A Novel Color Printer Characterization Model

Alessandro Artusi and Alexander Wilkie

Institute of Computer Graphics and Algorithms

Vienna University of Technology

Karlsplatz 13/186, A–1040 Vienna, Austria

ABSTRACT

A key problem in multimedia systems is the faithful reproduction of color. One of the main reasons why this is a complicated issue are the different color reproduction technologies used by the various devices; displays use easily modeled additive color mixing, while printers use a subtractive process, the characterization of which is much more complex than that of self–luminous displays.

In order to resolve these problems several processing steps are necessary, one of which is accurate device characterization. Our study examines different learning algorithms for one particular neural network technique which already has been found to be useful in related contexts – namely radial basis function network models – and proposes a modified learning algorithm which improves the colorimetric characterization process of printers.

In particular our results show that is possible to obtain good performance by using a learning algorithm that is trained on only small sets of color samples, and use it to generate a larger look–up table (LUT) through use of multiple polynomial regression or an interpolation algorithm. We deem our findings to be a good start point for further studies on learning algorithms used in conjunction with this problem.

Keywords: Radial basis function networks, regression, colorimetric characterization of printing devices

1. INTRODUCTION

In multimedia systems, different color reproduction devices — while serving the same purpose — exhibit large discrepancies in their raw output. This is due to the fact that they usually employ different color mixing technologies (additive or subtractive),

Correspondence: Email: artusi@cg.tuwien.ac.at

use different input color spaces and hence have different gamuts, and that their device characteristics can change with time and usage. These facts usually do not permit a faithful matching of colors between devices if no precautions are taken.

Colorimetric characterization is one step in the colorimetric reproduction process that permits faithful image reproduction across different display devices. Its goal is to define a mapping function between the device–dependent color spaces in question (such as RGB or CMYK) and device–independent color spaces (such as CIELAB or CIEXYZ), and vice versa.

There are three main approaches to defining this mapping function: physical models, empirical models and exhaustive measurements.[8] Physical modeling of imaging devices involves building mathematical models that find a relationship between the colorimetric coordinates of the input (or output) image element and the signals used to drive an output device (or the signals originating from an input device). The advantage of these approaches is that they are robust, typically require few colorimetric measurements in order to characterize the device, and allow for easy recharacterization if some component of the imaging system is modified. The disadvantage is that the models are often quite complex to derive and can be complicated to implement. Physical models are often used for the colorimetric characterization of displays and scanners.

Empirical modeling of imaging devices involves collecting a fairly large set of data and then statistically fitting a relationship between device coordinates and colorimetric coordinates. Empirical models are often higher–order multidimensional polynomials, or neural network models. They require fewer measurements than LUT techniques, but they need more than physical models. Empirical models are often used for scanners and printers.

Often the colorimetric characterization of printers requires an exhaustive measurement in order to obtain good performances. Typically $9 \times 9 \times 9$ samples of the device drive signals are sampled and colorimetrically measured. Many more measurements have to be used for devices with poor repeatability.

Lookup tables can be used to process image data via multidimensional interpolation. This technique has different disadvantages: the large number of measurements that has to be made, difficulties in interpolating the highly nonlinear data and difficult recharacterization if any aspect of the device changes. The advantage of exhaustive measurement techniques is that they require no knowledge of the device physics.

In general a good algorithm for colorimetric characterization must have the following characteristics: small training set, fast response, good accuracy and it must allow for a fast recharacterization. This paper proposes a modification of an existing learning algorithm[6] to train radial basis function networks to solve the problem discussed so far, namely the colorimetric characterization of printers. This learning algorithm has fast training and test phases, good accuracy, and it also requires a comparatively small training set.

In section 2 we discuss related previous work, section 3 describes the background for radial basis function networks, the models used in our experiments, and the proposed new model. Section 4 explains how the training and test set were generated and reports some experimental results.

2. STATE OF THE ART

There is a large number of publications on the colorimetric characterization of printers that propose different models for solving this problem: Kang and Anderson[16] propose the application of neural networks and polynomial regression. Albanese, Campadelli and Schettini[3] and Tominaga[20] have used feed–forward neural networks trained by back–propagation and obtained promising results. However, their approach also has some disadvantages: the need for a big training set (several hundred to several thousand samples), high computational cost, and a comparatively large maximum color error for high quality color reproductions. One of these problems has been solved by Artusi, Campadelli and Schettini[2]: in their work they reduced the size of the training set to 216 measured samples, while retaining a maximum error that is comparable to — in some cases even better than — previous approaches.

There are no references to be found in the literature about the use of radial basis function networks for the colorimetric characterization of printers, but there is a wealth of other publications about them and their applications (such as Orr,[15] Bishop,[4] Carozza[6] and Lee[10]).

The work we present is novel in seven ways: to begin with, this is the first work that uses radial basis function networks to resolve the colorimetric characterization of printers. Second, we used a new learning model to train such networks; our approach is based on a modification of the proposal by Carozza.[6] Third, we use only 125 measured samples for the training of the network. Fourth, the computational costs for this training are very low when compared to previous techniques and allow to use this model in consumer products. Fifth, it is a general model which one can also use to define other transformations between color spaces. Sixth, it is possible to have a fast recharacterization of the device because the computational cost of the training phase is low. Finally, it improves on the performance of multiple polynomial regression and tetrahedral interpolation.

3. BACKGROUND

There are two main types of regression problems in statistics[15]: parametric and non–parametric. In parametric regression the form of the functional relationship between the dependent and independent variables is known, but may contain parameters whose values are unknown, and it is possible to successfully estimate the desired result from the training set.

In the case of non–parametric regression there is no, or very little, *a priori* knowledge about the form of the true function which is being estimated. The colorimetric characterization problem, presented in these papers, is a non–parametric regression problem, because one does not know the mapping function properties the algorithm will arrive at in advance. There are different approaches to resolve non–parametric regression problems; when one uses equation systems in this context they may contain many free parameters that have no physical meaning in the problem domain (interpolation models, multiple polynomials regression), or one can use neural networks instead.

3.1. Neural networks

The base of a neural network is a formal neuron. It is defined as a series of inputs, a series of outputs and by a function that maps specific inputs to series of outputs.[5] Neural networks consist of collections of connected formal neurons. Each formal neuron computes a simple nonlinear function F on the weighted sum of its input. The function F is referred to as *activation function* and its output is defined as the activation of the formal neuron. Long term knowledge is stored in the network in the form of interconnection weights that link such formal neurons.

There are different neural network structures[1]: total connected networks, partial connected networks, multilayer networks (feedforward, feedback), and autoassociative networks. In a neural network, the learning phase is a process where a set of weights are defined that produces a desired response as a reaction to certain input patterns.[5] There are two main techniques for the learning phase: supervised learning and non–supervised learning. In supervised learning the function is learned from samples which a “teacher” supplies. This set of samples, referred to as the training set, contains elements which consist of paired values of the independent (input) variable and the dependent (output) variable.[15] In the case of non–supervised learning, it reaches an internal model that captures the regularity in the inputs without taking other information into account.[1]

3.2. Basis functions

A linear model for a function $f(x)$ can be expressed in the following form[15]:

$$f(x) = \sum_{j=1}^{m} w_j * h_j(x) \tag{1}$$

The model f is expressed as a linear combination of a set of m fixed functions, often referred to as basis functions. The flexibility of f, its ability to fit many different functions, derives only from the freedom to choose different values for the weights. The basis functions and any parameters which they might contain are fixed. If the basis function parameters can also change during the learning process, the model is considered non–linear. Any set of functions can be used as a basis set, although it is desirable that they are differentiable. There are many different classes of functions that one can use as basis functions, for example:

- Fourier series

$$hj(x) = \sin\left(\frac{2 * \pi * j * (x - \Theta j)}{m}\right) \tag{2}$$

- Logistic functions

$$h(x) = \frac{1}{1 + \exp(b^t * x - b_0)} \tag{3}$$

- Polynomial functions

Radial functions are a special class of basis function. Their characteristic feature is that their response decreases (or increases) monotonically with the distance from a central point. The center c, the distance scale, and the precise shape of the radial function r are parameters of the model, and are fixed if it is a linear model. Two possible examples of radial functions are:

- Gaussian

$$h(x) = \exp\left(\frac{-(x - c)^2}{r^2}\right) \tag{4}$$

- Multiquadratic

$$h(x) = \frac{\sqrt{r^2 + (x-c)^2}}{r} \tag{5}$$

3.3. Radial basis function networks

Radial Basis Function Networks (RBFn) are derived from the exact interpolation problem[4] by introduction of several changes. The exact interpolation problem attempts to map every input point exactly onto a corresponding target point. The Radial Basis Function (RBF) approach introduces a set of basis functions equal to the number of input points. Furthermore, the following modifications are necessary for the introduction of RBFns:

- The number of basis functions does not have to be the same as the number of input points, and is typically smaller.
- The bias parameters are included in the sum term of the linear model from equation (1).

In the case of a non–linear model there are two more modifications if the basis function can move, change size, or if there is more than one hidden layer:

- There is no constraint that the centers of basis functions have to be input points; instead, determining these centers is part of the training process.
- Instead of a unique parameter r, every basis function has a parameter r_j, the value of which is obtained during the training process.

An example of a traditional RBFn with one hidden layer is shown in figure 1. Each of n components of the input vector x feeds forward to m basis functions whose outputs are linearly combined with weights w_j into the network output. This example could be a linear model of RBFn if the parameters of the basis function H_i, in the hidden layer, do not change during the learning process. Instead if they change during the learning process the RFBn is non-linear. Also if there is more than one hidden layer of basis functions H_i in the structure of the RBFn the network is a non-linear model. There are two stages for the training phase: determining the basis function parameters, and the finding of appropriate weights.

3.4. Linear network models

In the case of a linear model, the parameters of the basis functions are fixed, and the goal is to minimize the sum of the squared errors in order to obtain the optimal weights vector[15]:

$$S = \sum_{i=1}^{p} (y_i - f(x_i))^2, \tag{6}$$

where p is the pattern number, and (x_i, y_i) are the input and output vector targets of the respective training set. The optimal weights, in matrix notation, are:

$$W = A^{-1} * H^t * Y, \tag{7}$$

where H is referred to as design matrix and is the output of the RBF, A^{-1} is the covariance matrix of the weights W, and the matrix Y is the output target. In many cases this amounts to an over–fitting problem, and the main effect of this is that the neural network loses its generalization capacity. In order to counter the effects of over–fitting it is possible to utilize results from *regularization theory*. Regularization theory suggests to attach a term called *regularization parameter* in equation (6), in order to obtain a weight vector which is more robust against noise in the training set. In regularization theory, there are two main techniques: global ridge regression, where one uses unique regularization parameters for all basis functions, and local ridge regression, where there is a regularization parameter for every basis function. For the case of global ridge regression one has to modify equation (6) as follows:

$$C = \sum_{i=1}^{p} (y_i - f(x_i))^2 + k * \sum_{j=1}^{m} w_j^2, \tag{8}$$

where m is the basis function index. In the case of local ridge regression equation (6) has to be modified to:

$$C = \sum_{i=1}^{p} (y_i - f(x_i))^2 + \sum_{j=1}^{m} k_j * w_j^2. \tag{9}$$

3.5. Forward selection

One way to give linear models the flexibility of non–linear models is to go through a process of selecting a subset of basis functions from a larger set of candidates.[15] In linear regression theory[18] subset selection is well known and one popular version is forward selection in which the model starts empty $(m = 0)$ and the basis functions are selected one at a time and added to the network. The basis function to add is the one which most reduces the sum squared errors in equation (6); this is repeated until no further improvements are made. There are different criterions to decide when to stop the forward selection process: generalised cross-validation (GCV),[9] unbiased estimate of variance (UEV),[7] final predictor error (FPE)[11] and the Bayesian information criterion (BIC).[19]

A matrix which often used in the analysis of linear networks is the projection matrix P

$$P = I_p - H * A^{-1} * H^t; \tag{10}$$

this square matrix projects vectors in p-dimensional space perpendicular to the m-dimensional space spanned by the model. The importance of the matrix P is evident in the new form of the equation (6)

$$S = y^t * P^2 * y, \tag{11}$$

and, in the case of local ridge regression, the cost function (9) is

$$C = y^t * P * y. \tag{12}$$

A good measure of model performance is its variance. It can be estimated by

$$r^2 = \frac{S}{p - m}, \tag{13}$$

where p is the number of patterns in the training set and m is the number of parameters in the model (weigths). The variance (13) is calculated using one of these methods after each new basis function is added. With the GCV selection criterion the variance is esitimated as

$$r^2_{GCV} = \frac{p * y^t * P^2 * y}{(p-l)^2}, \tag{14}$$

where l is the effective number of parameters

$$l = p - trace(P). \tag{15}$$

For the *unbiased estimate of variance* criterion (UEV), it is

$$r^2_{UEV} = \frac{y^t * P^2 * y}{(p-l)}, \tag{16}$$

while for the *final prediction error* (FPE) it amounts to

$$r^2_{FPE} = \frac{p+l}{p-l} * \frac{y^t * P^2 * y}{p}, \tag{17}$$

Finally, for the *Bayesian information criterion* (BIC) it is

$$r^2_{BIC} = \frac{p + (ln(p) - 1) * l}{p-l} * \frac{y^t * P^2 * y}{p}. \tag{18}$$

An efficient method of performing forward selection is the orthogonal least squares method as discussed in Orr[14]; it is based on the orthogonalisation of the columns of the design matrix. This involves a particular form of the covariance matrix, which consists of a triangular and a diagonal matrix; this fact can be used to greatly accelerate the computation.

3.6. Non–linear network models

In the non–linear model the basis function parameters are not fixed, and it is possible to estimate them during the learning process. This gives more flexibility to the network model. In literature there is a large number of publications that propose different models to estimate these basis function parameters. In this section we present two existing models, also used in our experiments[6].[10] In particular we present in detail the model of Carozza,[6] in order to understand the differences with the model presented in this paper. In Carozza[6] a new algorithm for function approximation from noisy data was presented. The authors proposed an incremental supervised learning algorithm for RBFn. It added a new node at every step of the learning process, and the basis function center c and the output connection weights are settled in accordance with an extended chained version of the Nadaraja–Watson estimator. The output network for the neuron M, in accordance with the Nadaraja–Watson estimator, is

$$net_M(x) = \frac{\sum_{j=1}^{M} w_j * exp(-\frac{||x-x_{k_j}||^2}{2*r_j^2})}{\sum_{j=1}^{M} exp(-\frac{||x-x_{k_j}||^2}{2*r_j^2})}. \tag{19}$$

The goal is to minimize the empirical risk

$$E_M = 1/p \sum_{k=1}^{p} (y_k - net_M(x_k))^2. \tag{20}$$

The chained version of the estimator (19) obtained for an incremental approach is

$$net_M(x) = \frac{net_{M-1}(x) * den_{M-1}(x) + w_M * exp(-\frac{||x-x_{k_M}||^2}{2*r_M^2})}{den_M(x)}, \tag{21}$$

where

$$den_M(x) = \sum_{j=1}^{M} exp(-\frac{||x-x_{k_j}||^2}{2*r_j^2}). \tag{22}$$

In order to reduce the empirical risk given by (20), the output weights for the neuron j are chosen as

$$w_j = y_{k_j} - net_{j-1}(x_{k_j}), \tag{23}$$

where the output y_j is such that

$$(y_{k_j} - net_{j-1}(x_{k_j}))^2 = max_{k=1,....,p}(y_k - net_{j-1}(x_k))^2, \tag{24}$$

Moreover, since this quantity is affected by normalization factor den_j, the authors multiply w_j for this factor, and arrive at the final chained version of (19)

$$net_M(x) = \frac{net_{M-1}(x) * den_{M-1}(x)}{den_M(x)} + w_M * exp(-\frac{||x-x_{k_M}||^2}{2*r_M^2}), \tag{25}$$

The variance r of the basis functions is determined by an empirical risk driven rule based on a genetic–like optimization technique. A population of individuals is generated by applying the mutation

$$r_{j,new}^2 = r_j^2 * (1 + k), \tag{26}$$

where k is a random number in a fixed range $[-a, a]$. This mutation step is iterated a fixed number of times if the empirical risk (20) associated with $r_{j,new}^2$ is less than the same quantity associated with r_j^2. On a different note, Lee[10] introduces the concept of *robust* RBFs and makes suggestions on how to choose a function candidate which fulfils this role.

4. PROPOSED MODEL

4.1. Modified estimator for RBFN weights

The proposed model is a modification of an existing one.[6] In particular we have modified the estimation of the weights by introducing a pseudoinverse matrix[4] instead of using the extend chained version of the Nadaraja–Watson estimator for updating the weights. The pseudoinverse method works by resolving the following general system of linear equations:

$$H * W = Y, \tag{27}$$

where H is the matrix of the basis functions o design matrix of dimension $numberinputvectors \times numberRBF$, Y is the matrix of output vectors of dimension $numberoutputvectors \times 3$, and W is the weights matrix of dimension $numberRBF \times 3$. The number 3 indicate the three dimensional space of the input and output vectors. In this equation there appears a Moore–Penrose pseudoinverse[12] in the form of the matrix B, which has the same dimensions as H^T, and that has to satisfy the following four conditions:

$$\begin{aligned} H * B * H &= H \\ B * H * B &= B \\ H * B & \quad \text{is Hermitian} \\ B * H & \quad \text{is Hermitian.} \end{aligned} \tag{28}$$

The solution of the linear system (27), throught the pseudo inverse matrix, is the matrix W (29):

$$W = H^t * (H * H^t)^{-1} * Y. \tag{29}$$

There are different methods to resolve the linear system (27): Gauss-Jordan elimination, Gaussian elimination with Back-substitution, LU decomposition and Singular value decomposition (SVD) etc. We used the SVD method because in comparision with the other methods it gives satisfactory results and is a most fast algorithm.[17] Let take a general matrix X. This method is based on the following theorem of linear algebra: any $M \times N$ matrix X whose number of rows M is greater than or equal its number of columns N, can be written as the product of an $M \times N$ column–orthogonal matrix U, an $N \times N$ diagonal matrix Q with positive or zero elements (the singular values), and the transpose of an $N \times N$ orthogonal matrix V. In other words the matrix X can be written as:

$$X = U * Q * V^t. \tag{30}$$

We have three different cases, about the form of the matrix X:

- In the case the matrix X is square, this means that $N = M$, then U, V, and W are all square matrices of the same size. Their inverse are also trivial to compute, in fact U and V are orthogonal, so their inverses are equal to their transposes. Instead Q is diagonal, so its inverse is the diagonal matrix whose elements are the reciprocals of the element q_j. In this way the inverse of the matrix X is:

  $$X^{-1} = V * [diag(\frac{1}{q_j})] * U^T, \tag{31}$$

  in our case $X = H * H^t$ and the value of the unknonws matrix W is:

  $$W = H^t * V * [diag(\frac{1}{q_j})] * U^T * Y. \tag{32}$$

  If the value of the term q_j is equal zero, we need to replace the term $1/q_j$ with zero, do not have a division by zero.

- If there are fewer linear equations M than unknowns N, then you are not expecting a unique solution. Usually there will be an $N - M$ dimensional family of solutions. In this case is possible to augment your left-hand side matrix with rows of zeros underneath its M nonzero rows, untill it is filled up to be square, of size $N \times N$. In this way the matrix becomes to be square and we can apply the SVD in the way explained for the square matrix, equation (32).

- If there are more equations than unknowns, we are in the case of overdeterminated set of linear equations, and the equation for the square case, equation (32), can be apply without modification.

The output of the network trained with our model is not calculated as in the model proposed by Carozza,[6] but rather by a sum of products of basis function output weights. The RBFn model used in our experiments adopts only one hidden layer, and three dimensional Gaussian basis functions for the nodes in the hidden layer with the same variance r. In our model there are some parameters: the average c and the variance r for the basis function, the weights w, *times* as the number of epochs and E as initial error. Other initializations depend on the application at hand. After preliminary experiments, we choose the following values to be suitable for our particular case: $c = \text{random}(0,1), r = 0.5, w = \text{random}(0,1), \text{times} = 10$. The initial error E has to be large in order for error reduction to work, and the node numbers of the hidden layer N begin from 1.

4.2. Our algorithm

In this section we present our algorithm in pseudo code format, followed by a brief explanation.

```
parameter initialization
while ( termination criterion is not met )
    if ( N > 1 )
        $r_N$ = 0.5;

        Compute the average error E of the output model
        with respect to the output target

/* update of basis function parameter  c */

c = the input vector with index of the patterns
            with maximum error
```

```
Compute the weights w with the pseudoinverse

    end if

    /* update of basis function parameter  r */

    for  ( k = N to 1 )
        while ( l <= times )
            alpha  = random( -0.5, 0.5 )
            $r_{new}$ = $r_k$ * ( 1 + alpha ) + epsilon

    Compute average error $E_{new}$ of the output model
            with respect to the output target

            if ( $E_{new}$ < E )
                $r_k$ = $r_{new}$
                E = $E_{new}$
            end if
            l = l + 1
        end while
    end for

    Compute the weights w with the pseudoinverse
    and use the basis function parameter update

    N = N + 1
```

```
end while.
```

The parameter initialization defines the initial values for the parameters of the radial basis functions: average c and variance r, and other network parameters such: the weights w, the number of epochs $times$, the initial value of the hidden number N and the error E. In order to obtain it we follow the initialization suggested in the section 4. This model has the capacity to generate the structure of the RBFn. In fact, it adds a new node during each iteration and initializes its parameters c, r and w. This operation stops when the termination criterion is satisfied. In the case of the first node, $N = 1$, we need only perform the initialization of the parameters r and w because the average c is chosen randomly while. In the other cases the average c is the input vector with index of the output model with the maximum error with respect to the output target. The updating of the weights parameters is performed with the pseudo inverse technique (32), the variance parameter r of the radial basis functions is updated following the technique proposed by Carrozza.[6] This technique consists to use the following equation for each node k:

$$r_{new} = r_k * (1 + alpha) + epsilon, \tag{33}$$

where the terms *alpha* and *epsilon*, after preliminar experiments, were set to the following initial values: for alpha a random value in the range from -0.5 to 0.5, and epsilon was set to 0.01. We update the value r_k for the node k with the new value obtained with equation (33) only if the new error value E_{new}, computed with the new parameters r, c and w, is less than the old error value. We repeat the updating operation for the parameter r_k until the maximum epochs condition is respected $l <= times$. In case the error condition is not matched till the maximum value of epochs $times$ the new value for the parameters r_k remain equal at the old one. This operation is repeated for all nodes k of the hidden layer. When the updating has been performed we compute the new value for the weights w with the pseudo inverse technique (32), with the new parameters values r, c and the weights w for the nodes of the hidden layer.

5. EXPERIMENTAL RESULTS

5.1. Training and test sets

The colorimetric patterns of the training and test sets which we used in these experiments are formed by pairs of three dimensional vectors. One of these vectors specifies the CMY coordinates, and the other specifies the CIELAB coordinates of a printed color. The color sets for the training and test phases are obtained by printing a number of hues, specified in CMY space, in squares of approximately 1 cm^2 at the highest resolution the printer has to offer. These color swatches are then measured with a SPECTROLINO spectrophotometer produced by GretagMacbeth.

For the training phase of our experiments we used four different sets of this kind, labeled *Training1* through *Training4*. The sets *Training1* and *Training2* were made up of 729 and 125 colors, respectively, which were obtained by uniform sampling in CMY space. The sets *Training3* and *Training4* consisted of 392 and 252 colors, which were obtained as suggested by Moroni.[13] The Test set contains 777 colors obtained by random sampling of CMY space. The error of the models was calculated in CIELAB space according to the formula

$$\Delta_E = \sqrt{(L - L')^2 + (a - a')^2 + (b - b')^2}, \tag{34}$$

where (L, a, b) is the output of the models and (L', a', b') is the target output. The experiments were conducted using several different ink–jet printers, namely an Epson Stylus Pro5000 (with photo quality paper), an Olivetti Artjet 20 (with coated paper) and a HP2000C (with cut sheet paper). The code for the algorithms tested in these experiments was written in C and Matlab.

5.2. Results

The first step in our research was to find a good learning algorithm to train the RBFn function with small training sets. In order to do this we evaluated different learning algorithms in the following order: first linear models with forward selection without and with local ridge regression (GCV,[9] UVE,[7] FPE,[11] BIC[19]); we used the Matlab implementation of these algorithms by Orr.[15] Then we considered nonlinear models as proposed by Lee[10] and Carozza[6]; we implemented these ourselves in C. Our proposed own non-linear model was again implemented in Matlab. All these algorithms were then compared to multiple polynomial regression and tetrahedral interpolation.

When we wanted to find out whether a particular learning algorithm is adequate to solve the posed problem, we first tested it only for the conversion CMY $\rightarrow$ CIELAB on an Epson Stylus Pro5000, and only if this preliminary test turned out favourably, we conducted further experiments on other printers and with the conversion CIELAB $\rightarrow$ CMY. In every table of the columns for the training and test sets we show the average error on the left and the maximum error on the right of the respective cells.

We began the experiments by initially testing existing learning algorithms, specifically those proposed by Golub,[9] Efron,[7] Mallows[11] and Schwarz[19] on the color printer Epson Stylus Pro 5000; we used two training sets (labeled *Training1* and *Training2* in the tables) and tested the networks with a set labeled *Test*. The results for linear model with forward selection and without ridge regression are shown in table 1 for the function CMY $\rightarrow$ CIELAB.

Our results demonstrate that this model is already able to improve on the performance of multiple polynomial regression for polynomials with up to 60 terms and tetrahedral interpolation, shown in table 3, in the case of the set *Training1*. However, this does not extend to the set *Training2*, where the unmodified RBFn approach fares no better than the conventional techniques. This is probably due to the fact that in this case the network encounters over–fitting problems as mentioned in section 3.4. In order to resolve this problem we have tried to use regularization theory (local and global ridge regression), but this failed to improve the results. These results are shown in the table 2.

Another approach was to generate more training sets with smaller numbers of samples compared to the original large set *Training1*. We produced two such sets in the way suggested by Moroni,[13] labeled *Training3* and *Training4*, with 392 and 252 samples respectively. Results from test runs with these sets are reported in table 6, for multiple polynomial regression, instead for RBFn linear models with forward selection without local ridge regression, and with local ridge regression in the tables 4 and 5 respectively. These results show that there are indeed improvements with respect to multiple polynomial regression with the set *Training3*, and equal performance with *Training4*.

However, compared to the still large size of the new reduced training sets the improvement is rather small. The methodology for generation of the condensed training set proposed by Moroni[13] apparently does not allow for the desired increase in efficiency.

In this phase of the experiments we also tried other innovative learning algorithms found in literature (such as Lee[10] and Carozza[6]), but the results we obtained were of poor quality. We then altered our strategy and decided to modify an existing learning algorithm. Our the choice here has been to modify the learning algorithm proposed by Carozza,[6] mainly because it does not get significantly more complex when it is modified, and also because it does not have inherent convergence problems like the algorithm of Lee.[10] The results we obtained are shown in tables 7 and 8; the tests were done on an Epson Stylus Pro5000 for the function CMY $\rightarrow$ CIELAB.

The results show how it is possible to obtain performance that is better than that of multiple polynomial regression and tetrahedral interpolation with only 125 samples. This is shown in the last two columns beginning at the right of table 3 for both sets (Training and Test) and both error metrics (average and maximum). In order to make sure that these results are consistently reproducible over time we repeated the experiment in May, June and July, and used different (but similar) sets of 125 training samples in each case; the progression of the error over time is shown in table 8. The results are in line with the first experiment reported in table 7. In order to validate our model we performed similar tests on two more ink-jet color printers from other manufacturers, namely a HP2000C and an Olivetti Artjet20. The results are shown in table 9 for the Epson Stylus Pro5000, in table 10 for the HP2000C and in table 11 for the Olivetti Artjet20. The data is also compared against results from multiple polynomial regression; all tests were done for the function CMY $\rightarrow$ CIELAB.

The results show that there is an improvement over multiple polynomial regression for every ink-jet color printer. The results shown in the tables 9 to 11 demonstrate that our model has general validity.

The final problem we discuss in this paper is the definition of the function CIELAB $\rightarrow$ CMY using our model; now the main problem is that of the definition of a suitable termination condition for the learning process. If we use the same condition that we used for the definition of the function CMY $\rightarrow$ CIELAB and compute the error in CIELAB space, is necessary to use the inverse transformation for each termination check, which in turn also implies that a neural network for this inverse case ought to be used. This solution is not feasible because the computational cost of the inverse network training phase is significant, and too much accuracy is lost through the repeated conversions. Neither is it possible to compute the error in CMY space, because this approach is inherently incapable of knowing when to stop the learning process.

The solution that we have adopted has been to generate a LUT with 729 uniform samples in CMY space with our type of RBF network, which was trained with 125 samples. We then used this LUT with multiple polynomial regression on the set *Test*. We compared this result to the result obtained with a LUT of 729 uniform samples, which were printed and measured from CMY space, and which were also used with multiple polynomials regression on the same set of samples. The results are reported in table 12; this test was performed on an ink–jet printer of the type Epson Stylus Pro5000.

These results show that is possible to generate a LUT from only 125 initial printed and measured samples with our method, compared to 729 samples used by multiple polynomial regression or interpolation models. Our model permits a fast recharacterization of ink–jet color printers because it needs only 125 printed and measured samples, and in addition its training phase is very fast. On a 500 MHz Pentium II Celeron system with 128 Mbytes of RAM the training time with the initial 125 samples is in the order of a few minutes, and the time it takes to generate the 729 entry LUT is one additional minute.

6. CONCLUSION AND FUTURE WORK

We have presented a new learning algorithm, which is a modification of a known technique, that trains the RBFn model for the colorimetric characterization of color printers. Our algorithm needs a training set of only 125 samples in order to train the RBFn. With this model is even possible to generate a LUT of 729 samples, beginning with only 125 printed and measured samples, and to use this LUT with other standard algorithms of colorimetric characterization.

The computational cost is very low in the training and testing phases, and is even better than the performance of other standard colorimetric characterization models (multidimensional polynomials regression an tetrahedral interpolation). It is our opinion that the results suggest that may be possible to use this algorithm in consumer products, because we have been able to resolve the two problems that has so far limited the more widespread use of such methods: high computational cost, and the large number of training samples needed. The small size of the training set also permits a fast recharacterization of devices.

We believe that there are several possible ways to evolve these models for colorimetric characterization problems: investigation of different mathematical models for the estimation of the basis function parameters, research on different mathematical models for the estimation of the weights, introduction of one or two more hidden layers in the structure of the RBFn, and eventually experiments that involve combinations of these new techniques.

REFERENCES

1. A. Artusi, “Applicazione di algoritmi di apprendimento alla caratterizzazione colorimetrica di stampanti a colori” Tesi di laurea in Informatica, Univ. Degli studi di Milano, A.A 1996-97.
2. A. Artusi, P. Campadelli, R. Schettini “Boosting learning Algorithms for the Colorimetric Characterization of Color Printers”. Proc. WIRN’98: 283-289, Vietri, Giugn 1998.
3. S. Albanese, P. Campadelli, R. Schettini “Inkjet color printer caliration by back-propagation”. Communication at the Workshop on Envaluation Criteria of Neural Net Efficiency in Industrial Applications, Vietri, November 1995.
4. C. M. Bishop “Neural Networks for Pattern Recognition”. Calendor Press Oxford, 1996.
5. J.M.Bishop, M.J. Bushnell, S. Westland “Application of Neural Networks to the Computer Recipe Prediction” Color research and application, Volume 16, Number 1, 3-9, February 1991.
6. M. Carozza, S. Rampone “Function approximation from noisy data by an incremental RBF network”. Pattern Recognition, volume 32, numero 12, 2081-2083, 1999.
7. B. Efron, R. J. Tibshirani “An Introduction to the Bootstrap”. Chapman and Hall, 1993.
8. Mark D. Fairchild “Color Appearance Models”, Addison Wesley 1998.
9. G.H. Golub, M. Heat, G. Wahba “Generalised croos-validation as a method for choosing a good ridge parameters”. Technometrics, 21(2), 215-223, 1979.
10. Lee et al. “Robust radial basis function neural networks”. IEEE trans, on systems, man and cybernetics, vol 29, 674-685, 1999.
11. C. Mallows “Some comments on Cp”. Technometrics, 15, 661-675, 1973.
12. Mathworks “Matlab Function reference”. website ”http://www.mathworks.com” , 2000.
13. N. Moroni Barcelona HP Res. Labs, Personal Communication, 1996.
14. M. J. L. Orr “Regularisation in the selection of radial basis function centres”. Neural Computation, 7(3), 606-623, 1995.
15. M. J. L. Orr “Introduction to Radial Basis Function Networks”. Center of Cognitive Science, University of Edinburgh, 1996.
16. H.R. Kang, P.G. Anderson “Neural network application to the color scanner and printer calibrations”. Journal of Electronic Imaging 1: 25-135, 1992.

17. W. H. Press, S. A. Teukolsky, W. T. Vetterling, B. P. Flannery "Numerical Recipes in C The Art of Scentific Computing". Second Edition, Cambrige University press, 1992.

18. J.O. Rawlings "Applied Regression Analysis". Wadsworth & Brooks/Cole, Pacific Grove, CA, 1988.

19. G. Schwarz "Estimating the dimension of a model". Annals of Statistics, 6, 461-464, 1978.

20. S. Tominaga "A color mapping method for CMYK Printers". Proc. 4th IS&T&SID's Color Imaging Conference: Color Science, Systems and Applications, 1996.

Alessandro Artusi

received his masters degree in computer science at the University of Computer Science in Milano in 1997. From November 1998 to August 2000, he worked on an Italian project ”Progetto Italiano Tema 4 Nuove Technologie per la stampa tessile”. In 2000, he joined the Institute of Computer Graphics and Algorithms of the Vienna University of Technology, as a researcher on an European project ”PAVR”, and started his PhD. He worked from November 2001 to March 2002 on an Austrian project ”Real Time Renderin of Urban Environments”. In April 2002 started to work on an European project ”RealReflect”. The main field of his present interests include colour science, colorimetric characterization, color appearance, gamut mapping, tone mapping, real-time visualization and neural networks.

Alexander Wilkie

is an assistant professor at the Institute of Computer Graphics and Algorithms of the Vienna University of Technology. He finished his masters degree in computer science there in 1996, and received his PhD in 2001, also from the Vienna University of Technology. His research interests include predictive rendering, colour science and global illumination.

TABLES

Table 1. Error comparison of CMY → CIELAB conversion using the initial RBFn, with the selection criterious (GCV, UEV, FPE, BIC), without ridge regression for the *Training1*, *Training2* and *Test* datasets on the Epson Stylus Pro5000.

Method	**Training1**		**Test**		**Training2**		**Test**	
	Avg	**Max**	**Avg**	**Max**	**Avg**	**Max**	**Avg**	**Max**
Initial RBFn GCV	0.287	1.182	0.823	5.551	0.525	1.487	3.025	10.867
Initial RBFn UEV	0.114	0.486	0.823	5.249	0.016	0.067	3.006	10.635
Initial RBFn FPE	0.157	0.706	0.808	5.188	0.016	0.067	3.006	10.635
Initial RBFn BIC	0.185	0.754	0.827	5.570	0.016	0.067	3.006	10.635

Table 2. Error comparison of CMY → CIELAB conversion using the initial RBFn, with the selection criterious (GCV, UEV, FPE, BIC), with ridge regression for the *Training1*, *Training2* and *Test* datasets on the Epson Stylus Pro5000.

Method	**Training1**		**Test**		**Training2**		**Test**	
	Avg	**Max**	**Avg**	**Max**	**Avg**	**Max**	**Avg**	**Max**
Initial RBFn GCV	0.477	2.171	0.834	4.818	0.525	1.487	3.024	10.873
Initial RBFn UEV	0.469	2.118	0.827	4.841	0.035	0.100	3.003	10.637
Initial RBFn FPE	0.469	2.118	0.827	4.841	0.035	0.100	3.003	10.637
Initial RBFn BIC	0.522	2.295	0.867	4.653	0.035	0.100	3.003	10.637

Table 3. Error comparison of CMY $\rightarrow$ CIELAB conversion using regression with different polynomials and tetrahedral interpolation for the *Training1*, *Training2* and *Test* datasets on the Epson Stylus Pro5000.

Method	**Training1**		**Test**		**Training2**		**Test**	
	Avg	**Max**	**Avg**	**Max**	**Avg**	**Max**	**Avg**	**Max**
Regression polynomials 60 terms	2.391	10.976	3.172	10.979	2.495	7.674	3.228	9.798
Regression polynomials 69 terms	2.310	10.924	3.109	10.918	2.458	7.639	3.176	10.111
Regression polynomials 87 terms	1.813	8.041	2.515	8.195	1.721	5.453	2.654	8.277
Regression polynomials 105 terms	1.636	8.798	2.306	8.661	1.475	4.337	2.571	7.676
Tetrahedral interpolation	0.0	0.0	0.812	5.018	0.0	0.0	2.116	7.490

Table 4. Error comparison of CMY $\rightarrow$ CIELAB conversion using the initial RBFn, with the selection criterious (GCV, UEV, FPE, BIC), without ridge regression for the *Training3*, *Training4* and *Test* datasets on the Epson Stylus Pro5000.

Method	**Training3**		**Test**		**Training4**		**Test**	
	Avg	**Max**	**Avg**	**Max**	**Avg**	**Max**	**Avg**	**Max**
Initial RBFn GCV	0.223	1.030	1.296	4.127	0.200	0.787	3.429	11.637
Initial RBFn UEV	0.099	0.378	1.577	5.231	0.119	0.600	3.723	16.176
Initial RBFn FPE	0.109	0.432	1.528	5.108	0.192	0.735	3.446	11.746
Initial RBFn BIC	0.138	0.578	1.496	4.531	0.192	0.735	3.446	11.746

Table 5. Error comparison of CMY → CIELAB conversion using the initial RBFn, with the selection criterious (GCV, UEV, FPE, BIC), with ridge regression for the *Training3*, *Training4* and *Test* datasets on the Epson Stylus Pro5000.

Method	**Training3**		**Test**		**Training4**		**Test**	
	Avg	**Max**	**Avg**	**Max**	**Avg**	**Max**	**Avg**	**Max**
Initial RBFn GCV	0.390	1.849	1.286	4.347	0.355	1.027	2.811	9.033
Initial RBFn UEV	0.366	1.738	1.271	4.383	0.310	0.939	2.995	9.046
Initial RBFn FPE	0.390	1.849	1.286	4. 347	0.313	0.938	2.984	8.900
Initial RBFn BIC	0.390	1.849	1.286	4.347	0.330	0.966	2.902	9.000

Table 6. Error comparison of CMY → CIELAB conversion using the regression with different polynomials for the *Training3*, *Training4* and *Test* datasets on the Epson Stylus Pro5000.

Method	**Training3**		**Test**		**Training4**		**Test**	
	Avg	**Max**	**Avg**	**Max**	**Avg**	**Max**	**Avg**	**Max**
Regression polynomials 60 terms	2.386	8.114	3.315	10.674	2.470	9.217	3.603	10.934
Regression polynomials 69 terms	2.328	8.432	3.271	10.863	2.372	9.320	3.731	9.920
Regression polynomials 87 terms	1.771	6.066	2.669	8.644	1.771	5.708	3.271	8.096
Regression polynomials 105 terms	1.556	5.810	2.543	7.501	1.461	5.642	3.497	10.043

Table 7. Initial error measurements of CMY $\rightarrow$ CIELAB conversion using the proposed RBFn for the *Training2* and *Test* datasets on the Epson Stylus Pro5000.

	Training2		**Test**	
	Average	**Maximum**	**Average**	**Maximum**
Proposed RBFn	0.797	1.893	1.831	6.763

Table 8. Subsequent error measurements of CMY $\rightarrow$ CIELAB conversion using the proposed RBFn for the *Training2* and *Test* datasets on the Epson Stylus Pro5000 measured at one month intervals.

Month	**Training2**		**Test**	
	Average	**Maximum**	**Average**	**Maximum**
May	0.927	1.980	2.024	6.833
June	0.744	1.537	2.043	5.780
July	0.784	1.737	2.210	5.699

Table 9. Error comparison of CMY $\rightarrow$ CIELAB conversion using the proposed RBFn and regression with different polynomials for the *Training2* and *Test* datasets on the Epson Stylus Pro5000.

Method	**Training2**		**Test**	
	Average	**Maximum**	**Average**	**Maximum**
Proposed RBFn	0.797	1.893	1.831	6.763
Regression polynomials 60 terms	2.495	7.674	3.228	9.798
Regression polynomials 69 terms	2.458	7.639	3.176	10.111
Regression polynomials 87 terms	1.721	5.453	2.654	8.277
Regression polynomials 105 terms	1.475	4.337	2.571	7.676

Table 10. Error comparison of CMY $\rightarrow$ CIELAB conversion using the proposed RBFn and regression with different polynomials for the *Training2* and *Test* datasets on the Hewlett Packard HP 2000 C.

Method	**Training2**		**Test**	
	Average	**Maximum**	**Average**	**Maximum**
Proposed RBFn	2.876	5.774	3.691	9.800
Regression polynomials 60 terms	6.610	19.141	5.396	18.794
Regression polynomials 69 terms	6.516	19.344	5.728	18.995
Regression polynomials 87 terms	5.182	11.187	5.502	13.077
Regression polynomials 105 terms	4.607	10.714	4.744	13.072

Table 11. Error comparison of CMY $\rightarrow$ CIELAB conversion using the proposed RBFn and regression with different polynomials for the *Training2* and *Test* datasets on the Olivetti Artjet 20.

Method	**Training2**		**Test**	
	Average	**Maximum**	**Average**	**Maximum**
Proposed RBFn	1.752	4.084	2.537	6.660
Regression polynomials 60 terms	1.901	10.268	2.236	9.739
Regression polynomials 69 terms	1.851	10.400	2.266	9.864
Regression polynomials 87 terms	1.514	9.100	1.994	8.664
Regression polynomials 105 terms	1.305	9.100	1.936	8.664

Table 12. Error of CIELAB $\rightarrow$ CMY conversion for the *Test* dataset on the Epson Stylus Pro5000.

Method	**Average**	**Maximum**
Regression with printer LUT (729)	2.244	9.894
Regression with LUT created by proposed RBFn (729)	2.290	8.890

FIGURE CAPTIONS

Figure 1 *Radial Basis Function Network.*

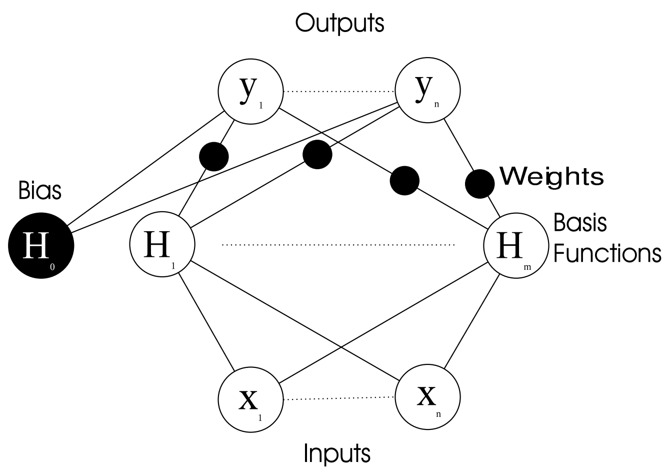

Figure 1. Radial Basis Function Network