Optimized Regression for Efficient Function Evaluation

Eric Garcia, Raman Arora, and Maya R. Gupta, *Senior Member, IEEE*

***Abstract*— In many applications of regression, one is concerned with the efficiency of the estimated function in addition to the accuracy of the regression. For efficiency, it is common to represent the estimated function as a rectangular lattice of values—a lookup table (LUT)—that can be linearly interpolated for any needed value. Typically, a LUT is constructed from data with a two-step process that first fits a function to the data, then evaluates that fitted function at the nodes of the lattice. We present an approach, termed lattice regression, that directly optimizes the values of the lattice nodes to minimize the post-interpolation training error. Additionally, we propose a second-order difference regularizer to promote smoothness. We demonstrate the effectiveness of this approach on two image processing tasks that require both accurate regression and efficient function evaluations: inverse device characterization for color management and omnidirectional super-resolution for visual homing.**

***Index Terms*— Color, image fusion, image registration.**

I. Introduction

THE ACCURACY of a regression technique is typically of primary importance. In high-throughput applications, the computation required for evaluating the fitted function becomes important as well. Gamut mapping of video, color transformations of printed images, and perspective renderings of panoramic images are all applications where a function must be estimated accurately and quickly evaluated for millions of pixels. However, most regression methods do not produce estimated functions that are optimized for fast implementation. For example, kernel-based methods, such as Gaussian process regression and support vector regression require kernel computations between each test sample and a (possibly large) subset of training examples, whereas local smoothing techniques, such as weighted nearest neighbors or local linear regression require a search for the nearest neighbors.

In this paper, we focus on problems in which one is given a training set of n sample input–output pairs, and asked to produce a function that is both faithful to these samples and efficient to evaluate. Consider the common *two-step* solution [1]: 1) estimate an *intermediate* function from the given n sample pairs using a regression method that is as accurate as possible and 2) for efficiency, evaluate the intermediate function at the nodes of a regular lattice and store those values. Then, given a new input x that lies within the domain of the lattice, the estimated function $f(x)$ is evaluated by interpolating x from its surrounding lattice nodes. For a given x, solving for its interpolation weights is trivial on a well-designed rectangular lattice using bit operations [2]. Also, interpolating the lattice is independent of the size of the original training set: each function evaluation requires linearly interpolating the 2^d surrounding lattice nodes for a d-dimensional function. However, this exponential scaling with respect to dimension makes the lattice-interpolation approach best-suited for low-dimensional applications. This approach is used ubiquitously in color management (typically 3–4-D) where real-time performance often requires millions of evaluations every second, it is standardized by the international color consortium (ICC) with a file format called an ICC profile [3], and the lattice is referred to as a lookup table (LUT).

However, the above approach for estimating the lattice outputs is suboptimal in terms of accuracy because the effect of interpolation is not considered when estimating the intermediate function. Most regression methods use the training data to choose parameters that minimize the training set error (empirical risk), but in this situation, the empirical risk does not take into account the interpolation step. In this paper, we investigate *lattice regression*, a solution that produces a lattice that minimizes the interpolation error on the training samples while enforcing regularity of the estimated function. The key to this estimation is that any interpolation operation that reduces to a linear combination of lattice node outputs can be inverted to solve for the node outputs that minimize the squared error of the training data.

We begin in Section II with an explanation of lattice regression, including a discussion of different regularization strategies to ensure a unique and smooth regression. Then in Section III, we relate lattice regression to other regression approaches, detailing how it belongs to the family of spline methods as well as structural risk minimization methods. We illustrate the value of lattice regression with two image processing applications where fast-to-evaluate regression is essential: color management of printers in Section IV, and super-resolution of omnidirectional images in Section V. It is already standard for efficiency to interpolate a lattice for color management, and we show that lattice

Manuscript received July 7, 2011; revised February 10, 2012; accepted May 1, 2012. Date of publication May 22, 2012; date of current version August 22, 2012. This work was supported in part by a United States PECASE Award and the United States Office of Naval Research. The associate editor coordinating the review of this manuscript and approving it for publication was Dr. Ramin Samadani.

The authors are with the Department of Electrical Engineering, University of Washington, Seattle, WA 98195 USA (e-mail: eric.garcia@gmail.com; rmnarora@u.washington.edu; gupta@ee.washington.edu).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TIP.2012.2200902

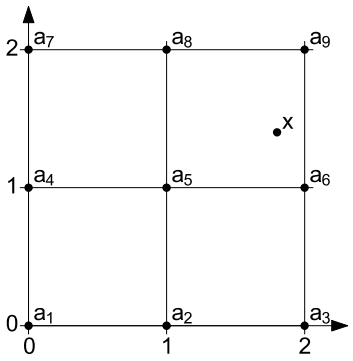


Fig. 1. 3×3 lattice with test point $x = [1.8, 1.4]^T$.

regression yields statistically significant higher accuracy than the state of the art due to its optimization of the interpolation step. For omnidirectional super-resolution from unregistered images, we show that the fast evaluation time of lattice regression (for a lattice on the sphere) makes it more feasible to search a large space of possible solutions, and that results in super-resolved images that are both more accurate and more useful for visual homing than the state of the art.

Preliminary versions of this paper have been published as conference papers and in a thesis [4]–[7]. This paper differs from those preliminary publications in that we present a simpler regularizer, a deeper discussion of related work, all-new and more exhaustive experiments, and the new application to super-resolution.

II. Lattice Regression

In this paper, we use the term *lattice* to refer to any set of m nodes $\{a_j \in \mathbb{R}^d\}_{j=1:m}$ that can be linearly transformed to a d-dimensional rectangular grid. It is not necessary to have a regular spacing between the nodes of this grid, but for ease of exposition, we will restrict our attention to such regular lattices. Further, without loss of generality, we will assume that the domain has undergone an affine transformation placing one corner of the lattice at the origin with unit spacing between nodes, producing a lattice as illustrated in Fig. 1. For every lattice node a_j, let $b_j \in \mathbb{R}$ be the corresponding lattice output.

Given a set of n inputs $\{x_i \in D\}_{i=1:n}$ where $D \subset \mathbb{R}^d$, and corresponding n outputs $\{y_i \in \mathbb{R}\}_{i=1:n}$, a common approach to regression is to estimate a function $\hat{f} : D \to \mathbb{R}$ within a "well-behaved" class of functions $\mathcal{F}$ such that $\sum_{i=1}^n \ell(\hat{f}(x_i), y_i)$ is minimized for some loss function $\ell : \mathbb{R} \times \mathbb{R} \to \mathbb{R}_+$. In lattice regression, we restrict the class of functions $\mathcal{F}$ to be the set of functions that can be implemented by linearly interpolating a rectangular lattice. For a fixed set of lattice nodes $\{a_j\}_{j=1:m}$ and a given linear interpolation method, any function in this class is fully defined by a set of lattice outputs $\{b_j\}_{j=1:m}$, and the task of lattice regression is to solve for the $\{b_j\}_{j=1:m}$ that minimize the *empirical risk* (i.e., interpolation error for the training samples) $\sum_{i=1}^n \ell(\hat{f}(x_i), y_i)$ subject to regularization on the smoothness of $\hat{f}$. In the following sections, we describe this empirical risk and introduce a "smoothness" functional that can be used as effective regularization.

A. Lattice Regression Empirical Risk

The lattice regression *empirical risk* term solves for lattice outputs $\{b_j\}_{j=1:m}$ that accurately interpolate the training data, using a pre-specified form of *linear interpolation*. To linearly interpolate the point x_i, a set of linear interpolation weights $\{w_{ij}\}_{j=1:m}$ are computed that satisfy the following equations:

$$\sum_{j=1}^{m} w_{ij} a_j = x_i, \quad \sum_{j=1}^{m} w_{ij} = 1. \tag{1}$$

Note that the equations in (1) are, in general, underdetermined and there are many strategies for choosing a unique solution, each corresponds to a particular form of linear interpolation. For example, in three dimensions, possible solutions include trilinear, pyramidal, and tetrahedral interpolation [8]. For all of these popular linear interpolation techniques, $w_{ij} \neq 0$ only for those a_j that are vertices of the cell containing x_i. In our experiments, we use d-linear interpolation (i.e., bilinear, trilinear, etc.), details for computing these linear interpolation weights are given in the Appendix. The d-linear interpolation variant is arguably the most popular variant of linear interpolation, can be implemented efficiently, and is the maximum entropy solution to (1) [9, Th. 10].

Given the linear interpolation weights $\{w_{ij}\}$, the output corresponding to x_i is calculated as $\hat{y}_i = \sum_j w_{ij} b_j$. Thus, the $m \times 1$ vector of output lattice values $\hat{b}$ that minimizes the post-interpolation error on the training data is given by

$$\begin{aligned} \hat{b} &= \arg\min_{b \in \mathbb{R}^m} \sum_{i=1}^{n} \left(\hat{y}_i - y_i\right)^2 \\ &= \arg\min_{b \in \mathbb{R}^m} \sum_{i=1}^{n} \left(\left(\sum_{j=1}^{m} w_{ij} b_j \right) - y_i \right)^2. \end{aligned} \tag{2}$$

Let $y = \left[y_1, \ldots, y_n\right]^T$, and let $\mathbf{W} \in [0,1]^{n \times m}$ denote the matrix with ith-jth element w_{ij}, where $\mathbf{W}$ is determined uniquely by the inputs $\{x_i\}_{i=1:n}$ and the lattice nodes $\{a_j\}_{i=1:m}$. Interpolating the lattice for the training inputs $\{x_i\}_{i=1:n}$ yields the estimated output values $\hat{y} = \mathbf{W}b$. Thus, the empirical risk objective (2) can be written as

$$\hat{b} = \arg\min_{b \in \mathbb{R}^m} \|\hat{y} - y\|_2^2 = \arg\min_{b \in \mathbb{R}^m} \|\mathbf{W}b - y\|_2^2. \tag{3}$$

The solution to (3) is $(\mathbf{W}^T\mathbf{W})^{-1}\mathbf{W}^T y$, unless (3) is underdetermined and, unfortunately, this is often the case. For example, (3) is underdetermined if there are no training samples in any of the lattice cells bordering some lattice node a_j because there will be no constraints on the corresponding b_j. In this case, $\mathbf{W}^T\mathbf{W}$ will not be invertible, and (3) will have an infinite set of minimizers. To avoid this problem and to reduce noise and over-fitting, we add a regularizer, as discussed in the next section.

In related work, researchers in geospatial analysis [10] and in digital color management [11] have proposed solutions to learning lattices that iteratively post-process an initial lattice solution to reduce the interpolation error. For efficiency, we use a fixed regular grid of knots (a lattice). However, one could instead optimize the node output values of an irregular grid,

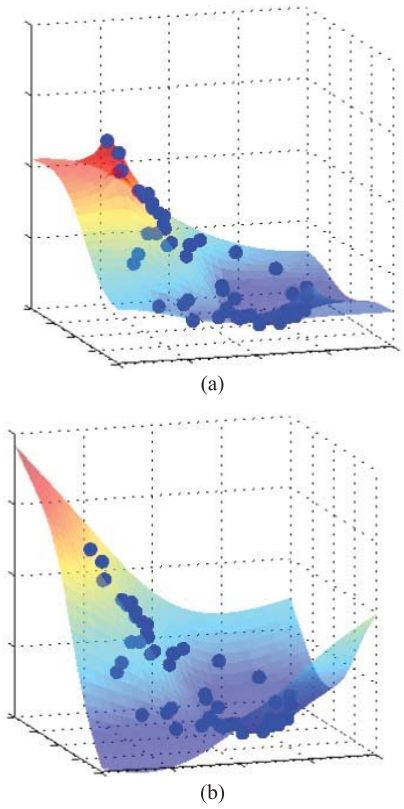


Fig. 2. Examples of the functions estimated by lattice regression (after interpolation of the lattice nodes). The blue dots are the training points, and are the same in both figures. (a) Graph Laplacian regularizer. (b) Graph Hessian regularizer. The graph Hessian does not penalize linear extrapolations, while the graph Laplacian does.

though this requires introducing an appropriate scaling factor for each cell of the lattice to normalize for differing cell volumes. A number of researchers have tackled the related problem of choosing good grid knot locations for use with linear interpolation [12]–[17].

B. Lattice Regression Regularizer

We add a regularizer to (3) to ensure a unique solution and to promote *smoothness*, which reduces over-fitting to the possibly noisy training data. But how exactly should one measure the "smoothness" of a function that is to be interpolated from a lattice? The answer to this question is likely to be highly application-dependent and thus the best regularizer may also be application-dependent. One generic choice for a regularizer on a graph such as a lattice, is the *graph Laplacian* regularizer [18], [19], a first-order measure of smoothness in the sense that it penalizes squared-differences in output values for adjacent lattice nodes. However, our preferred generic choice is a second-order difference regularizer we call a *graph Hessian*. We compare and contrast these choices below, and illustrate their differences in Fig. 2.

1) Graph Laplacian Regularizer: A lattice 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 [18], [19]. The un-normalized Laplacian regularization penalizes the sum of the squared differences between the lattice values at adjacent vertices

$$b^T \mathbf{K}_L b = \sum_{\text{adjacent } a_i, a_j} (b_i - b_j)^2. \tag{4}$$

The Laplacian matrix $\mathbf{K}_L$ is

$$\mathbf{K}_L = \text{diag}(\mathbf{1}^T \mathbf{A}) - \mathbf{A}$$

where $\mathbf{1}$ is the $m \times 1$ all-ones vector, $\mathbf{A}$ is the graph adjacency matrix, and $\text{diag}(\mathbf{1}^T \mathbf{A})$ is a diagonal matrix with $\mathbf{1}^T \mathbf{A}$ on the diagonal.

Solving for a lattice that minimizes the empirical risk (2) and also minimizes the Laplacian given by (4) forces the values chosen for adjacent nodes in the lattice to be close, and is expressed as

$$\hat{b} = \arg\min_{b \in \mathbb{R}^m} \|\mathbf{W}b - y\|_2^2 + \lambda b^T \mathbf{K}_L b \tag{5}$$

where the regularization parameter $\lambda > 0$ tradesoff the two goals.

The optimization (5) has a closed-form solution

$$\hat{b} = (\mathbf{W}^T \mathbf{W} + \lambda \mathbf{K}_L)^{-1} \mathbf{W}^T y. \tag{6}$$

Because $\mathbf{W}^T \mathbf{W}$ is positive semidefinite and $\mathbf{K}_L$ is positive definite, the inverse in (6) is always well-defined for $\lambda > 0$. Further, because both $\mathbf{W}$ and $\mathbf{K}_L$ are sparse, the matrix inversion in (6) can be computed efficiently by sparse Cholesky factorization (for instance the `ldivide` command in MATLAB).

2) Graph Hessian Regularizer: We argue that the graph Laplacian regularizer is suboptimal for many applications (including color management [5]) because it penalizes linear trends. To avoid penalizing the estimation of linear functions, we prefer to regularize by penalizing the second-order difference in each dimension of the lattice, summed over the d dimensions

$$\sum_{k=1}^{d} \sum_{\substack{a_h, a_i, a_j \\ \text{adjacent in} \\ \text{dimension } k}} \big((b_h - b_i) - (b_i - b_j)\big)^2$$

$$= \sum_{k=1}^{d} \sum_{\substack{a_h, a_i, a_j \\ \text{adjacent in} \\ \text{dimension } k}} \big(b_h - 2b_i + b_j\big)^2 = b^T \mathbf{K}_H b \tag{7}$$

where $\mathbf{K}_H$ is an $m \times m$ matrix. Unfortunately, this $\mathbf{K}_H$ is not positive definite – the smallest two eigenvalues are zero. In order to use it as a regularizer, we form a positive definite matrix by adding a touch of identity: $\tilde{\mathbf{K}}_H = \mathbf{K}_H + 10^{-6}\mathbf{I}$. This small correction adds a touch of bias, and acts like a ridge regularizer in that it slightly penalizes $\sum_i b_i^2$. We refer to $b^T \tilde{\mathbf{K}}_H b$ as the *graph Hessian* regularizer.

Using the graph Hessian regularizer, the lattice nodes that minimize the empirical risk (2) are given by

$$\hat{b} = \arg\min_{b \in \mathbb{R}^m} \|\mathbf{W}b - y\|_2^2 + \lambda b^T \tilde{\mathbf{K}}_H b \tag{8}$$

where the regularization parameter $\lambda > 0$ tradesoff the two goals of training-accuracy and smoothness. The problem (8)

has a closed-form solution

$$\hat{b} = (\mathbf{W}^T\mathbf{W} + \lambda\tilde{\mathbf{K}}_H)^{-1}\mathbf{W}^T y \tag{9}$$

which, again, can be efficiently computed via sparse Cholesky factorization due to the sparse nature of $\mathbf{W}$ and $\tilde{\mathbf{K}}_H$.

In some cases, different regularization may be needed in different parts of the LUT. For example, in color management it can be the case that certain parts of the colorspace are known to exhibit high curvature, whereas other parts are fairly linear. Cell-specific regularization can be implemented by modifying the $\mathbf{K}_H$ matrix so that a separate weight is applied to each term of the summation in (7).

3) Alternative Regularizers: Regularization may be applied with aims other than functional smoothness. For instance, one may start with an existing LUT toward which the lattice regression estimate may be regularized. In the color management of printers, one often needs to update an existing LUT to compensate for expected data drift or a rough estimate of the LUT may be given from fitting a model, such as the Yule–Nielsen Neugebauer [20]. Alternatively, a regression function with few free parameters could be first fit to the data, and then one may bias lattice regression toward this. For details on how to include such a regularizer, see the prior work on *global bias* [7, Sec. 2.3].

C. Sensitivity to Noise

Suppose the ith training sample output y_i is corrupted by independent identically distributed (i.i.d.) additive zero-mean noise ϵ_i such that $Y_i = y_i + \epsilon_i$, where $E[\epsilon_i] = 0$ and $\text{var}(\epsilon_i) = \sigma^2$ for $i = 1, \ldots, n$. Then the lattice regression LUT constructed from $\{Y_i\}_{i=1:n}$ will be $\{a_j, \hat{b}_j(\epsilon)\}_{j=1:m}$, where $\hat{b}(\epsilon) = (\mathbf{W}^T\mathbf{W} + \lambda\tilde{\mathbf{K}}_H)^{-1}\mathbf{W}^T(y+\epsilon)$ from (9). Let the weight vector $v \in [0,1]^{2^d \times 1}$ specify the linear interpolation weights for some test sample x for this lattice. Then due to the linearity of expectation, the lattice regression estimate $\hat{Y}(\epsilon)$ of the function value for x is unbiased by the noise

$$E_\epsilon\left[\hat{Y}\right] = v^T(\mathbf{W}^T\mathbf{W} + \lambda\tilde{\mathbf{K}}_H)^{-1}\mathbf{W}^T y.$$

In applications like color management where the LUT is learned to invert a black box model, such as a printer, the training sample inputs x_j are actually the outputs of the black box model and thus may be noisy. Additive noise on x_j affects the linear interpolation weight matrix W in two nonlinear ways: 1) the linear interpolation weights multiply the noise in each dimension as per (17) and 2) noise on the x_j can change which lattice cell it affects, nonlinearly changing the weight matrix W as per (18). The same effect occurs if there is additive noise on a test sample x.

III. Related Regression Methods

Lattice regression belongs to two major families of regression methods: *structural risk minimization* and *splines*. It is a structural risk minimization method because (8) selects a function (by choosing the lattice node values $\{b_j\}_{j=1:m}$ that are to be linearly interpolated) that tradesoff minimization of an empirical risk term with a regularizer. In the rest of this section, we focus on lattice regression's relationship to other spline methods.

A. Splines

The word *spline* is used to mean various related things, here, we use one common definition that a spline is *a piecewise polynomial function*. Lattice regression does in fact produce piecewise polynomial functions because linear interpolation is applied in each cell of the lattice and, when applied to the vertices of any parallelotope, linear interpolation produces a polynomial function [21, Th. 11]. For example, in 3-D d-linear interpolation produces a piecewise function of the form

$$\begin{aligned} f(x) = {} & c_0 + c_1x[1] + c_2x[2] + c_3x[1]x[2] + c_4x[3] \\ & + c_5x[1]x[3] + c_6x[2]x[3] + c_7x[1]x[2]x[3] \end{aligned}$$

where $x[k]$ is the kth component of the test sample k, and $c_j > 0$ for all j except in degenerate cases.

A more convenient representation is in terms of basis functions, also a common formulation for other types of splines [22]. In this perspective, a spline is a linear combination of some piecewise-polynomial basis function $k : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$, repeated and centered at each knot (i.e., lattice node), such that

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

where b_j is the weight given to the jth basis function centered at the knot (lattice node) a_j. For 1-D lattice regression (see Appendix for details)

$$\hat{f}_{LR}(x) = \sum_{j=1}^{m} b_j(1 - |x - a_j|)_+$$

where $(\cdot)_+ = \max(\cdot, 0)$ and the basis function is $k_{lr} = (1 - |x - a_j|)_+$. This is also known as the linear b-spline basis function.

We compare the linear interpolation basis k_{lr} with the cubic interpolation basis k_c [23], and cubic b-spline basis k_s [22]. These basis functions can be expanded to d-dimensions via the tensor product of multiple 1-D bases, constructed as

$$k^d(x, x') = \prod_{m=1}^{d} k\big(x[m], x'[m]\big). \tag{10}$$

Linear b-splines are not a common basis function for splines, in large part because they do not belong to the class of splines called *smoothing splines*. A smoothing spline is a piecewise-polynomial function that arises as the solution to the following structural risk minimization problem:

$$\arg\min_{g \in \mathcal{G}_{\{a\}}} \sum_{i=1}^{n} (y_i - g(x_i))^2 + \lambda \int \nabla g(x)^T \mathbf{1}\mathbf{1}^T \nabla g(x) dx \tag{11}$$

where $\mathbf{1}$ is the vector of all ones, and $\mathcal{G}_{\{a_j\}}$ is a pre-defined set of piecewise-polynomial functions whose pieces are delineated by the knot locations $\{a_j\}$ [22]. An important special case of (11) is when a knot is placed at each training point, this is called a *natural spline* and $\mathcal{G}_{\{a_j\}}$ is the set of all functions

that are differentiable and have an absolutely continuous first derivative. Although, natural splines are arguably the most commonly used instantiation of splines, we do not focus on them here because—for efficiency—we place knots at the lattice nodes, independent of the training data.

When the knots are fixed in a rectangular lattice, the solution to (11) is known as a *tensor* spline and there exists a matrix $\mathbf{K}$, completely determined from the spline basis k and the knot locations $\{a_j\}$, such that (11) can be equivalently stated as

$$\arg\min_{\mathbf{b}\in\mathbb{R}^m} \sum_{i=1}^{n} \left(y_i - b^T \mathbf{k}_{x_i}\right)^2 + \lambda b^T \mathbf{K} b \tag{12}$$

where $\mathbf{k}_{x_i} = \left(k(a_1, x_i), k(a_2, x_i), \ldots, k(a_m, x_i)\right)^T$ [22]. See [22] for details on how to construct the regularization matrix $\mathbf{K}$ for an arbitrary twice-differentiable basis function k.

Because $\mathbf{K}$ depends on the second partial derivative of g, it cannot be defined for the linear b-spline basis function used by lattice regression. Thus, for lattice regression, we use the discrete Hessian matrix given in (7) in place of $\mathbf{K}$. In Section III-B, we compare lattice regression to smoothing tensor splines using the cubic interpolation basis k_c and the cubic tensor b-spline basis k_s.

To summarize the above discussion, the proposed lattice regression is a spline method that: 1) uses a fixed rectangular grid of knots independent of the training samples; 2) uses a d-linear interpolation function; and 3) uses a discrete regularizer (for example, the proposed graph Hessian).

B. Comparison With Higher Order Basis Functions

Lattice regression provides fast test evaluations by applying d-linear interpolation to a regular lattice of knots in d dimensions, such that only 2^d lattice values are needed to calculate the output for each test point. In this section, we consider the alternative approach of using a higher order interpolation, such as the cubic interpolation basis or spline basis to evaluate a lattice learned with the corresponding higher order basis function and its second-derivative smoothing spline regularizer. The larger support of these basis functions requires 4^d lattice points to calculate an estimated function value, resulting in significantly longer test times than d-linear interpolation.

Further, the arithmetic of linear interpolation is amenable to simple operations, such as bit-shifting, on fixed-point inputs and can be implemented very efficiently in dedicated hardware [2], [3]. This is particularly beneficial in applications, such as printer color management where dedicated hardware is used to perform test evaluations. In contrast, these low-level speed-ups are not possible for the computation of cubic interpolation and spline weights.

In general, the accuracy differences between the different interpolation functions will depend on the true function being approximated, with smoother functions being better approximated with the higher order basis functions. We illustrate the accuracy differences with the following simulation. The domain was constrained to $[0, 1]^3$, and the function to be approximated was the Gaussian mixture $f(x) = \sum_{i=1}^{10} p(x; \mu_i, \sigma_i^2)$ where p is a Gaussian pdf with mean $\mu_i \sim U([-.5, 1.5]^3)$ and variance $\sigma_i^2 = 1$. In all experiments, a training sample of size n was drawn i.i.d. with $X_i \sim U([0, 1]^3)$ and $Y_i = f(X_i)$ for $i = 1, \ldots, n$. Regularly-spaced 3-D lattices with $\tilde{m} \times \tilde{m} \times \tilde{m}$ knots were constructed with $\tilde{m} \in \{9, 17, 33\}$ so that the number of cells in each dimension of the lattice is a power of two for efficient implementation. The root mean square error between the true function f and the estimated $\hat{f}$ (interpolated from the lattice) was then computed for each method at 10 000 uniformly drawn locations in $[0, 1]^3$. Based on a preliminary study of performance for different choices of the regularization parameter λ (see [4] for more experimental details and results), we fixed $\lambda = 10^{-6}$ for all the experiments shown here.

The results in Fig. 3 show that using higher order interpolation functions may not provide an accuracy advantage for their higher computational cost. Specifically, with few training samples and many lattice nodes, lattice regression performs almost identically to the higher order basis functions.

IV. Application to Color Management

Due to the physics involved in the printing process, printers exhibit the most nonlinear and unpredictable color mappings among all digital imaging devices [1]. For printers, the empirical approach to *inverse device characterization* begins with printing a page of color patches for a set of input values that span the gamut (the range of displayable colors) of the device. These patches are subsequently measured with a spectrophotometer under standard illumination conditions and recorded as output values in a device-independent color space. For many printers, it is possible to specify the input as CMYK, but we restrict ourselves to RGB inputs and treat the conversion and printing as a single black-box model, this emulates a typical consumer printing environment. For the device-independent output space, we use the typical choice of CIELAB, which is an approximately perceptually uniform colorspace. From these training pairs of (RGB, CIELAB) values, one estimates the *inverse mapping* f : CIELAB $\rightarrow$ RGB that specifies what RGB input to send to the printer in order to reproduce a desired CIELAB color. Estimating f is challenging for a number of reasons [1]: 1) f is often highly nonlinear; 2) although it can be expected to be smooth over regions of the colorspace, it is affected by changes in the underlying printing mechanisms—for example, undercolor removal—that can introduce discontinuities; and 3) device instabilities and measurement error introduce noise into the training data.

Fig. 4 shows a typical color-managed system. The estimated inverse mapping is implemented with a 3-D LUT with nodes $\{a_j\} \subset$ CIELAB that are regularly spaced in each dimension. This is followed by an array of 1-D LUTs used for *calibration*: the R, G, and B channels are independently corrected to a linear neutral response. Once estimated, the entire set of LUTs can be stored in an ICC profile, a standardized color management format, developed by the ICC. Input CIELAB colors that are not a node of the 3-D LUT are interpolated and, although interpolation technique is not specified in the standard, the most commonly used method is *trilinear interpolation* [24].

The standard solution to estimating the inverse mapping f is a two-step process. First, one applies regression to estimate

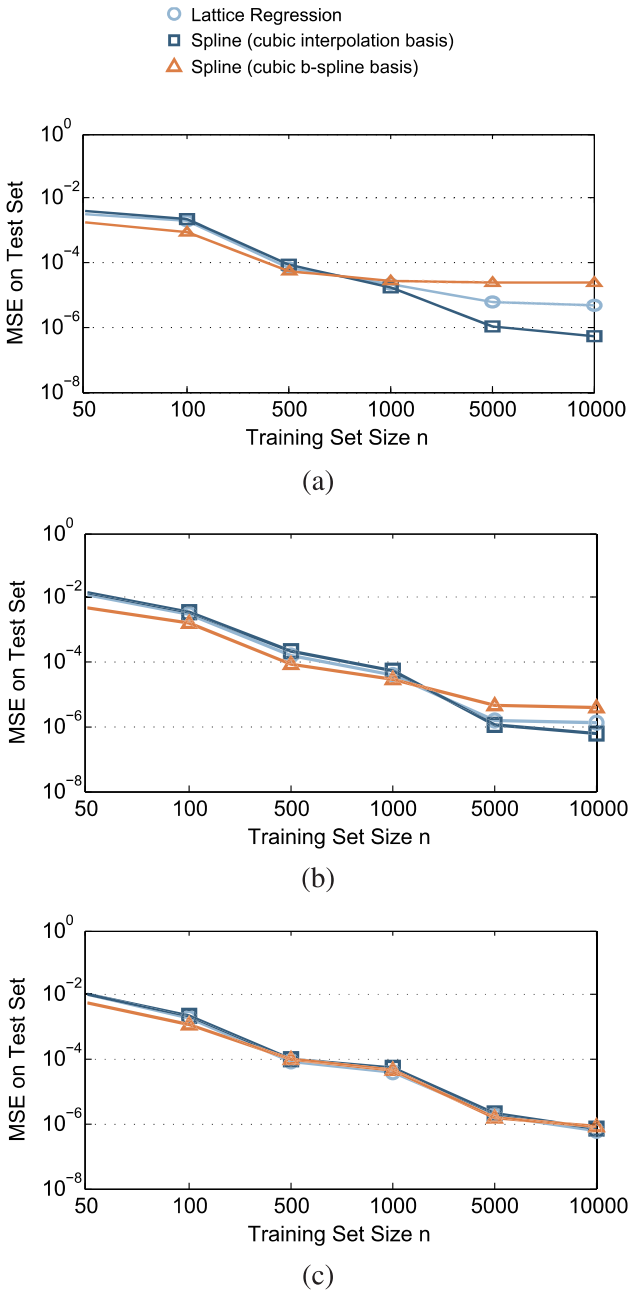


Fig. 3. Illustration of the comparative accuracy of lattice regression with higher order interpolations for estimating a $d = 3$ dimensional Gaussian mixture. Shown are RMS test error for $k = 10\,000$ uniformly drawn locations in $[0, 1]^3$ for varying lattice sizes m and training sample sizes n. (a) $m = 9^3$ lattice nodes, (b) $m = 17^3$ lattice nodes, and (c) $m = 33^3$ lattice nodes.

an intermediate function that fits the observed training data. Once estimated, this intermediate function is evaluated at the nodes of the lattice LUT and discarded. We compare lattice regression to this standard approach applied with three state-of-the-art regression techniques. The first two, shown previously to work well for color management, are local ridge regression [25] and local Tikhonov regression [26]; they are variations on local linear regression that differ only in the regularization applied. For these, we use an enclosing k-NN neighborhood [4], [25] to automatically choose a set of local training points that enclose the test point in a convex hull, eliminating the need to cross-validate a neighborhood size for

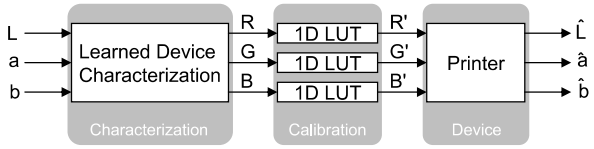


Fig. 4. Color-managed printer system. For evaluation, errors are measured between (L, a, b) and $(\hat{L}, \hat{a}, \hat{b})$ for a given device characterization.

the local regressions. We also compare to Gaussian process regression [27] (also known as *kriging*), which is a popular kernel-based regression technique that has been successful in diverse low-dimensional problems, such as geospatial interpolation and inverse dynamic control in robotics.

A. Experimental Details

Experiments were run with an HP Photosmart D7260 ink jet printer and a Brother HL-4040CDN laser printer. Training samples were created by printing the standard 918 sample Gretag MacBeth TC9.18 RGB target. CIELAB values for the printed color patches were measured with an X-Rite iSis spectrophotometer using D50 illuminant at a 2° observer angle and UV filter. As shown in Fig. 4 and as is standard practice for this application, the data for each printer was first gray-balanced using 1-D calibration LUTs for each color channel (see [1] for details). The same 1-D LUTs were shared among the techniques applied in order to reduce the number of confounding variables in the experiments. These 1-D LUTs were estimated using local Tikhonov-regularized regression with enclosing k-NN neighborhood [25], [26] and the regularization parameter $\lambda = 1$. The $17 \times 17 \times 17$ nodes of each 3-D LUT formed a regular lattice spanning $L \in [0, 100]$, $a \in [-100, 100]$, and $b \in [-100, 100]$.

For each printer, RGB values were drawn uniformly, printed, and measured, forming an independent set of RGB $\rightarrow$ CIELAB pairs. This set provided three uses in our experiments. The first role is as a test set, since the resulting CIELAB values are guaranteed to be within the gamut of each printer, we used this set to assess the color accuracy of the constructed LUTs. That is, the CIELAB values were processed by each LUT producing $\widehat{\text{RGB}}$ which were then printed, the subsequently measured $\widehat{\text{CIELAB}}$ values were compared to the original CIELAB values. To compare, we used the ΔE_{2000} metric which is standard in color management and is a perceptually-corrected Euclidean distance in CIELAB. The second role is to characterize the innate variability in the printing/measuring process. This was done as follows: 1) we printed and measured the 918 random RGB color patches (without modification) five times for each printer; 2) we computed the mean Lab value for each patch over the five prints; and 3) we computed the average ΔE_{2000} difference, across all 918×5 printed patches, between the measured Lab value and the corresponding mean Lab value. The third role of this set of random RGB values was in choosing the regularization parameter for each method. The value of λ for each algorithm was chosen to produce a LUT that minimized $\sum_i \|\widehat{\text{RGB}}_i - \text{RGB}_i\|_2$ for the test CIELAB values. Table I shows for each method the parameter settings tested and the

TABLE I
REGULARIZATION PARAMETER CHOICES

	λ Brother	λ HP	λ choices
Local ridge regression	10	1	$\{10^{-3}, \ldots, 10^{3}\}$
Local Tikhonov regression	100	10	$\{10^{-2}, \ldots, 10^{6}\}$
Lattice regression	10^{-5}	10^{-5}	$\{10^{-8}, \ldots, 10^{0}\}$

TABLE II
ΔE_{2000} PERFORMANCE ON HP PHOTOSMART D7260 INKJET PRINTER

	Mean	# Errors > 2 ΔE_{2000}
Lattice regression	0.98	58
Local ridge	1.43	181
Local Tikhonov	1.30	143
GPR	1.67	282

TABLE III
ΔE_{2000} PERFORMANCE ON BROTHER HL-4040CDN LASER PRINTER

	Mean	# Errors > 2 ΔE_{2000}
Lattice regression	1.79	317
Local ridge	1.83	341
Local Tikhonov	1.82	335
GPR	2.16	417

chosen parameters for each printer. Parameters for Gaussian process regression were chosen using the maximum likelihood method detailed in [27].

Tables II and III show the performance on the 918 test samples in terms of ΔE_{2000} error.[1] To put the results in perspective, the innate ΔE_{2000} variability (described above) for the inkjet and laser printer, respectively, was 0.46 and 0.52. For further perspective, note that a ΔE_{2000} error of $1-2$ is generally considered a "just noticeable difference." On both printers, lattice regression achieves the smallest number of colors above this threshold. On the inkjet printer, lattice regression provides a 24% mean-error improvement over the next-best performer, local Tikhonov regression. On the laser printer, lattice regression gives only a small 1% mean-error improvement, but for both printers a Wilcoxon one-sided signed rank test indicated that lattice regression performs consistently better than the other methods.[2] We hypothesize that the difference on the laser printer is smaller because these results are close to the limit of how well one could control this laser printer.

[1]Using ΔE_{1994} or ΔE_{1976} instead of ΔE_{2000} results in the same pairwise ranking of algorithms under the Wilcoxon signed rank test.

[2]The Wilcoxon signed rank test is a nonparametric significance test of the difference in error between competing algorithms, it allows one to determine if the median of this difference distribution is nonzero, thus addressing the consistency by which one algorithm outperforms another. This significance test is more appropriate than the Student's t-test, as we do not expect the difference in error to be normally distributed.

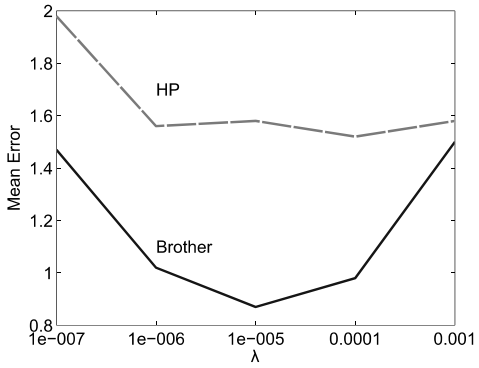


Fig. 5. Lattice regression test error for different choices of regularization parameter λ.

To assess the sensitivity of the proposed lattice regression to the regularization parameter λ, we printed and measured the test patches for a range of parameter settings. The result of this analysis is presented in Fig. 5, which shows the mean ΔE_{2000} error on the test patches. On both printers, the mean error is robust to the value of the parameter λ within one order of magnitude of the optimal value. One sees that for the HP, our cross-validation of the regularization parameter λ did not quite produce the optimal λ in terms of the printed ΔE_{2000} error, which would have been achieved by 10^{-4}.

V. APPLICATION TO OMNIDIRECTIONAL SUPER-RESOLUTION

We next illustrate the usefulness of the fast-evaluation time of lattice regression, by showing how it can be used to significantly improve the state of the art for the super-resolution of unregistered omnidirectional images. Omnidirectional cameras have been shown to be useful for autonomous robots, particularly for visual homing and motion-estimation tasks [28]–[30]. However, even medium-resolution omnidirectional sensors are expensive, so effective super-resolution is desirable. The state of the art in omnidirectional super-resolution is arguably Arican and Frossard's spherical Fourier transform (SFT) method [31]–[33]. We will show that by using lattice regression, we can outperform this SFT super-resolution in terms of peak signal-to-noise-ratio (PSNR), scalability, and visual homing performance.

Until recently, most of the efforts on image super-resolution assumed the images were perfectly registered, both for planar image super-resolution [34], [35], and omnidirectional super-resolution [36], [37]. More recent planar image super-resolution algorithms simultaneously register and reconstruct [38], [39]. For unregistered omnidirectional images, Arican and Frossard [31]–[33] extended the method of He *et al.* [38] to formulate a nonlinear least-squares optimization problem which is solved with the Levenberg–Marquardt algorithm. As in this previous work, we formulate the problem as finding the registration parameters that reconstruct a high-resolution image that best matches the given unregistered images. However, this is a nonconvex optimization problem, and in practice has many local minima. The key to our improved

TABLE IV
NOTATION

Symbols	Description
Z	True spherical function
N	Number of given images
Z_i	ith given image
$\mathcal{I}$	A set of indices
i^C	The set of indices of all images except the ith
M^2	Number of pixels in each given image
L^2	Number of pixels in super-resolved image
$\hat{Z}$	Estimated spherical function
θ, ϕ	Spherical coordinate angles
Ω	Domain of Euler angles
g_i	Euler angles for ith image Z_i
$R(g)$	Rotation matrix corresponding to Euler angles g
G_0	Reference grid: M regular samples of θ and ϕ
G^{g_i}	G_0 rotated by g_i [see (14)]

performance is that we better search this optimization space by using a global optimization strategy, and we search the space more efficiently by using lattice regression to quickly evaluate each candidate output. We refer to this approach as super-resolution of omnidirectional images by lattice regression (SOLaR).

In the next section, we overview the SOLaR approach. Then, in Section V-B, we describe how we parameterize image registration on the sphere. In Section V-C, we describe how we greedily globally optimize (13). Next, in Section V-D, we explain how we apply lattice regression to a spherical grid to form the estimates of the high-resolution image needed in the global optimization. In Section V-E, we describe our experimental setup, and in the remaining sections discuss the results.

A. Overview of SOLaR

The notation used in this section is summarized in Table IV. We take as given N omnidirectional images $\{Z_i\}, i = 1, \ldots, N$, each defined on a regular spherical grid with M^2 sampling nodes, but with unknown registration. We assume that there exists a real-valued function Z defined over the unit sphere, and we treat each of the N given images $Z_1, Z_2, \ldots, Z_N$ as samplings of Z. The objective is to form an estimate $\hat{Z}$ of Z, and then evaluate $\hat{Z}$ for the desired higher resolution regular spherical grid with L^2 sampling nodes. While the super-resolution problem assumes $L > M$, SOLaR can also be used to solve the general resampling problem for arbitrary L and M.

We propose estimating registration parameters $\hat{g} = \{\hat{g}_1, \hat{g}_2, \ldots, \hat{g}_N\}$ that minimize the total leave-one-out reconstruction error for the N given images. That is, we would like to solve for

$$\hat{g} = \arg\min_{g \in \Omega^N} \sum_{i=1}^{N} \left\| Z_i - \hat{Z}(g_i \,|g_{i^C}) \right\|_F \tag{13}$$

where $g = \{g_1, g_2, \ldots, g_N\}$ parameterizes the relative rotations of the N images, $\|\cdot\|_F$ is the Frobenius norm, and $\hat{Z}(g_i \,|g_{i^C})$ is the estimated function evaluated on the spherical grid G^{g_i} where $\hat{Z}$ is estimated from the complement set i^C of all images except the ith image, with rotations given by g_{i^C} (this notation is explained in further detail in Section V-B).

To make solving the global optimization problem (13) feasible, we use a greedy global optimizer to efficiently search over the image registrations g, and a sphere-adapted version of lattice regression to quickly evaluate each estimated image $\hat{Z}(g_i \,|g_{i^C})$.

B. Spherical Registration Using Euler Angles

We use Euler angles to describe the relative registrations of the given images. Denote the Euler angles for the ith image as $g_i = (\alpha_i, \beta_i, \gamma_i) \in \Omega$, where $\Omega = [-\pi, \pi) \times [0, \pi] \times [-\pi, \pi)$. Euler angles parameterize a 3×3 rotation matrix R in terms of rotations about z-axis, y-axis, and then again about the z-axis

$$R(g_i) = R_Z(\alpha_i)\ R_Y(\beta_i)\ R_Z(\gamma_i)$$

where $R_Z(\alpha_i)$ is the rotation matrix that defines rotation about the z-axis by angle α_i.

Points on the unit sphere can be described by spherical coordinates (r, θ, ϕ), where $r = 1$ is the radius, $\theta \in [0, \pi]$ is the co-latitude angle, and $\phi \in [-\pi, \pi)$ is the longitude angle, and thus each image Z_i can be described by a $M \times M$ matrix of pixel values corresponding to a regular sampling of the coordinates θ and ϕ. Specifically, we take as a reference grid the regular spherical grid G_0 with M^2 total samples made up of M regularly spaced samples in θ and M regularly spaced samples in ϕ

$$G_0 = \left\{ (\theta_j, \phi_k) = \left(\frac{\pi}{M}(j + 1/2), \frac{2\pi}{M}(k - M/2) \right) \right\} \quad \text{for } j, k = 0, 1, 2, M - 1.$$

Note that G_0 can be described as a matrix with (j, k)th entry given by the pair (θ_j, ϕ_k).

Let the function τ convert spherical coordinates (r, θ, ϕ) to Cartesian coordinates, and let τ^{-1} convert Cartesian coordinates to spherical coordinates so that $\tau^{-1}(\tau(r, \theta, \phi)) = (r, \theta, \phi)$. Then rotating the reference grid G_0 by a rotation parameterized by Euler angles g_i produces a new set of $M \times M$ spherical angle coordinates that we denote G^{g_i}, where

$$G^{g_i} = \left\{ \tau^{-1}\left(R(g_i)\tau(1, \theta, \phi)\right) \text{ for all } (\theta, \phi) \in G_0 \right\}. \tag{14}$$

We treat the ith given image Z_i as being a sampling of the underlying function Z on an unknown grid G^{g_i}, that is, $Z_i = Z(G^{g_i})$. Given an index set $\mathcal{I} \subseteq \{1 : N\}$, let the corresponding indexed set of given images be denoted $Z_{\mathcal{I}} = \{Z_i\}_{i \in \mathcal{I}}$, and let $g_{\mathcal{I}} = \{g_i\}_{i \in \mathcal{I}}$ be their corresponding set of rotations. We use $\mathcal{I}^C$ to denote the complement set of indices, $\{1 : N\} \backslash \mathcal{I}$. Last, let $\hat{Z}(g_i \,|g_{\mathcal{I}})$ denote the $M \times M$ matrix formed by evaluating the estimated function $\hat{Z}$ on the grid G^{g_i}, where $\hat{Z}$ has been estimated from the set of given images $Z_{\mathcal{I}}$ with estimated rotations given by $G^{g_{\mathcal{I}}}$.

C. SOLaR's Greedy Global Optimization

The registration cost function given in (13) has many local minima and, given N images, it requires searching the $3(N-1)$ dimensional space of relative Euler angles to perform super-resolution. Here, we present an efficient (and approximate) greedy method.

Initialize: Set the iteration counter $t = 0$ and initialize the training set with the first image: $\mathcal{I}_t = \{1\}$ with registration $\hat{g}_1 = (0, 0, 0)$. Further, initialize the vector of reconstruction errors $\epsilon = (0, 0, \ldots, 0) \in \mathbb{R}^{N\times 1}$, and error threshold $\epsilon^* = 0$.

While $\mathcal{I}_t^C \neq \emptyset$.

Add Images: Images are registered to the training set and immediately added when the reconstruction error is below the threshold ϵ^*. Let π be a random permutation of the indices in $\mathcal{I}_t^C$ and set $t\_ = t$.

For $i = \pi_1, \pi_2, \ldots, \pi_{|\mathcal{I}_t^C|}$, Find the registration $\hat{g}_i$ that best matches Z_i to its estimate from $\hat{g}_{\mathcal{I}_t}$

$$\hat{g}_i = \underset{g_i \in \Omega}{\arg\min} \quad \left\| Z_i - \hat{Z}(g_i \,|\hat{g}_{\mathcal{I}_t}) \right\|_F \tag{15}$$

and update $\epsilon_i = \left\| Z_i - \hat{Z}(\hat{g}_i \,|\hat{g}_{\mathcal{I}_t}) \right\|_F$. If $\epsilon_i < \epsilon^*$, then add image i to the training set $\mathcal{I}_{t+1} = \mathcal{I}_t \bigcup i$, and update $\epsilon^* = \epsilon_i$ and $t = t + 1$.

Ensure an Image was Added: If no image meets the threshold ϵ^*, then the training set remains unchanged. To ensure progression of the algorithm in this case, the image with the lowest reconstruction error is added. Let $i^* = \underset{i\in\mathcal{I}_t^C}{\arg\min}\, \epsilon_i$, if $t = t\_$, then add image i^* to the training set $\mathcal{I}_{t+1} = \mathcal{I}_t \bigcup i^*$, and update $\epsilon^* = \epsilon_{i^*}$ and $t = t + 1$.

Swap: To mitigate poor registrations that are likely to occur at the early rounds of the algorithm, we allow the worst-registered image a chance to be re-registered while ensuring that the algorithm still completes in $N-1$ iterations. Compare the image in $\mathcal{I}_t$ with the largest reconstruction error $i_{\max} = \underset{i\in\mathcal{I}_t}{\arg\max}\, \epsilon_i$ to the image in $\mathcal{I}_t^C$ with the smallest reconstruction error $i_{\min} = \underset{i\in\mathcal{I}_t^C}{\arg\min}\, \epsilon_i$. If $\epsilon_{i_{\min}} < \epsilon_{i_{\max}}$, then swap $Z_{i_{\min}}$ and $Z_{i_{\max}}$ in and out of the training set, respectively, by setting $\mathcal{I}_{t+1} = \left(\mathcal{I}_{t+1} \backslash i_{\max}\right) \bigcup i_{\min}$, and update $\epsilon^* = \epsilon_{i_{\min}}$ and $t = t + 1$.

To perform the global optimization in (15) for each $i \in \mathcal{I}_t^C$, we used the meta-heuristic global optimizer *fully-informed particle swarm* (FIPS) [40], which is a variant of the popular particle swarm optimizer (PSO) [41]. PSO begins with a number of random guesses (we use the number of search dimensions squared) called birds. At each iteration, the optimizer moves each bird to a new point in the domain, where each bird is moved in part toward the best location it has ever seen, in part toward the best location any of the birds has ever seen, and its movement has an inertial term that encourages exploration. FIPS imposes further connections between the birds movements, we used the connecting ring neighborhood variant [40].

Note that the greedy registration algorithm described above always finishes after $N - 1$ iterations. Each time new images are added to the training set $\mathcal{I}_t$, the function $\hat{Z}$ must be re-estimated, so that lattice regression must be trained at most N times, and often fewer than N times because more than one image can be added to the training set at once in the add-image step. In addition, if each global optimization for (15) is allowed V guesses, then there will be $O(VM^2N^2)$ total evaluations of the estimated function $\hat{Z}$, and so evaluating $\hat{Z}$ must be fast. To that end, we use lattice regression adapted for the sphere, as described next.

To improve results, the greedy registration algorithm can be run multiple times, initializing each run with a subset of the currently registered images and their estimated registrations. In our experiments we perform two runs of the above algorithm, where on the second run we set the initial training image set $\mathcal{I}_0$ to be the set of images with reconstruction error lower than the median error, this heuristic improved the resulting registration in most cases. The inclusion of the swap step is also a heuristic that reduces the propensity of this greedy approach to be trapped in poor local minima, by swapping out images that were initially poorly registered.

D. Lattice Regression on the Sphere

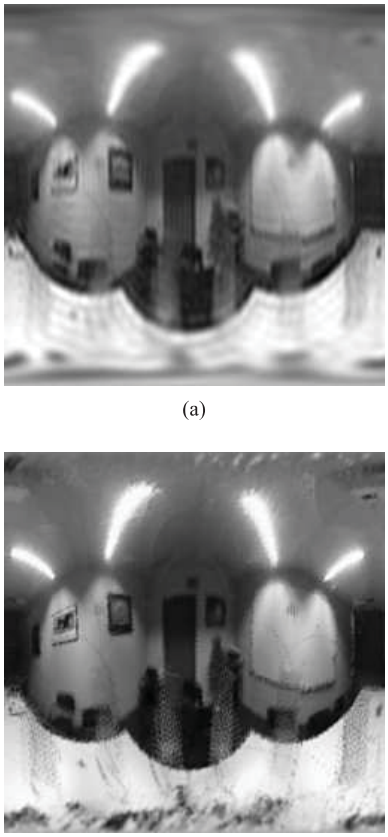
We apply lattice regression to a regular co-latitude/longitude lattice on the sphere, where each grid node $a_j \in [0, \pi] \times [-\pi, \pi)$. We use m_θ lattice nodes over the co-latitude angle range $[0, \pi]$, and m_ϕ samples over the longitude angle range $[-\pi, \pi)$, for a total of $m = m_\theta \times m_\phi$ lattice nodes. In our experiments, the lattices are all regularly-spaced in terms of θ and ϕ.

This spherical lattice differs from the standard rectangular grid only in that it wraps around so that the longitude angle node for $-\pi$ and π are the same node, which simply changes the specification of the graph Hessian regularizer for this node and its neighbors. However, it is important to note that the use of an equiangular grid implies that the geodesic spacing of nodes will vary over the surface of the sphere. Accordingly, one could easily modulate the strength of the Hessian regularizer by dividing by the geodesic distance between adjacent lattice nodes, which would ensure a more consistent regularization in terms of *geodesic* smoothness. However, for simplicity, we opted not to make this adjustment here.

E. Super-Resolution Results

We compare SOLaR and SFT in terms of visual quality, super-resolution PSNR, computational time, and performance on visual homing using the super-resolved images. In the experiments, the low-resolution images, as well as the super-resolved image, span the entire surface of the sphere. However, if one was to use low-resolution images that did not span the entire surface of the sphere, the SOLaR method can still be applied to produce a super-resolved image over the entire sphere. In either case, if there are cells within the high-resolution lattice that do not contain data, lattice regression will produce a smooth gradient that is affected by the nearest cells that *do* contain data.

1) Super-Resolution from Real Low-Resolution Images: First, we illustrate SOLaR and SFT for super-resolving thirty-seven 64×64 images captured by a commercial camera



(a)

(b)

Fig. 6. (a) 256 × 256 SFT super-resolved image and (b) 256 × 256 SOLaR super-resolved image.

with unknown orientations. Most commercial omnidirectional cameras do not capture the scene on a full sphere as the view is obstructed by physical mounts. In this case, the captured 64 × 64 images were reduced to 15 × 64 images after removing occluded parts, and these images were given as input to SOLaR and SFT. The resulting 256 × 256 super-resolved images are shown in Fig. 6(a) and (b). Although no ground truth is available for the high-resolution image, the SOLaR super-resolution is generally sharper than the SFT super-resolution.

2) Super-Resolution Performance Versus Number of Low-Resolution Images and Differing Registration Uncertainty: Next, we present experimental results for two real omnidirectional images from a public domain image database [42], and for a synthetically generated omnidirectional image, which was used previously in evaluating the SFT method [31], [33]. For these experiments, low-resolution images were generated by down-sampling the original image in the SFT domain (as done in [31]–[33]). The low-resolution images were sampled on uniformly random rotations of the reference grid G_0. The super-resolution algorithms are given the low-resolution images and their grids with a random registration error drawn uniformly from $[-\psi^\circ, \psi^\circ]$ in latitude and longitude independently, and the maximum registration error ψ is given to the super-resolution method.

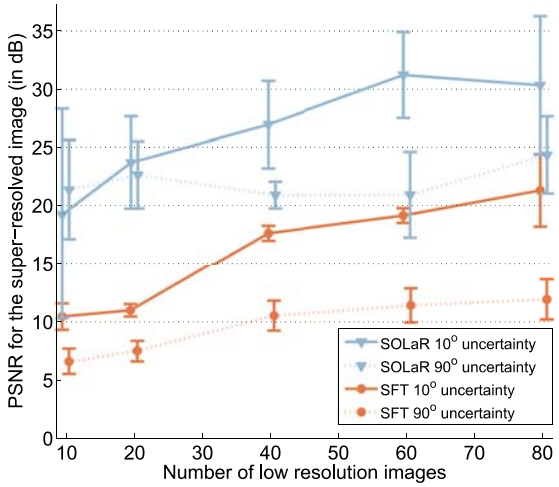


Fig. 7. Reconstruction PSNR of a 128 × 128 image versus the number of low-resolution 16 × 16 images with the registration uncertainty of 10° and 90°. Results were averaged over all three test images.

Fig. 7 shows the results for super-resolving a 128 × 128 image from a varying number of 16 × 16 images. The proposed SOLaR method performs roughly 10 dB better on average than the benchmark SFT method. The solid lines compare the methods when the registration is known within 10°. The dashed lines compare the methods when the registration is completely unknown (that is, there is 90° registration uncertainty). The reported PSNR was averaged over ten different super-resolutions estimated from different realizations of the random image registrations uniformly drawn from the stated uncertainty range. The error bars do not overlap, showing that these differences are statistically significant.

3) Scalability and Computational Requirements: Table V compares the runtime for the SOLaR and SFT super-resolution methods. All the experiments reported in this paper were run on an Intel quad-core 3.20-GHz machine with 12 GB of memory. SFT [32] takes significantly longer, and scales poorly with respect to the reconstructed image size. The computation of the fast spherical transform requires $\mathcal{O}(B^2(\log(B))^2)$ operations, where B is typically chosen to correspond with the size of the high-resolution image that needs to be reconstructed. Furthermore, this operation needs to be repeated several times at each iteration of the gradient descent approach [33]. Due to this polynomial increase in computation time and memory constraints, we were not able to use SFT to reconstruct images larger than 128 × 128.

4) Effect on Visual Homing: Mobile robots capture images of their surroundings in order to visually home in on a target or return to a home location. omnidirectional cameras are better than conventional cameras for visual homing because they can capture the complete field of view. However, high-resolution omnidirectional cameras are expensive and the performance of visual homing has been shown to deteriorate at lower resolutions [29]. In this section, we compare the super-resolution methods in terms of a standard visual homing metric.

We report experimental results on the Bielefeld panoramic image database [43], which comprises omnidirectional images

TABLE V
RUNTIME-PER-SUPER-RESOLVED-IMAGE IN MINUTES FOR SOLAR AND SFT [32] FOR RECONSTRUCTING A 128×128 IMAGE FROM 16×16 IMAGES, AND A 256×256 IMAGE FROM 64×64 IMAGE. THE REGISTRATION UNCERTAINTY WAS 10°

	10 images	20 images	40 images	60 images	80 images
SFT (128×128)	13.41	24.88	40.82	49.36	70.85
SOLaR (128×128)	0.07	0.18	0.57	1.31	3.43
SOLaR (256×256)	0.16	0.43	1.17	2.53	5.64

TABLE VI
MEAN RATE OF RETURN TO HOME COMPARISON USING 128×128 IMAGES SUPER-RESOLVED FROM 16×16 SIZED IMAGES

# Images	15° reg. uncertainty		90° reg. uncertainty	
	SFT	SOLaR	SFT	SOLaR
2	0.12	0.10	0.03	0.09
5	0.22	0.42	0.04	0.38
10	0.50	0.78	0.07	0.66
20	0.85	0.91	0.09	0.79

captured by the ImagingSource DFK 4303 camera and a large wide-view hyperbolic mirror mounted on an ActivMedia Pioneer 3-DX robot [29]. The database consists of 170 omnidirectional images captured at 17×10 regularly-spaced locations on the floor (see Fig. 4 in [29]).

Two recent methods for visual homing with omnidirectional images are a differential visual-homing technique based on analyzing the optical flow [29], and an iterative approach based on local matched-filtering [44], [45]. The matched-filtering method performed better at the low signal-to-noise-ratios typical of low-cost imaging devices [44], [45], which is the scenario considered here, so we used it for these experiments.

The experimental setup is the following. The mobile robot is initialized randomly at a grid point, and given the "target" omnidirectional image at the unknown home location. Each time the robot moves to a new location it is given the omnidirectional images captured at that location and its immediately neighboring locations, which it correlates with the target image of its home location. The robot then moves to the location at which the scene has maximum correlation with the scene at the home location. Homing is considered successful if the robot reaches the home location in no more than 27 steps (because in 27 steps it could traverse the entire width and length of the room).

A standard metric for visual homing is the *mean return ratio*, which is the fraction of times the robot successfully locates home within 27 steps, averaged over uniformly-randomly chosen home locations and uniformly randomly chosen initial-placements of the robot.

Visual homing results are shown in Table VI for different numbers of low-resolution images and for two levels of registration uncertainty. Given a single 16×16 image, the visual homing return ratio is 0.05. Even with only two images and no information about their registration, SOLaR is able to double the return ratio to 0.09. In the best case considered, 20 such low-resolution images with registration known up to 15° were stitched together to produce a 91% chance of homing - on par with having originally captured a true 128×128 image. Even with no information about the relative registrations of the 20 images, SOLaR achieved a return ratio of 0.79. In general, the SFT method does not super-resolve adequately in the completely-unregistered case, whereas SOLaR provides useful gains. With the better-registered 15° uncertainty images, SOLaR can achieve higher return ratios with fewer images.

VI. CONCLUSION

Linearly interpolating a lattice of stored function values is a standard approach for the fast evaluation of a function. In this paper, we have shown that lattice values that minimize a regularized post-interpolation training error can be determined in closed-form. To ensure a unique and smooth solution, we proposed using a graph Hessian regularizer of second-order differences, but note that different applications may profit from more tailored regularizers. Large performance gains over the state of the art were shown for two applications, color management and omnidirectional super-resolution.

APPENDIX
SPECIFYING THE WEIGHT MATRIX W

Without loss of generality, assume that the domain has undergone an affine transformation such that the lattice sits at the origin with nodes at integer coordinates in $\mathbb{R}^d_+$. Further, let the d-dimensional vector $\tilde{m} = [\tilde{m}_1, \tilde{m}_2, \ldots, \tilde{m}_d]^T$ denote the number of nodes in each dimension (thus $m = \prod_{k=1}^{d} \tilde{m}_k$).

Interpolating a test point $x \in D$ requires the ability to index the nodes of the cell in which it is contained. To that end, define the function $c_j(x) : \mathbb{R}^d \to \mathbb{N}$ that returns the index of the jth vertex ($j = 1, \ldots, 2^d$) of the lattice cell that contains x. The function $c_j(x)$ can be computed as follows:

$$c_j(x) = 1 + \sum_{k=1}^{d} \left(\lfloor x[k] \rfloor + \alpha_j[k] \right) \left(\prod_{i=0}^{k-1} \tilde{m}_k \right) \qquad (16)$$

where $\tilde{m}_0 = 1$, $x[k]$ is the kth element of the vector x, and α_j is a vector whose components form the d-bit binary expression of j. For example, in Fig. 1, we have

$$c_1(x) = 1 + (\lfloor 1.8 \rfloor + 0)(1) + (\lfloor 1.4 \rfloor + 0)(3) = 5$$
$$c_2(x) = 1 + (\lfloor 1.8 \rfloor + 1)(1) + (\lfloor 1.4 \rfloor + 0)(3) = 6$$
$$c_3(x) = 1 + (\lfloor 1.8 \rfloor + 0)(1) + (\lfloor 1.4 \rfloor + 1)(3) = 8$$
$$c_4(x) = 1 + (\lfloor 1.8 \rfloor + 1)(1) + (\lfloor 1.4 \rfloor + 1)(3) = 9$$

which are indeed the indices of the four lattice nodes surrounding x.

Let $w_j(x)$ be the weight associated with the jth vertex $a_{c_j(x)}$ (for $j = 1, \ldots, 2^d$) of the lattice cell that contains x.

For d-linear interpolation, let $\Delta(x) = x - \lfloor x \rfloor$ with $\lfloor \cdot \rfloor$ performed component-wise. Then $w_j(x)$ can be computed as

$$w_j(x) = \prod_{k=1}^{d} \Big(\Delta(x)[k]\Big)^{\alpha_j[k]} \Big(1 - \Delta(x)[k]\Big)^{1-\alpha_j[k]}. \quad (17)$$

Note that the exponent acts like a selector of either the $\Delta(x)[k]$ or $(1 - \Delta(x)[k])$ term, automatically selecting whichever of the two is positive.

Let $W(x)$ be the $1 \times m$ sparse vector with kth element

$$W(x)[k] = \begin{cases} w_j(x), & \text{if } k = c_j(x), \text{ for } j = 1, \ldots, 2^d \\ 0, & \text{otherwise.} \end{cases} \quad (18)$$

For the $m \times 1$ matrix of lattice outputs b, the function value at x is interpolated as $W(x) \cdot b$. Likewise, given the matrix of n training vectors $\mathbf{X} = \big[x_1, \ldots, x_n\big]$, let $\mathbf{W}$ be the $n \times m$ matrix $\mathbf{W} = \big[W(x_1), \ldots, W(x_n)\big]$.

ACKNOWLEDGMENT

The authors would like to thank Z. Arican for helpful discussions about the spherical Fourier transform, and for sharing code and data.

REFERENCES

[1] R. Bala, "Device characterization," in *Digital Color Handbook*. Boca Raton, FL: CRC Press, 2003, ch. 5, pp. 269–384.

[2] H. S. Warren, Jr., *Hacker's Delight*. Boston, MA: Addison-Wesley, 2003.

[3] D. Wallner, "ICC profile processing models," in *Building ICC Profiles - the Mechanics and Engineering*. Reston, VA: ICC, 2000, ch. 4, pp. 150–167.

[4] E. K. Garcia, "Regression strategies for low-dimensional problems with application to color management," Ph.D. dissertation, Dept. Electr. Eng., Univ. Washington, Seattle, 2010.

[5] E. K. Garcia and M. R. Gupta, "Building accurate and smooth ICC profiles by lattice regression," in *Proc. 17th IS&T Color Imaging Conf.*, 2009, pp. 101–106.

[6] E. K. Garcia and M. R. Gupta, "Optimized construction of ICC profiles by lattice regression," in *Proc. 18th IS&T Color Imaging Conf.*, 2010, pp. 1–6.

[7] E. K. Garcia and M. R. Gupta, "Lattice regression," in *Advances in Neural Information Processing Systems*. Cambridge, MA: MIT Press, 2009, pp. 1–9.

[8] R. Bala and R. V. Klassen, "Efficient color transformation implementation," in *Digital Color Handbook*. Boca Raton, FL: CRC Press, 2003, ch. 11.

[9] M. R. Gupta, R. M. Gray, and R. A. Olshen, "Nonparametric supervised learning by linear interpolation with maximum entropy," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 28, no. 5, pp. 766–781, May 2006.

[10] W. R. Tobler, "Lattice tuning," *Geograph. Anal.*, vol. 11, no. 1, pp. 36–44, 1979.

[11] R. Bala, "Iterative technique for refining color correction look-up tables," U.S. Patent 5 649 072, Jul. 15, 1997.

[12] M. J. Baines, "Algorithms for optimal discontinuous piecewise linear and constant L_2 fits to continuous functions with adjustable nodes in one and two dimensions," *Math. Comput.*, vol. 62, no. 206, pp. 645–669, 1994.

[13] J. Z. Chan, J. P. Allebach, and C. A. Bouman, "Sequential linear interpolation of multidimensional functions," *IEEE Trans. Image Process.*, vol. 6, no. 9, pp. 1231–1245, Sep. 1997.

[14] J. Pittman and C. A. Murthy, "Fitting optimal piecewise linear functions using genetic algorithms," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 22, no. 7, pp. 701–718, Jul. 2000.

[15] V. Monga and R. Bala, "Sort-select damp: An efficient strategy for color look-up-table lattice design," in *Proc. Color Imaging Conf.*, 2008, pp. 247–253.

[16] I. Tastl, J. Recker, Y. Zhang, and G. Beretta, "An efficient high quality color transformation," in *Proc. Color Imaging Conf.*, 2009, pp. 111–116.

[17] V. Monga and R. Bala, "Algorithms for color look-up-table (LUT) design via joint optimization of node locations and output values," in *Proc. IEEE Int. Conf. Acoust. Speech Signal Process.*, Mar. 2010, pp. 998–1001.

[18] U. Luxburg, "A tutorial on spectral clustering," *Stat. Comput.*, vol. 17, no. 4, pp. 395–416, 2007.

[19] M. Belkin, P. Niyogi, and V. Sindhwani, "Manifold regularization: A geometric framework for learning from labeled and unlabeled examples," *J. Mach. Learn. Res.*, vol. 7, pp. 2399–2434, Dec. 2006.

[20] Y. Zhao and R. Bala, "Dynamic patch optimization for color printer characterization," in *Proc. Color Imaging Conf.*, 2011, pp. 201–204.

[21] M. R. Gupta, "An information theoretic approach to supervised learning," Ph.D. dissertation, Dept. Electr. Eng., Stanford Univ., Stanford, CA, 2003.

[22] P. J. Green and B. W. Silverman, *Nonparametric Regression and Generalized Linear Models: A Roughness Penalty Approach*. London, U.K.: Chapman & Hall, 1994.

[23] R. G. Keys, "Cubic convolution interpolation for digital image processing," *IEEE Trans. Acoust., Speech, Signal Process.*, vol. 29, no. 6, pp. 1153–1160, Dec. 1981.

[24] H. Kang, *Color Technology for Electronic Imaging Devices*. Bellingham, WA: SPIE, 1997.

[25] M. R. Gupta, E. K. Garcia, and E. M. Chin, "Adaptive local linear regression with application to printer color management," *IEEE Trans. Image Process.*, vol. 17, no. 6, pp. 936–945, Jun. 2008.

[26] N. Hrustemovic and M. R. Gupta, "Multiresolutional regularization of local linear regression over adaptive neighborhoods for color management," in *Proc. Int. Conf. Image Process.*, 2008, pp. 1–4.

[27] C. E. Rasmussen and C. K. I. Williams, *Gaussian Processes for Machine Learning* (Adaptive Computation and Machine Learning). Cambridge, MA: MIT Press, 2005.

[28] N. D. Jankovic and M. D. Naish, "Developing a modular active spherical vision system," in *Proc. IEEE Int. Conf. Robot. Autom.*, Apr. 2005, pp. 1234–1239.

[29] A. Vardy and R. Möller, "Biologically plausible visual homing methods based on optical flow techniques," *Connect. Sci.*, vol. 17, pp. 47–89, Mar. 2005.

[30] R. Orghidan, E. M. Mouaddib, and J. Salvi, "Omnidirectional depth computation from a single image," in *Proc. IEEE Int. Conf. Robot. Autom.*, Apr. 2005, pp. 1222–1227.

[31] Z. Arican and P. Frossard, "Super-resolution from unregistered omnidirectional images," in *Proc. Int. Conf. Pattern Recognit.*, 2008, pp. 1–4.

[32] Z. Arican and P. Frossard, "Joint registration and super-resolution with omnidirectional images," *IEEE Trans. Image Process.*, vol. 20, no. 11, pp. 3151–3162, Nov. 2011.

[33] Z. Arican and P. Frossard, "l_1 regularized super-resolution from unregistered omnidirectional images," in *Proc. Int. Conf. Acoust. Speech Signal Process.*, 2008, pp. 829–832.

[34] S. Park, M. Park, and M. Kang, "Super-resolution image reconstruction: A technical overview," *IEEE Signal Process. Mag.*, vol. 20, no. 3, pp. 21–36, May 2003.

[35] S. Farsiu, D. Robinson, M. Elad, and P. Milanfar, "Advances and challenges in super-resolution," *Int. J. Imaging Syst. Technol.*, vol. 14, no. 2, pp. 47–57, 2004.

[36] H. Nagahara, Y. Yagi, and M. Yachida, "Super-resolution from an omnidirectional image sequence," in *Proc. 26th Conf. IEEE Ind. Electron. Soc.*, Oct. 2000, pp. 2559–2564.

[37] H. Kawasaki, K. Ikeuchi, and M. Sakauchi, "Super-resolution omnidirectional camera images using spatio-temporal analysis," *Electron. Commun. Jpn. Part 3, Fundam. Electron. Sci.*, vol. 89, no. 6, pp. 47–59, 2006.

[38] Y. He, K. Yap, L. Chen, and L. Chau, "A nonlinear least square technique for simultaneous image registration and super-resolution," *IEEE Trans. Image Process.*, vol. 16, no. 11, pp. 2830–2841, Nov. 2007.

[39] P. Vandewalle, L. Sbaiz, J. Vandewalle, and M. Vetterli, "Super-resolution from unregistered and totally aliased signals using subspace methods," *IEEE Trans. Signal Process.*, vol. 55, no. 7, pp. 3687–3703, Jul. 2007.

[40] R. Mendes, J. Kennedy, and J. Neves, "The fully informed particle swarm: Simpler, maybe better," *IEEE Trans. Evolut. Comput.*, vol. 8, no. 3, pp. 204–210, Jun. 2004.
[41] R. Eberhart and J. Kennedy, "A new optimizer using particle swarm theory," in *Proc. 6th Int. Symp. Micro Mach. Human Sci.*, Oct. 1995, pp. 39–43.
[42] *Omnidirectional Images*. (2009) [Online]. Available: http://www.artstor.org/
[43] A. Vardy and R. Möller. (2005). *Panoramic Image Database* [Online]. Available: http://www.ti.uni-bielefeld.de/html/research/avardy/index.html
[44] R. Arora and H. Parthasarathy, "Navigation using a spherical camera," in *Proc. Int. Conf. Pattern Recognit.*, Dec. 2008, pp. 1–4.
[45] R. Arora, "Group theoretical methods in signal processing: Learning similarities, transformations and invariants," Ph.D. dissertation, Dept. Electr. Eng., Univ. Wisconsin-Madison, Madison, 2009.

Eric Garcia received the Bachelor's degree in computer engineering from Oregon State University, Corvallis, and the Ph.D. degree in electrical engineering from the University of Washington, Seattle, in 2010.

Raman Arora received the B.Eng. degree in electronics and communication engineering from the Netaji Subhas Institute of Technology, Delhi University, New Delhi, India, in 2001, and the M.S. and Ph.D. degrees in electrical engineering from the University of Wisconsin-Madison, Madison, in 2005 and 2009, respectively.

He was a Post-Doctoral Researcher with the University of Washington, Seattle, from 2009 to 2011. He is currently a Researcher with the Toyota Technological Institute at Chicago, Chicago, IL.

Maya R. Gupta (SM'04) received the Bachelor's degree in economics and electrical engineering from Rice University, Houston, TX, in 1997, and the Ph.D. degree in electrical engineering from Stanford University, Stanford, CA, in 2003.

She is an Associate Professor with the University of Washington, Seattle.