highly commended student paper award

Lossless Image Compression based on Kernel Least Mean Squares

Ruben Verhack[⋆†], Lieven Lange[†], Peter Lambert[⋆], Rik Van de Walle[⋆], and Thomas Sikora[†]
[⋆]Ghent University - iMinds - Multimedia Lab, Ghent, Belgium
[†]Technische Universität Berlin - Communication Systems Lab, Berlin, Germany

***Abstract*—This paper introduces a novel approach for coding luminance images using kernel-based adaptive filtering and context-adaptive arithmetic coding. This approach tackles the problem that is present in current image and video coders; these coders depend on assumptions of the image and are constrained by the linearity of their predictors. The efficacy of the predictors determines the compression gain.**

The goal is to create a generic image coder that learns and adapts to the characteristics of the signals and handles non-linearity in the prediction. Results show that pixel luminance prediction using the *Kernel Least Mean Squares* (KLMS) yields a significant gain compared to the standard Least Mean Squares algorithm. By coding the residual using a *Context-Adaptive Arithmetic Coder* (CAAC), the codec is able to outperform the current industry standards of lossless image coding. An average bitrate reduction of more than 2.5% is found for the used test set.

I. Introduction

Linear predictive coding is a powerful technique and well-established method for coding of audio and video samples. The strategy involves a linear prediction filter and the coding of prediction error amplitudes $e(n)$ rather than the amplitudes of the original signal $x(n)$. The purpose of the prediction filter is to arrive at a prediction error variance σ_e^2, that is smaller than the variance σ_x^2 of the original signal, resulting in the prediction gain $G_P = \frac{\sigma_x^2}{\sigma_e^2}$.

Optimal prediction filters in the "Wiener" sense also result in a decorrelated signal $e(n)$. The prediction filters thus serve as pre-whitening filters prior to coding the prediction error samples. Coding theory reveals that a prediction gain larger than 1 usually results into a coding gain, thus coding rate becomes $R_e < R_x$.

A linear predictor is, however, only optimal as long as the source is stationary and Gaussian distributed. Unfortunately, audio and video samples in practice resemble amplitude density distributions that are far from Gaussian - thus linear filters are deemed to provide suboptimal results for prediction and coding. Surprisingly, few attempts have been made thus far to explore non-linear prediction strategies for coding of audio and video.

During the last 20 years, powerful non-linear techniques evolved in the machine learning community, among others, around the non-linear support vector machine theory for classification and regression of signals [1]. The immense success of non-linear support vector classification in general is due to the use of a non-linear transform $\phi(x)$ of the signals/data into a (possibly infinite) dimensional feature space. Non-linear regression using kernels is intimately related to the problem of non-linear prediction in signal theory, and thus highly relevant for predictive coding in general.

Our challenge is the exploration of non-linear statistical dependencies between amplitudes of image samples for reducing redundancy of the source prior to coding. To this end we investigate the feasibility of kernel filters for non-linear prediction and lossless coding of images. In other words, our goal is to create a generic image coder that adapts to the characteristics of the signal and handles the non-linearity in the prediction step. As such, we tackle some limitations that are still present in other image coders.

Standard image coders depend on a predefined model of the image statistics. In JPEG-LS/LOCO-I and HEVC intracoding, linear predictors are used to predict pixel values. These basic models try to capture how the signal is most likely to behave. In JPEG-2000, the images are built as linear combinations of wavelets, and in JPEG linear combinations of DCT-blocks are used [2]. These codecs are based on the assumption that these bases are the basic building blocks of an image [3]. There are few approaches that incorporate non-linear prediction. One example is AdNN$_+$, which is built as an ensemble of three adaptive neural networks. AdNN$_+$ is a parametric method, where ours is non-parametric or *memory-based* [4].

The models that are used in the current state of the art coding techniques, excluding the high complexity coders such as TMW^{LEGO} and AdNN$_+$, are very basic, but perform very fast and are already very efficient [5]. These lower complexity coders share the same characteristics, i.e. linearity and basic assumptions of the image. Our goal is to extend these to non-linearity and having no or minimal assumptions of the content of an image.

Our method for tackling this is to use the above mentioned *kernel methods*. These methods have recently gained popularity in signal processing and have led to a new transform coding scheme and an adaptive sampling image coding approach [6][7][8]. These methods allow to create non-linear filters and to learn the characteristics of a signal. The idea is not to use a predefined set of predictors or bases, but to build these generically by learning from the images themselves. Furthermore, due to the use of kernel methods, the predictors can function non-linearly while maintaining a reasonable complexity, depending on the size of the dataset.

A generic method makes the coder suitable for a wide range of content, or a mixture of content. For example, JPEG-LS generally yields good results, although JPEG-2000 in lossless mode often outperforms JPEG-LS. This is especially true on images with very high spatial frequencies, for which the simple assumptions in JPEG-LS do not hold. Otherwise, the Portable Network Graphics (*PNG*) format clearly outperforms JPEG-LS and JPEG-2000 for specific computer generated images, such as line drawings [9].

Our coder first performs a prediction using an kernel based adaptive filter, which results in a decorrelated image, the *residual*. This residual is then later entropy coded using our *Context-Adaptive Arithmetic Coder* (CAAC). Due to this structure, the presented encoder and decoder are identical for the greatest part of the code. Therefore, the complexity of the encoder and the decoder is identical.

Since the use of kernel filters is not well introduced in the coding community, we start with an overview on the essentials of kernel filters to motivate the subject in Section II. Next, we will discuss the techniques used for prediction and the entropy coder in Section III. The experiments are discussed in Section IV, followed by the conclusions in Section V.

II. Kernel Filters

Assume signal $x(n)$ is a collection of pixel amplitudes in a scan line of a grey-level image, n being the position of a pixel in horizontal dimension. Our purpose is to predict a pixel amplitude $x(n)$ based on previously L coded/transmitted amplitudes $x(n-1), x(n-2), \ldots, x(n-L)$.

A linear predictor is of the form

$$x_p(n) = \mathbf{w}^T \mathbf{x} \tag{1}$$

with previous sample vector $\mathbf{x}$ and weight vector $\mathbf{w}$ with L filter parameters. The optimal prediction filter parameters can be found by employing the orthogonality principle according to the *Wiener-Hopf* equation.

A non-linear kernel predictor is based on the following non-linear expansion of the previous samples vector $\mathbf{x}$

$$x_p(n) = \boldsymbol{\omega}^T \boldsymbol{\phi}(\mathbf{x}) \tag{2}$$

Here, $\boldsymbol{\phi}(\boldsymbol{x})$ is a feature vector containing the non-linear expansion terms of $\mathbf{x}$ and $\boldsymbol{\omega}^T$ the weight vector. Please note, that usually both $\mathbf{x}$ and $\boldsymbol{\omega}^T$ are very high dimensional, and even of infinite dimension for the Gaussian expansion.

Depending on the exact non-linear feature mapping the filter design would involve the optimization of a possibly infinite dimensional weight vector, which is not feasible. Therefore, the following extension to Eq. 2 allows to arrive at a finite dimensional filter design,

$$\boldsymbol{\omega}^T = \sum_{k=1}^{J} w_k \boldsymbol{\phi}(\mathbf{c}_{\boldsymbol{k}}) \tag{3}$$

where $\boldsymbol{\phi}(\boldsymbol{c_k})$ is now non-linear expansion on each of the J centers $\boldsymbol{c_k}$. J being a finite number of, what can be thought of, reference points $\mathbf{c}_k$ in the feature space.

The predictor equation (Eq. 2) thus extends into

$$x_p(n) = \sum_{k=1}^{J} w_k \boldsymbol{\phi}(\mathbf{c}_k)^T \boldsymbol{\phi}(\mathbf{x}) \tag{4}$$

We have derived a filter that is non-linear in $\mathbf{x}$ but fortunately linear in $\mathbf{w}$. If $J \ll \dim\{\boldsymbol{\phi}(\mathbf{x})\}$ then we also have reduced the number of filter parameters significantly. Nevertheless, the predictor filter still involves the calculation of an inner vector product whereby the dimension of the vectors can be infinite.

According to the Mercer theorem [10], there exist positive, semi-definite kernel functions $\kappa_h(\cdot,\cdot)$ whereby

$$\kappa_h(\mathbf{c}_k, \mathbf{x}) = \boldsymbol{\phi}(\mathbf{c}_k)^T \boldsymbol{\phi}(\mathbf{x}) = <\boldsymbol{\phi}(\mathbf{c}_k), \boldsymbol{\phi}(\mathbf{x})> \tag{5}$$

As a consequence, it is possible to replace the inner products in our predictor equation with a well defined Mercer kernel function (the Mercer kernel) and we do not need to calculate inner products with vector of infinite dimensions. In other words, the kernel function $\kappa(\cdot,\cdot)$ of the vectors in the lower dimensional space (the *input space*), is the inner-product of the vectors mapped to the higher dimensional space (the *feature space*). This mapping is also frequently referred to as the *kernel trick*.

This enables us to work in the feature space without an excessive increase of complexity. In simpler terms, this results in a non-linear comparison metric between patterns.

Consequently, the output of our non-linear predictor filter is expressed as the weighted sum of kernel functions:

$$x_p(n) = \sum_{k=1}^{J} w_k \kappa_h(\mathbf{c}_k, \mathbf{x}) \tag{6}$$

Here, each input past samples vector $\mathbf{x}$ is correlated against each center vector $\mathbf{c}_k$ using the kernel function $\kappa_h(\cdot,\cdot)$. The predicted value of $x(n)$ is the weighted sum of all J kernel evaluations. The non-linearity of the filter is embedded into the choice of the particular kernel function.

III. Lossless coding

A. Prediction

A common approach used in lossless coding is to reduce correlation in an image by *online prediction*, in which the image is processed in raster scan. As such, previously encoded neighbouring pixels (the *neighbourhood*) are used to estimate the value of the current pixel. Thus, only the prediction error needs to be saved, since the prediction itself can be restored at the decoder side. The collection of all prediction errors is called the *residual*. Note that we always work with a mean subtracted image.

Our challenge is to design a generic, self-adapting algorithm that does not rely on dedicated and fine-tuned switching operations (as in JPEG-LS and related standards). We want to *learn* the relationship between the pixel and its neighbourhood. Since our strategy involves the continuous prediction of samples $x(n)$ based on an ever-growing dataset (the previous coded pixels), it was advisable to resort to *online* kernel

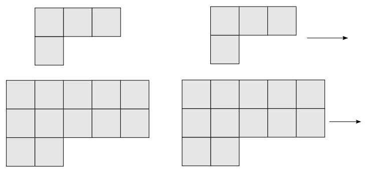
Fig. 1. An example of the neighbourhood of a pixel. Different masks are possible.

```
% Choose step-size parameter $\eta$ and kernel $\kappa$
$\mathbf{a}_1(1) = \eta d(1)$
$f_1 = \mathbf{a}_1(1)\kappa(\mathbf{u}(1), \cdot)$

% Computation
while $[\mathbf{u}(i), d(i)]$ available do
    % Predict output using estimated function $f_{i-1}$
    $f_{i-1}(\mathbf{u}(i)) = \sum_{j=1}^{i-1} \mathbf{a}_i(i-1)\kappa(\mathbf{u}(i), \mathbf{u}(j))$
    % Calculate the error made
    $e(i) = d(i) - f_{i-1}(\mathbf{u}(i))$
    % Compute and store the new filter coefficient
    $\mathbf{a}_i(i) = \eta e(i)$
end while
```

Fig. 2. The kernel least-mean-square algorithm [10]

algorithms in order to avoid the inversion of very large matrices.

A linear way of doing this is using the *Least Mean Squares* algorithm. However, these relations do not have to be linear. For example, linear predictors have difficulties learning strong edges. For this reason, the LMS algorithm can be extended using kernels, resulting in the *Kernel Least Mean Squares* (KLMS) [11], which is able to learn non-linear relations between the neighbourhoods and their respective pixel, as seen in Fig. 1. KLMS, which is shown in Fig. 2, is an algorithm particularly known in the machine learning world and universally applied for various time-series prediction problems.

The predictor parameters are adapted in an iterative way as the filter moves over the pixels. Intuitively, the kernel function $\kappa(\mathbf{u}_i, \cdot)$ is used as a similarity measure to find the distances between the current pattern $\mathbf{u}_i$ and all the patterns in the training set. These similarities are then used to weight the filter coefficients $\mathbf{a}_{i-1}$ of the filter. This weighted sum gives the prediction $f_{i-1}(\mathbf{u}_1)$ for the pattern $\mathbf{u}_i$ (with corresponding target output d_i). The error that was made $e = d_i - f_{i-1}(\mathbf{u}_1)$ is used to compute a new filter coefficient a_i using η, which is the step-size parameter. The step-size parameter is the compromise between convergence time and maladjustment [10]. Finally, the pattern $\mathbf{u}_i$ is stored in the training set.

The problem with KLMS is that the complexity grows as the dataset grows. In this initial research, every new pattern encountered during the encoding is is considered as a center and is thus stored in the dataset, i.e. J is the amount of samples. Implementation with a fixed-budget method such as the *Kernel Recursive Least-Squares Tracker* [12], is still ongoing research.

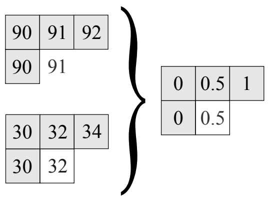

Fig. 3. Transform the patterns from absolute values towards relative values

We have adapted the KLMS algorithm to be better suited for the prediction of images in the following ways.

1) Relative Space: Instead of using the absolute luminance values, we normalize the values in the neighbourhood (Eq. 7).

$$x_i = \frac{x_i - \min(\mathbf{x})}{\max(\mathbf{x}) - \min(\mathbf{x})}, \forall x_i \in \mathbf{x} \tag{7}$$

This allows multiple patterns in the absolute luminance space to be projected on a single pattern in the relative space, as illustrated in Fig. 3. In other words, we compare relative patterns instead of exact matches. As such, patterns learned in a dark area of the image are also relevant in patterns in light areas if the changes in the environment are uniform.

This increases the prediction capability of the algorithm, but decreases the resilience towards contradicting patterns. However, experiments have shown that the prediction gain is higher than the loss through contradicting patterns.

2) Choice of kernel: Experiments have shown that the rapid descend of the Laplace kernel improves the prediction capability of the algorithm.

$$\kappa_L(\mathbf{u}, \mathbf{v}) = \exp(-\frac{||\mathbf{u} - \mathbf{v}||}{\sigma}) \tag{8}$$

3) Adaptive Bandwidth: Bandwidth or *kernel size* estimation is a well known problem in using kernel-based techniques. If the bandwidth is too large, then the system would reduce to linear regression. If the bandwidth is too small, then the system cannot do inference on unseen samples that fall between the training data points [10].

To make the algorithm more adaptive towards different types of images, we have used an *adaptive bandwidth* (AB) approach, first introduced by Takeda et al. [13]. The bandwidth adapts towards the *local density* of the patterns in the input space. The local density μ_i is calculated as

$$\mu_i = \left\{ \frac{\hat{d}(\mathbf{x}_i)}{\exp\left(\frac{1}{P}\sum_{i=1}^{P} \log(\hat{d}(\mathbf{x}_i))\right)} \right\}^{-\alpha}. \tag{9}$$

Where P is the number of elements in our training set, the *sample density*, $\hat{d}(\mathbf{x})$, is measured as

$$\hat{d}(\mathbf{x}) = \frac{1}{P}\sum_{i=1}^{P} \kappa(\mathbf{x}_i, \mathbf{x}), \tag{10}$$

with α being the *density sensitivity* parameter. α is a scalar satisfying $0 < \alpha \leq 1$. This results in the fact that μ_i has an expected value of 1. μ_i is then multiplied with the bandwidth parameter of the kernel function, σ in the Laplacian case. This expands the bandwidth when the local density is lower than the mean, or shrinks the bandwidth in higher density areas.

This approach increases the robustness, i.e. if the bandwidth parameter is not optimal for the image, the bandwidth can tune itself to better fit the particular data set. The impact is more noticeable when using slow decaying kernel functions such as the Gaussian kernel, instead of the Laplacian kernel.

B. Entropy coding

Arithmetic coders are well known entropy coders with the ability to reach a bitrate close to the first order entropy of the source. The idea is based on an interval nesting algorithm depending on the symbol probabilities [14][15].

Our implementation, dubbed *Context-Adaptive Arithmetic Coder* (CAAC), receives the normalized histogram of the residual image and fits a Laplacian shaped probability mass function via maximum likelihood estimation. Consequently, the only costs for signalling the initial probabilities consist of the mean and the variance. The probability mass function updates after each symbol, in order to make the coding context-adaptive. Because it is the histogram of the values to be yet encoded, the value after the occurrence of a symbol decreases by one at each step.

From our experiments, CAAC results in a bitstream that has a bit-per-pixel ratio that is not more than 0.01 bpp more than the entropy of the residual.

IV. Experiments

A. Implementation

The coder is implemented as a combination of MATLAB, C and NVIDIA CUDA C. Some parts of KLMS are very well suited for parallelism, such as the kernel evaluations that are done in every iteration between the current neighbourhood and the neighbourhoods of all previous patterns. Parallelizing this on the GPU made the algorithm approximately 90 times faster for a 512-by-512 pixel image.

B. Mask

Experiments were performed with different types of masks, all masks have their advantages and disadvantages towards types of patterns they are able to capture. The larger the mask, the more distinct the patterns are, but lowers the prediction capability for new unseen patterns. In contrast, the smaller the masks are, the more problems are present with instationarity. That is, the patterns are small in size, they overgeneralise, which leads to contradictions in our data set which leads to maladjustments of the weights and lowers the convergence speed. Our experiments have shown that for images with sufficient high resolution, a square mask having 12 elements yields the best results.

C. Test set

For our experiments, continuous-tone grayscale images of different resolutions (256x256, 512x512, 720x576) were selected. These test sets are widely used for comparing performance of lossless image coding [16].

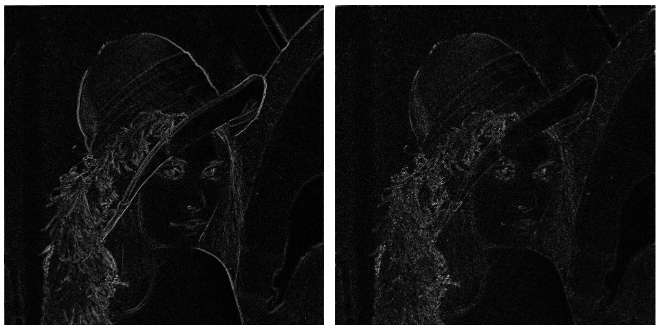
(a) LMS (b) KLMS

Fig. 4. Comparison between the absoluted residuals for *Lena*.

D. Comparing LMS and KLMS

Here, we compare the KLMS algorithm with the online *Least Mean Square* (LMS) algorithm. The efficacy of prediction is measured through calculating the entropy of the residual. In both cases, the mean value was subtracted from the image. Consequently, the benefit of the non-linearity of the method is shown. Both algorithms were trained on the image they were coding in order to find the best possible parameters. For KLMS, the parameters are the learning step, bandwidth, and the mask size. The same parameters, except for the bandwidth, are present in LMS.

In Table I, the lowest entropies are shown for LMS and KLMS. For both LMS and KLMS, the same masks and a large set of parameters were used in the experiments to find these minimal entropies. It is clear that using kernels instead of the inner-product introduces a gain for every image in our test set. Although KLMS increases the complexity, it drastically reduces the entropy of the residual. Entropy reductions up to 0.55 bpp are found. A compression gain of 1.70 is found on average, which would lead to a 5.7% bitrate reduction. A visual comparison of the absoluted residuals is shown in Fig. 4. For KLMS the edges in the image are much darker, and thus better predicted.

E. Impact of Adaptive Bandwidth

As mentioned above, the purpose of introducing an *Adaptive Bandwidth* (AB) is to increase robustness towards non-optimal kernel bandwidth settings for a given image. Experiments have shown that the usage of AB is beneficial in cases where the bandwidth was too small. Although for some images there is an entropy reduction of up to 0.05 bpp on non-optimal bandwidths, but in other images there was no substantial gain. The experiments were performed with a density sensitivity of 0.1.

F. Comparison with other coders

In this section, we provide experiments on the test set and compare our results to other state-of-the-art coders, being JPEG-LS and JPEG-2000 in lossless mode [17][18]. For JPEG-LS, the JPEG-LS reference coder v1.1.00 by HP was used, and the JasPer reference coder was used for JPEG-2000 [19].

TABLE I
COMPARING ENTROPY, VARIANCE, AND PREDICTION GAIN FOR LMS AND PLAIN KLMS

	LMS		KLMS		
	Entropy	Var	Entropy	Var	Gain
Aerial	5.70	184.41	5.61	164.63	1.12
Baboon	4.56	44.92	4.33	34.49	1.30
Balloon	3.12	8.88	2.85	4.31	2.06
Barb	4.92	88.62	4.34	36.70	2.41
Barb2	5.04	105.46	4.70	61.51	1.71
Board	4.00	40.30	3.59	14.25	2.83
Boats	4.31	42.12	3.99	23.59	1.79
Girl	4.07	24.50	3.73	13.58	1.80
Gold	4.69	51.49	4.57	42.62	1.21
Hotel	4.83	88.82	4.41	41.96	2.12
Lena	4.42	44.24	4.21	28.06	1.58
Livingroom	4.94	80.24	4.74	61.26	1.31
Moon	5.00	72.09	4.96	66.37	1.09
Zelda	3.88	18.10	3.73	12.26	1.48
Average	4.53	63.87	4.27	43.26	1.70

TABLE II
COMPARING LOSSLESS CODERS WITH ADAPTED KLMS + CAAC IN *bits per pixels* (BPP)

	JPEG-2000	JPEG-LS	KLMS/CAAC
Aerial	5.95	5.70	5.62
Baboon	4.20	5.04	4.33
Balloon	3.03	2.90	2.85
Barb	4.60	4.69	4.34
Barb2	4.79	4.69	4.70
Board	3.77	3.68	3.59
Boats	4.07	3.93	3.99
Girl	4.06	3.93	3.72
Gold	4.60	4.48	4.56
Hotel	4.59	4.38	4.41
Lena	4.31	4.24	4.20
Livingroom	4.87	4.71	4.74
Moon	5.26	5.08	4.97
Zelda	3.88	3.89	3.74
Average	4.42	4.38	4.27

Results can be seen in Table II. Remarkable is the *Baboon* image, which consists of almost only high spatial frequencies. Our scheme heavily outperforms JPEG-LS here, as the predefined predictors do not capture the image statistics.

Our conceptually simple non-linear prediction performs on average more than 0.1 bpp better (2.5% bitrate reduction) than the state-of-the-art on average. Although, the complexity of our scheme is considerately higher.

V. CONCLUSIONS

A novel generic lossless image compression scheme is presented. Through kernel adaptive filtering, a non-linear predictor is learned on the image itself. The residual is encoded by a context adaptive arithmetic coder. Experiments have shown that a bitrate reduction of more than 2.5% is achieved on average compared with JPEG-LS and JPEG-2000 lossless.

ACKNOWLEDGEMENT

The research activities described in this paper were funded by Ghent University, iMinds, Communication Systems Lab - Technische Universität Berlin, the Agency for Innovation by Science and Technology in Flanders (IWT), the Fund for Scientific Research Flanders (FWO Flanders), and the European Union.

REFERENCES

[1] C. Cortes and V. Vapnik, “Support-vector networks,” *Machine learning*, vol. 20, no. 3, pp. 273–297, 1995.
[2] T. Sikora, “Trends and perspectives in image and video coding,” *Proceedings of the IEEE*, vol. 93, no. 1, pp. 6–17, 2005.
[3] M. S. D. Ze-Nian Li, *Fundamentals of Multimedia*. Pearson Education, Inc., 2004.
[4] G. Ulacha and R. Stasinski, “Improving neural network approach to lossless image coding,” pp. 173–176, 2012.
[5] B. Meyer and P. Tischer, “TMWLego—-an object oriented image modelling framework,” p. 504, 2001.
[6] T. Sikora, “A novel kernel PCA/KLT approach for transform coding of waveforms,” in Press, Data Compression Conference (DCC ’15).
[7] ——, “R-D Performance of a novel 2D kernel transform for coding of signals,” submitted to IEEE International Conference on Image Processing (ICIP 2015), Montreal, 2015.
[8] R. Verhack, A. Krutz, P. Lambert, R. Van de Walle, and T. Sikora, “Lossy image coding in the pixel domain using a sparse steering kernel synthesis approach,” *IEEE International Conference on Image Processing (ICIP ’14)*, pp. 4807–4811, 2014.
[9] D. Santa-Cruz, R. Grosbois, and T. Ebrahimi, “Jpeg 2000 performance evaluation and assessment,” *Signal Processing: Image Communication*, vol. 17, no. 1, pp. 113–130, 2002.
[10] J. Principe, W. Liu, and S. Haykin, *Kernel Adaptive Filtering: A Comprehensive Introduction*, ser. Adaptive and Learning Systems for Signal Processing, Communications and Control Series. Wiley, 2010.
[11] W. Liu, P. P. Pokharel, and J. C. Principe, “The kernel least-mean-square algorithm,” *IEEE Transactions on Signal Processing*, vol. 56, no. 2, pp. 543–554, 2008.
[12] S. Van Vaerenbergh, M. Lázaro-Gredilla, and I. Santamaría, “Kernel recursive least-squares tracker for time-varying regression,” *IEEE Transactions on Neural Networks and Learning Systems*, vol. 23, no. 8, pp. 1313–1326, 2012.
[13] H. Takeda, S. Farsiu, and P. Milanfar, “Kernel regression for image processing and reconstruction,” *IEEE Transactions on Image Processing*, vol. 16, no. 2, pp. 349–366, 2007.
[14] K. Sayood, *Lossless Compression Handbook (Communications, Networking and Multimedia)*, 1st ed. Academic Press, 1 2003.
[15] P. Howard and J. Vitter, “Arithmetic coding for data compression,” *Proceedings of the IEEE*, vol. 82, no. 6, pp. 857–865, Jun 1994.
[16] University of South California, Signal and Image Processing Institute, “The USC-SIPI image database,” 1977. [Online]. Available: http://sipi.usc.edu/database/
[17] A. Skodras, C. Christopoulos, and T. Ebrahimi, “The JPEG 2000 still image compression standard,” *IEEE Signal Processing Magazine*, vol. 18, no. 5, pp. 36–58, 2001.
[18] M. J. Weinberger, G. Seroussi, and G. Sapiro, “The LOCO-I lossless image compression algorithm: principles and standardization into JPEG-LSko,” *IEEE Transactions on Image Processing*, vol. 9, no. 8, pp. 1309–1324, 2000.
[19] M. D. Adams and F. Kossentini, “JasPer: A software-based JPEG-2000 codec implementation,” vol. 2, pp. 53–56, 2000.