displayed RGB values are intended to be printed by the respective printers, the actual smoothness and color is not reproduced accurately but, when treated as SRGB, the images do effectively illustrate the kind of contours seen in the prints.*

## Results

In Fig. 3, the performance of each algorithm was evaluated in both smoothness and median $\Delta E_{2000}$ accuracy over a range of parameter settings. Here we see similar performance between the two regularization techniques for lattice regression with no clear winner between the two. However, we see that lattice regression outperforms local-linear Tikhonov regression, as the results for the latter are to the lower-right. Additionally, a comparison of the median and 95$^{\text{th}}$-percentile $\Delta E_{2000}$ errors and smoothness in terms of $\Delta_s$, the smallest significant smoothness difference, is shown in Table 3 for the cross-validated parameter settings. Here, we see comparable errors but increased smoothness for lattice regression (vs. Tikhovnov) for the Brother 4040CDN laser printer. For the HP Photosmart D7260 inkjet printer, the errors are again comparable, but lattice regression with the $J_K$ regularizer does not produce a significantly different smoothness than Tikhonov.

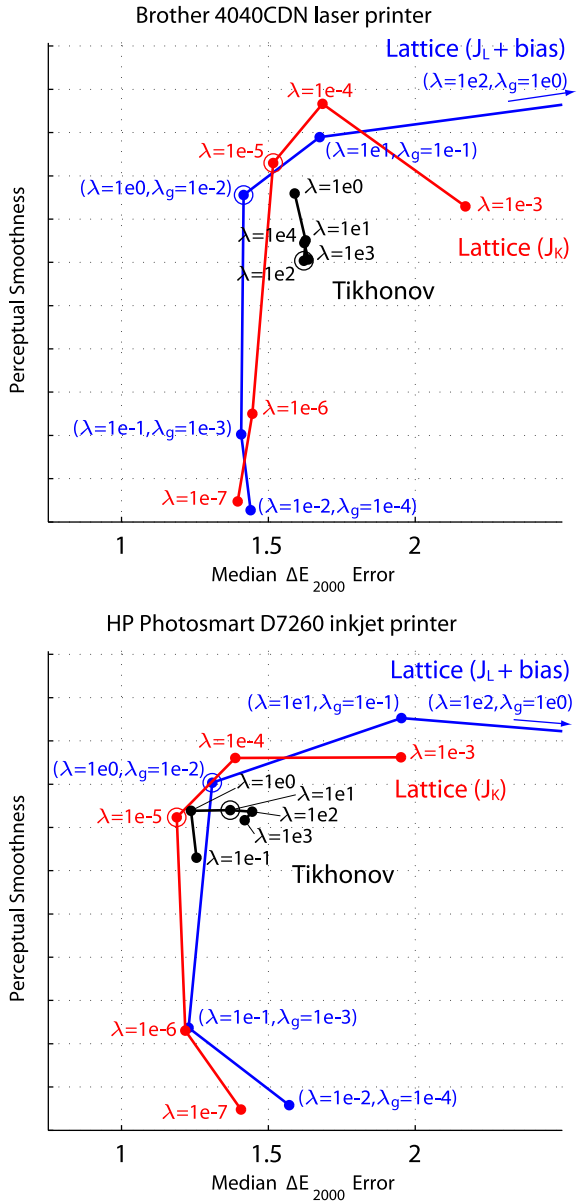


***Figure 3.*** *Figure shows the psychometrically scaled smoothness obtained from the ranking experiment plotted against median $\Delta E_{2000}$ error for varying algorithm parameters. The "best" algorithms will have values in the upper-left (high smoothness and low error). The data point indicated by the circle is the cross-validated algorithm setting. Legend: Red=LR ($J_K$), Blue=LR ($J_L$+bias), Black=Tikhonov.*

## Conclusions

In this paper we show that using only the thin-plate regularizer can achieve the same performance as the combination of two regularizers previously proposed for lattice regression. Further, we demonstrated through controlled objective and subjective experiments that the regularized lattice regression achieves state-of-the-art performance for color management in terms of accuracy and smoothness.

**Table 3. Cross-validated Algorithm Comparison**

| | | $\Delta E^*_{2000}$ Error | | Smooth-ness* |
|---|---|---|---|---|
| | | Median | 95%-ile | |
| Brother | Lattice ($J_L$+bias) | 1.42 | 3.48 | $4.03\,\Delta_s$ |
| | Lattice ($J_K$) | 1.52 | 3.92 | $6.00\,\Delta_s$ |
| | Tikhonov | 1.62 | 3.37 | $0.00\,\Delta_s$ |
| HP | Lattice ($J_L$+bias) | 1.31 | 2.53 | $2.12\,\Delta_s$ |
| | Lattice ($J_K$) | 1.19 | 2.21 | $0.00\,\Delta_s$ |
| | Tikhonov | 1.37 | 2.90 | $0.44\,\Delta_s$ |

***Note the smoothness scores should not be compared across printers, as the scores are based on relative assessments made for each printer separately.**

## References


[1] A. Artusi and A. Wilkie, Novel color printer color characterization model, Journal of Electronic Imaging, vol. 12, no. 3 (2003).

[2] R. Bala, Digital Color Handbook, chapter 5: Device Characterization, CRC Press, pp. 269–384 (2003).

[3] R. Bala, Iterative technique for refining color correction look-up tables, U.S. Patent 5649072 (1997).

[4] T. J. Cholewo and S. Love, Gamut boundary determination using alpha-shapes, Proc. 7th IS&T Color Imaging Conference (1999).

[5] P. Engeldrum, Psychometric Scaling, chapter 8: Indirect Interval Scaling - Case V and Paired Comparison, Imcotek Press, pp. 93–108 (2000).

[6] E. K. Garcia and M. R. Gupta, Building Accurate and Smooth ICC Profiles By Lattice Regression, Proc. 17th IS&T Color Imaging Conference pp. 101–106 (2009).

[7] E. K. Garcia and M. R. Gupta, Lattice Regression, Advances in Neural Information Processing Systems (NIPS) pp. 1–9 (2009).

[8] P.J. Green, B.W. Silverman, Nonparametric Regression and Generalized Linear Models: A roughness penalty approach, (1994).

[9] M. R. Gupta, E. K. Garcia, and E. M. Chin, Adaptive local linear regression with application to printer color management, IEEE Trans. on Image Processing, vol. 17, no. 6, pp. 936–945 (2008).

[10] N. Hrustemovic and M. R. Gupta, Multiresolutional regularization of local linear regression over adaptive neighborhoods for color management, Proc. of the Intl. Conf. on Image Processing (2008).

[11] H. R. Kang and P. G. Anderson, Neural network application to the color scanner and printer calibrations, Journal of Electronic Imaging, vol. 1, pp. 25–135 (1992).

[12] U. Luxburg, A Tutorial on Spectral Clustering, Statistics and Computing, pp. 395 - 416 (2007).

[13] V. Monga and R. Bala, Algorithms for Color Look-up-table (LUT) Design via Joint Optimization of Node Locations and Output Values, ICASSP (2010).

[14] J. Morovic, A. Albarran, J. Arnabat, Y. Richard, and M. Maria, Accuracy-preserving smoothing of color transformation LUTs, Proc. 16th IS&T Color Imaging Conference pp. 243–246 (2008).

[15] T. Olson, Smooth Ramps: Walking the Straight and Narrow Path through Color Space, Proc. 7th IS&T Color Imaging Conference, pp. 57–64 (1999).

[16] W. R. Tobler, Lattice Tuning, Geographical Analysis, vol. 11, no. 1, pp. 36–44 (1979).