SINGLE IMAGE SUPERRESOLUTION BASED ON SUPPORT VECTOR REGRESSION

Karl S. Ni, Sanjeev Kumar, Nuno Vasconcelos, Truong Q. Nguyen

ECE Dept, UCSD, La Jolla, CA 92093-0407

Abstract - Support vector machine (SVM) regression is considered for a statistical method of single frame superresolution in both the spatial and Discrete Cosine Transform (DCT) domains. As opposed to current classification techniques, regression allows considerably more freedom in the determination of missing high-resolution information. In addition, since SVM regression approaches the superresolution problem as an estimation problem with a criterion of image correctness rather than visual acceptableness, its optimization results have better mean-squared error. With the addition of structure in the DCT coefficients, DCT domain image superresolution is further improved.

1. INTRODUCTION

Single frame superresolution, image upscaling, or image interpolation is the process by which a single low resolution image is expanded spatially to a higher resolution image. Along with the original information inherent in a low resolution image, superresolution requires additional information (i.e. new pixel values for new pixels) to contribute so that the missing information that is required to create the high-resolution image is provided. The process of determining the values of the missing information is the crux of our problem.

Simpler techniques for single frame superresolution such as bilinear and bicubic interpolation only consider low-resolution image information. The resulting upscaled image from these techniques is often blurry and remains at lower resolution. For this reason, statistical learning is preferred because additional information (other than the low-resolution image) will be brought to the table. The application of statistical learning to the superresolution problem can be termed image *estimation*.

Many image estimation implementations [1] use classification in order to achieve high-resolution counterparts. Whether the result is dynamic filtering or simple addition of samples from a training set, classification tends to "quantize" the output of the process when considering the number of possible outputs. This limits the capacity of the algorithm in such a way that the reconstructed image is only as precise as the number of classes included in the learning algorithm.

In an attempt to remedy the problems introduced by classification, a regression based approach concentrates on function estimation rather than discriminating between classes. The remainder of this work explores the application of Support Vector Machine regression to determine a regression for the relationship between known and unknown elements in the superresolution problem.

This work is supported in part by the CWC and matching funds from the U.C. Discovery Grant

2. A REVIEW AND EXPLANATION OF SUPPORT VECTOR REGRESSION

Support Vector Machines (SVM) is a learning algorithm ([2] [3]), with the ability to provide high-dimensional function estimation. Support Vector Regression (SVR), the use of SVMs for regression, operates in feature space to approximate unknown functions in output space, thereby using nonlinear functions to linearly estimate an unknown function.

For $y \in \Re$, $\forall\ i \in [1, N]$, we have two inequalities that bound the output points of the function to be estimated: one for the upper boundary and one for the lower boundary. Suppose that we are given a training set:

$$\Omega_r = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), ..., (\mathbf{x}_N, y_N)\} \tag{1}$$

where $\mathbf{x}_i \in \Re^n$ and $y_i \in \Re, \forall i \in [1, N]$, then it is possible to estimate the function $f : x \rightarrow y$ with the following optimization.

$$\min_{\mathbf{w},\xi^+,\xi^-,\rho,b} \frac{1}{2}\|\mathbf{w}\|^2 + C(\nu\rho + \frac{1}{n}\sum_{i=1}^{n}(\xi_i^+ + \xi_i^-))$$

subject to

$$\begin{aligned} &(\langle\phi(\mathbf{x}_i), \mathbf{w}\rangle + b) - y_i \leq \rho + \xi_i^+, \\ &y_i - (\langle\phi(\mathbf{x}_i), \mathbf{w}\rangle + b) \leq \rho + \xi_i^-, \\ &\rho, \xi_i^+, \xi_i^- \geq 0, \quad \text{for } i = 1, \ldots, n \end{aligned} \tag{2}$$

Slack variable vectors ξ_i^+ and ξ_i^- correspond to the upper and lower parameters in which the function $f(\mathbf{x}) = \mathbf{w}^T\phi(\mathbf{x}) + b$ is allowed to deviate for a prespecified error and cost. The function $\phi : \mathbf{x} \rightarrow \mathbf{z}$ maps the features $\mathbf{x}$ to a higher-dimensional space for greater flexibility. We can write the Lagrangian and express the dual problem to obtain a better representation shown in (3).

$$f(\mathbf{x}) = \sum_i(\alpha_i^+ - \alpha_i^-)K(\mathbf{x}, \mathbf{x}_i) + b \tag{3}$$

where $K(\cdot, \cdot) = \phi(\mathbf{x})^T\phi(\mathbf{x}) = kernel function$.

The Kernel function is a dictionary of functions in high dimensional space, and $\alpha^{+/-}$ values guide these functions to create the output $f(\mathbf{x})$ determined by the input $\mathbf{x}$. The $\alpha^{+/-}$ values (derived from the Karush-Kuhn-Tucker conditions) are values that are learned in the SVM regression and can be thought of as coefficients in feature space. Thus, given a feature $\mathbf{x}$, we will use the function $f(\mathbf{x})$ as defined by (3) to estimate a value as suggested by the training set (1).

3. SPATIAL DOMAIN SUPERRESOLUTION

Given an $N \times N$ patch from a low resolution image, the proposed algorithm predicts 4 pixels of a high resolution image corresponding to the center pixel of the $N \times N$ patch.

In the terminology of (1), (2), and (3), we need to learn the output label $\{y = f(\mathbf{x})\} \subset \mathbb{R}^4$ associated with the input $\{\mathbf{x} = vec(p)\} \subset \mathbb{R}^{N^2}$, where p is the $N \times N$ patch and $vec(.)$ denotes stacking all columns of a matrix as a single row.

This is a multiple output regression problem and in the literature it is often solved as separate single output regressions. Recently, there has been some work on learning vector valued function using operator-valued kernel [4], but these methods are not yet well understood. In this regard, we adopt the traditional method of treating a multiple output regression problem as separate single output regression problems for each output dimension.

The separated output regression problems for each output dimension is stated as learning the four outputs $\{y_i = f_i(\mathbf{x})\} \subset \mathbb{R}$ for $i = 1, \ldots, 4$, given the input $\mathbf{x} \in \mathbb{R}^{N^2}$.

We use ν-SVR [2] for learning the interpolation functions. Training data consists of sets of input-output pairs (x, y) obtained from high resolution test images and corresponding low resolution images.

The choice of kernel is especially important in this situation since as opposed to other interpolation methods, the algorithm doesn't have access to individual pixels except via kernel evaluation resulting in a scalar.

4. DCT DOMAIN SUPERRESOLUTION

Applications such as JPEG, MPEG, and H.26X typically transform raw images or video information into the DCT domain. The DCT's significant energy compaction alleviates the effects of estimation errors in the higher frequencies if weighting is correctly evaluated. Also, structural properties that make sense in the frequency domain can be utilized, where the spatial or time domain equivalent is very complex.

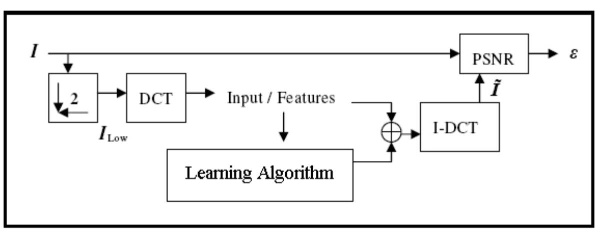


Fig. 1. The evaluation of a proposed algorithm's approximation of an image.

In Fig. 1, we have constructed an image from its corresponding downsampled version after odd and diagonal pixels are removed. To upscale the image, it is appropriate to observe how the image was downsampled in the first place. As we are working in the DCT domain, we will look at the decimation of a signal in that domain.

4.1. Decimation in Time

In image upscaling, we are typically trying to recover signal samples that have been removed by decimation. Decimation in time (DIT) by n retains every n^{th} sample of a signal. For decimation by two, n equals 2, and we retain all even samples while discarding all odd samples.

There are several types of DCTs: DCT-I, DCT-II, etc. Most signal compression algorithms rely on the DCT-II, to compress their signals. The definition of $X_N^{C2}(m)$, the N-point DCT-II, of a signal $x(n)$ is:

$$X_N^{C2}(m) = k_m \sum_{i=0}^{N-1} x(n) cos\left[\frac{(2n+1)m\pi}{2N}\right], \qquad m = 0, 1, ..., N-1. \tag{4}$$

As reported by Yip and Rao [5], we can write the N-point DCT as operations on combinations of even and odd samples.

By letting:

$$\begin{aligned} G(m) &= k_m \sum_{n=0}^{\frac{N}{2}} \{x(2n) + x(2n-1)\} cos\left[\frac{mn\pi}{\frac{N}{2}}\right] \\ H(m) &= k_m \sum_{n=0}^{\frac{N}{2}-1} \{x(2n) + x(2n+1)\} cos\left[\frac{(2n+1)m\pi}{N}\right] \end{aligned} \tag{5}$$

$$\Rightarrow X_N^{C2}(m) = k_m \left[\frac{G(m) + H(m)}{\cos\frac{mn\pi}{N}}\right], \qquad m = 0, 1, ..., \frac{N}{2} - 1. \tag{6}$$

$$\Rightarrow X_N^{C2} = k_m \left\{ \begin{array}{c} DCT\text{-}I(x(2n)) + DCT\text{-}I(x(2n-1)) \\ +DCT\text{-}II(x(2n)) + DCT\text{-}II(x(2n+1)) \end{array} \right\} \tag{7}$$

This final expression allows us to decompose the image that we wish to be estimated into what is known and what is unknown. Our known signal is $x(2n)$, the even samples, and our unknown signal is $x(2n-1)$ and $x(2n+1)$, the odd samples.

4.2. Application of DCT Properties to Two Dimensions

The equivalent formulation in two dimensions is not trivial. The amount of information that is lost from decimation is squared. From (7), half of the information comes from the even samples and half of the information from the unknown odd samples. In the 2-dimensional case (decimation in space), only a quarter of information will be known.

We use matrix representations to achieve corresponding expressions analogous to those seen in Sec. 4.1. Defining C_1 as the $\frac{N}{2}$-point DCT-I matrix and C_2 as the $\frac{N}{2}$-point DCT-II matrix, we express their forms as:

$$C_1 = k_m \begin{bmatrix} 1 & 1 & 1 & \cdots \\ 1 & cos(\frac{\pi}{\frac{N}{2}}) & cos(\frac{2\pi}{\frac{N}{2}}) & \cdots \\ 1 & cos(\frac{2\pi}{\frac{N}{2}}) & cos(\frac{4\pi}{N}) & \cdots \\ \vdots & \vdots & \vdots & \ddots \end{bmatrix} \tag{8}$$

and

$$C_2 = k_m \begin{bmatrix} 1 & cos(\frac{1}{2}\frac{\pi}{N}) & cos(1\frac{\pi}{N}) & \cdots \\ 1 & cos(\frac{3}{2}\frac{\pi}{N}) & cos(3\frac{\pi}{N}) & \cdots \\ 1 & cos(\frac{5}{2}\frac{\pi}{N}) & cos(5\frac{\pi}{N}) & \cdots \\ \vdots & \vdots & \vdots & \ddots \end{bmatrix} \tag{9}$$

From (5), it is necessary to take the DCT-I from $n = 0$ to $\frac{N}{2}$, yielding $\left(\frac{N}{2} + 1\right)$ evaluation points rather than $\left(\frac{N}{2}\right)$ evaluation points. For this reason, C_1 has an extra row over C_2 and any input matrix must be padded with an extra column of zeros to accommodate for the matrix size.

To apply the DIT algorithm in two dimensions, it is necessary to partition the two dimensional signal as has been done in the one-dimensional case. In one dimension, the signal is partitioned into even and odd portions. In two dimensions, decimation is done in two directions, meaning there are four distinct partitions. The high resolution image or two-dimensional signal, **x**, is thus decomposed into four parts: $\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3,$ and $\mathbf{x}_4$, and these are organized as in (10).

$$\mathbf{x} = \left[\begin{array}{cc|cc|c} x_1(0,0) & x_2(0,0) & x_1(1,0) & x_2(1,0) & \\ x_4(0,0) & x_3(0,0) & x_4(1,0) & x_3(1,0) & \cdots \\ \hline x_1(0,1) & x_2(0,1) & x_1(1,1) & x_2(1,1) & \\ x_4(0,1) & x_3(0,1) & x_4(1,1) & x_3(1,1) & \cdots \\ \hline x_1(0,2) & x_2(0,2) & x_1(1,2) & x_2(1,2) & \\ x_4(0,2) & x_3(0,2) & x_4(1,2) & x_3(1,2) & \cdots \\ \hline & \vdots & & \vdots & \ddots \end{array}\right] \tag{10}$$

Decimation in space (DIS) yields only one in four pixel values, and each matrix $\mathbf{x}_i = [x_i(m,n)]$ represents a grouping of possible remaining pixel combinations after conventional decimation by two in each direction. We designate $\mathbf{x}_1$ to be the remaining matrix left after decimation.

Because the cosine transform is paraunitary, the 2-D DCT-II transform of a matrix **x** is $\hat{C}_2^T \mathbf{x} \hat{C}_2$. (The matrix $\hat{C}_2$ supports an $N \times N$ DCT-II as opposed to C_2, which supports an $\frac{N}{2} \times \frac{N}{2}$ DCT-II). This can also be written $\{(\mathbf{x}\hat{C}_2)^T \hat{C}_2\}^T$, where essentially, the transform has been taken in one direction and then another.

Applying DIT to a two-dimensional signal in one direction (say, the horizontally) requires four matrix multiplications and results in four terms (7). Applying the DIT in the remaining direction (vertically) decimates each of these four terms into four smaller terms, resulting in sixteen total terms.

The result, $\hat{C}_2^T \mathbf{x} \hat{C}_2$, can be grouped into the four terms relating to $\mathbf{x}_i$, where $i = 1, 2, 3,$ and 4. Each of these four terms is unique, and we denote these matrices as X_1, X_2, X_3, and X_4, ((11) through (14)).

$$X_1 = C_1^T \begin{pmatrix} \mathbf{x}_1 & \mathbf{0} \\ \mathbf{0}^T & \end{pmatrix} C_1 + C_1^T \begin{pmatrix} \mathbf{x}_1 \\ \mathbf{0}^T \end{pmatrix} C_2 + C_2^T \begin{pmatrix} \mathbf{x}_1 & \mathbf{0} \end{pmatrix} C_1 + C_2^T \mathbf{x}_1 C_2 \tag{11}$$

$$X_2 = C_1^T \begin{pmatrix} \mathbf{0} & \mathbf{x}_2 \\ & \mathbf{0}^T \end{pmatrix} C_1 + C_1^T \begin{pmatrix} \mathbf{x}_2 \\ \mathbf{0}^T \end{pmatrix} C_2 + C_2^T \begin{pmatrix} \mathbf{0} & \mathbf{x}_2 \end{pmatrix} C_1 + C_2^T \mathbf{x}_2 C_2 \tag{12}$$

$$X_3 = C_1^T \begin{pmatrix} \mathbf{0} & \mathbf{0}^T \\ & \mathbf{x}_3 \end{pmatrix} C_1 + C_1^T \begin{pmatrix} \mathbf{0}^T \\ \mathbf{x}_3 \end{pmatrix} C_2 + C_2^T \begin{pmatrix} \mathbf{0} & \mathbf{x}_3 \end{pmatrix} C_1 + C_2^T \mathbf{x}_1 C_2 \tag{13}$$

$$X_4 = C_1^T \begin{pmatrix} \mathbf{0}^T & \mathbf{0} \\ \mathbf{x}_4 & \end{pmatrix} C_1 + C_1^T \begin{pmatrix} \mathbf{0}^T \\ \mathbf{x}_4 \end{pmatrix} C_2 + C_2^T \begin{pmatrix} \mathbf{x}_4 & \mathbf{0} \end{pmatrix} C_1 + C_2^T \mathbf{x}_4 C_2 \tag{14}$$

The final result is shown in (15).

$$X_{N\times N}^{C2}(l,m) = \hat{C}_2^T \, \mathbf{x} \, \hat{C}_2 = k_l^T \cdot k_m \cdot \sum_{i=1}^{4} X_i, \tag{15}$$

for $l, m = 0, 1, \ldots \frac{N}{2}$. Because **x** was chosen as the remaining matrix after decimation, the terms in the DIT equation as given in (15) can only be exactly determined for $i = 1$.

4.3. DCT Domain Support Vector Regression

Using the derivations in Sec. 4.2, the super-resolution problem has the goal of recovering the DCT-II of a high-resolution image by assuming an observed DCT-II of the decimated version of it.

That is to say that, in the terminology of Sec. 4.2, given

$$\text{DCT-II } (\mathbf{x}_1) = k_l^T k_m \left(C_2^T \ \mathbf{x}_1 \ C_2, \right)$$

we wish to solve for

$$X_{N\times N}^{C2}(l,m) = k_l^T k_m \left(\hat{C}_2 \ \mathbf{x} \ \hat{C}_2 \right), \tag{16}$$

where **x** is twice as large in each direction or has twice the resolution of $\mathbf{x}_1$.

This is exactly the inverse DIS problem, where $C_2^T \mathbf{x}_1 C_2$ exists in X_1 from (11), and the remaining fifteen terms compose the recovery problem of (15). We proceed to estimate $\mathbf{x}_i$, for $i = 2, 3,$ and 4 with only the knowledge of $C_2^T \mathbf{x}_1 C_2$.

From (11), we know that all the terms in X_1 can be exactly determined mathematically. This takes care of four of the sixteen total terms that are required for the construction of $X_{N\times N}^{C2}(l,m)$, and we need to estimate X_2, X_3, and X_4 in (15).

It is here that we can apply our SVR learning algorithm, where the input to the algorithm is the DCT-II of $\mathbf{x}_1$, or $C_2^T \mathbf{x}_1 C_2$. Thus, our features are the set of DCT-II's of possible $\mathbf{x}_1$'s, and our ultimate goal is achieving the corresponding DCT-II of **x**, the frequency representation of the high resolution image.

Now, we can restate our problem as follows. Given $C_2^T \mathbf{x}_1 C_2$, can we find $X_{N\times N}^{C2}(l,m) = \hat{C}_2^T \mathbf{x} \hat{C}_2$, by using mathematical operations and SVR to estimate the rest of the terms given in Sec. 4.2?

As stated before, (11) allows us to exactly determine the terms in (11) from $C_2^T \mathbf{x}_1 C_2$. The sum of the rest of the terms (when $i \neq 1$) are estimated by SVR, and thus the SVR synthesis algorithm is completely defined. The solution is

$$\begin{aligned} X_N^{C2}(l,m) &= k_l^T k_m \left\{ \chi + g(\chi) + SVR(\chi) \right\} \\ &= k_l^T k_m \left\{ \chi + g(\chi) + \sum_i (\alpha_i^+ - \alpha_i^-) K(\chi, \chi_i) + b \right\} \end{aligned} \tag{17}$$

where the terms $\chi = C_2^T \mathbf{x}_1 C_2$ and $g(\chi) = C_2^T \mathbf{x}_1 C_1 + C_1^T \mathbf{x}_1 C_2 + C_1^T \mathbf{x}_1 C_1$ are exactly known, and $SVR(\chi)$ estimates the remaining terms shown in (18).

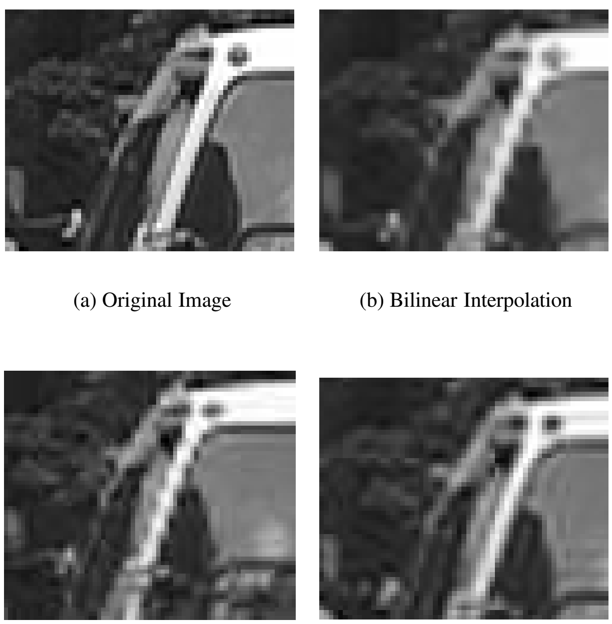
$$SVR(\chi) \to \sum_{i=2}^{4} X_i \tag{18}$$

5. EXPERIMENTAL RESULTS

All algorithms were simulated in Matlab using the libsvm library [2]. Different parameter values were tried with Gaussian and polynomial kernels, but due to the better performance of the Gaussian kernel, subsequent experiments used only the Gaussian Kernel. The bandwidth parameter of the Gaussian kernel was selected based on five-fold cross-validation. All training and reconstructed images were selected from various MPEG standard QCIF and CIF sized video sequences.

For realistic applications, a large and generic training set should be used to accommodate for any testing image. For ease of simulation and time, our training sets were small and representative of the specific testing image. In the spatial domain, this training set consisted of a single frame similar to the testing frame in which every 5×5 patch (overlapping allowed) in a QCIF image was paired with the four corresponding high-resolution pixels in the CIF image. In the frequency domain, this training set consisted of nine frames similar to the testing frame in which every 8×8 DCT block (overlapping *not* allowed) in the QCIF images was paired with the corresponding high-resolution summation values in 18. The results are shown in Fig. 2.

For comparison purposes, Matlab's imresize command was used to upscale the QCIF images to CIF images using bilinear and bicubic interpolation. We also implemented the algorithm proposed in [6] and adapted it for its use in image upscaling, but results were not promising enough for natural images. The higher order spectra estimated by their algorithm seems to suffer from overfitting, which is not surprising considering that there is no regularization involved.



(a) Original Image (b) Bilinear Interpolation

(c) Spatial Domain SVR (d) Frequency Domain SVR

Fig. 2. Reconstructed CIF frame 10 of "Bus" sequence. Zoomed in to show the effect of proposed algorithm.

The resulting PSNR values for the "Bus" sequence in Fig. 2 are as follows. As to be expected, bilinear interpolation in Fig. 2(b) has the lowest PSNR (23.301 dB), because it doesn't add any resolution, but rather averages out the observed information. Bicubic (not shown) gives 23.209 dB. What is gained in spatial SVR is sharpness, and we can see that in Fig. 2(c). The PSNR value for spatial domain SVR is 25.995 dB. This sharpness also remains in the frequency domain, however with the additional knowledge that the high-resolution DCT is in the form of (18), we are able to add some structure to our solution and thereby bring more information to the table. This is prevalent in Fig. 2(d), and has PSNR value of 26.843 dB.

Fig. 2 can be seen as a sample of our results. Several other scenarios with variably sized training sets and reconstruction frame offsets were tested to varying degrees of success. Obviously for smaller sample training sets, the quality of the proposed algorithm in both domains degrades. In addition, when blocks in the training sets and testing sets are similar except for a translational shift that is not an integral multiple of block size, then the block-based DCT coefficients fail to capture this similarity. This can be alleviated to some extent by using overlapping shifts of the same block area in the formation of the training set, thereby capturing any translational shift that may occur.

For additional results, images, and tables, please see consult research pages on UCSD's videoprocessing website at

`http://videoprocessing.ucsd.edu/research/~karl/svr_sr.html`.

6. CONCLUSIONS AND FUTURE WORK

We have proposed a single image superresolution algorithm based on support vector regression. This algorithm utilizes information learned from training data in addition to observed information, and results in better PSNR than traditional methods.

There are a number of future projects that can be considered with this work. In our experiments, the Gaussian kernel was inadequate near regions with strong edges, underscoring the necessity of constructing kernels which are better suited for the needs of a particular application domain. (Recently proposed kernel learning algorithms [7] [8] should prove useful.) Another interesting recent development is the learning of vectored functions using operator-valued kernels. Apart from exhibiting a desirable attribute of unified treatment of different components of input, such kernels should be able to extract more information about the training data than a scalar-valued kernel. Also, from an applications standpoint, the present algorithm can be extended to reconstruction or denoising problems. Finally, computational complexity of SVM-based learning algorithms makes it essentially unusable for applications with limited resources. A combination of aforementioned improvements can results in learning machines which make efficient use of limited resources.

7. REFERENCES

[1] W. Freeman, T. Jones, and E. Pasztor, "Example based super-resolution," *IEEE Computer Graphics and Application*, March/April 2002.

[2] Chih-Chung Chang and Chih-Jen Lin, *LIBSVM: a library for support vector machines*, 2001, Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

[3] V. Vapnik, *Statistical Learning Theory*, John Wiley & Sons Inc., 1998.

[4] C. A. Micchelli and M. Pontil, "On learning vector-valued functions," *Neural Computation*, vol. 17, 2005.

[5] K. R. Rao and P. Yip, *Discrete Cosine Transforms: Algorithms, Advantages, Applications*, Academic Press, Inc., 1990.

[6] M. O. Franz and Bernhard Scholkopf, "Implicit wiener series for higher-order image analysis," in *Advances in Neural Information Processing Systems*, 2004.

[7] Gert R. G. Lanckriet, Nello Cristianini, Peter Bartlett, Laurent El Ghaoui, and Michael I. Jordan, "Learning the kernel matrix with semidefinite programming," *J. Mach. Learn. Res.*, vol. 5, pp. 27–72, 2004.

[8] C. S. Ong and A. J. Smola, "Machine learning using hyperkernels," in *International Conference on Machine Learning*, 2003.