Evolutionary Optimized Tensor Product Bernstein Polynomials versus Backpropagation Networks

Günther R. Raidl, Gabriele Kodydek
Institute of Computer Graphics
Vienna University of Technology
Karlsplatz 13/1861, 1040 Vienna, Austria
e-mail: {raidl,kodydek}@eiunix.tuwien.ac.at

Abstract

In this paper a new approach for approximation problems involving only few input and output parameters is presented and compared to traditional Backpropagation Neural Networks (BPNs). The basic model is a Tensor Product Bernstein Polynomial (TPBP) for which suitable control points need to be found. It is shown that a TPBP can also be interpreted as a special class of feed-forward neural networks where control point coordinates are represented by input weights. Although optimal control points for a TPBP leading to the smallest possible approximation errors can be determined by the Method of Least Squares (MLS), this approach has only poor generalization capabilities. Instead, the usage of a (μ, λ)-Evolution Strategy is proposed. Experiments with different sets of test data indicate that the solutions obtained by the TPBP/ES approach generalize very well without exhibiting large approximation errors. When comparing this technique to BPNs, similar approximation and generalization errors were observed. One major advantage of the TPBP/ES approximation model over others such as BPNs is the possibility for humans to better understand a found approximation and to manually post-process it in a very intuitive way to achieve specific changes. Another benefit is that any error function might be used as optimization goal.

Key words: Function Approximation, Tensor Product Bernstein Polynomial, Backpropagation Neural Network

1 Introduction

Backpropagation Neural Networks (BPNs, see [13]) are known to be well suited for many kinds of problems where functions from $\mathbb{R}^m$ to $\mathbb{R}^n$ must be approximated using a given set of samples known in source and destination space. Especially because of their robustness and good generalization abilities, BPNs are often preferable to other techniques as e.g. polynomial regression or trigonometric polynomial regression. On the other hand, a disadvantage of most neural networks is their complex structure: Usually, it is impossible for humans to understand the entire meaning and interaction of certain weights of connections obtained by learning rules. Furthermore, a manual change of these weights to achieve some specific local adaptions or improvements is most of the time too difficult or too dangerous because of unknown side effects.

This article introduces a new approximation technique based on *Tensor Product Bernstein Polynomials* (TPBPs) and *Evolution Strategies* (ESs). Especially when dealing with complex problems containing only a moderate number of input and output parameters, this approach proved to be robust and to generalize very well. In contrast to BPNs, the underlying model enables easy understanding by humans and manual post-processing of automatically generated approximations.

2 Tensor Product Bernstein Polynomials

Bernstein Polynomials and their tensor products are well known in computer graphics due to their usage in Bézier curves, Bézier surfaces, and Free-Form Deformations (FFDs), see e.g. [3, 5, 8, 15, 16].

A tensor product Bernstein polynomial, which involves a mapping from $\mathbb{E}^m$ to $\mathbb{R}^n$, can be defined as

a weighted sum of $(w_1+1)(w_2+1)\ldots(w_m+1)$ control points $\vec{C}_{\vec{i}} = (C_{\vec{i},1}, \ldots, C_{\vec{i},n})^T$, $\vec{i} = (i_1, \ldots, i_m)^T$:

$$TPBP^{w_1,w_2,\ldots,w_m}(\vec{t}) = \sum_{i_1=0}^{w_1} \sum_{i_2=0}^{w_2} \ldots \sum_{i_m=0}^{w_m} \vec{C}_{\vec{i}}\; b_{i_1}^{w_1}(t_1)\; b_{i_2}^{w_2}(t_2) \ldots b_{i_m}^{w_m}(t_m) \quad (1)$$

with the weights $b_i^w(t)$ being the Bernstein polynomials

$$b_i^w(t) = \binom{w}{i} t^i (1-t)^{w-i}. \quad (2)$$

In general, different degrees w_j for different dimensions j $(j = 1, \ldots, m)$ may be useful, but to keep things simple, we assume that $w = w_1 = \ldots = w_m$ and call w the degree of the TPBP.

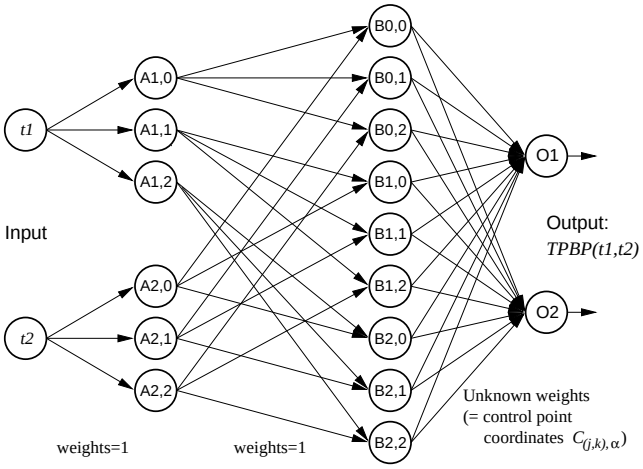
TPBPs are appreciated because of their good properties: The effects of displacing control points are very intuitive, which enables an easily understandable manual design process. Especially Bézier curves ($\mathbb{E} \to \mathbb{R}^2$) and Bézier surfaces ($\mathbb{E}^2 \to \mathbb{R}^3$) have been studied in detail, see [5], and several high quality criteria like the convex hull and variation diminishing properties have been derived. Altogether, they indicate that these curves, surfaces, or general transformation functions are very smooth and do not overshoot between control points. Due to these properties, the general TPBP seems to be an interesting model for general function approximation.

Within Eq. 1 the product of Bernstein polynomials may be substituted by a sum as follows:

$$TPBP^w(\vec{t}) = \sum_{\vec{i}} \vec{C}_{\vec{i}}\; \exp\left(\sum_{j=1}^{m} \ln b_{i_j}^w(t_j) \right). \quad (3)$$

In this way, TPBPs can also be interpreted as a special class of feed-forward neural networks with two hidden layers, see Fig. 1: The first layer calculates logarithms of Bernstein polynomials for all inputs, the next layer determines sums for all possible combinations and applies the exponential function, and the output layer calculates final weighted sums. Note that only the output layer has inputs with unknown weights, which correspond to the control point coordinates of the TPBP. Nodes such as these output nodes are known as *Adalines* (*ada*ptive *li*near *ne*urons), see e.g. [7, 17].

Usually, the goal is to minimize the mean square error $MSE(\mathcal{A})$ for a given sample set $\mathcal{A}$ of pairs of independent values $\vec{s}_k$ in source space and dependent values $\vec{d}_k$ in destination space ($\vec{s}_k = (s_{k,1}, \ldots, s_{k,m})^T$,



Functions of neurons:

$$\mathsf{A}_{\alpha,\beta} = \ln b_\beta^w(t_\alpha), \qquad \alpha \in \{1,2\},$$
$$\mathsf{B}_{\beta,\gamma} = e^{\mathsf{A}_{1,\beta}+\mathsf{A}_{2,\gamma}}, \qquad \beta,\gamma \in \{0,1,2\}$$
$$\mathsf{O}_\alpha = \sum_{i_1=0}^{2} \sum_{i_2=0}^{2} C_{(i_1,i_2),\alpha} \mathsf{B}_{i_1,i_2}$$

Figure 1: Neural network interpretation of a TPBP of order $w = 2$ for $\mathbb{E}^2 \to \mathbb{R}^2$.

$\vec{d}_k = (d_{k,1}, \ldots, d_{k,m})^T$, $k = 1, \ldots, |\mathcal{A}|$, $MSE(\mathcal{A})$ in percent):

$$MSE(\mathcal{A}) = \frac{100\%}{|\mathcal{A}|} \sum_{k=1}^{|\mathcal{A}|} \parallel TPBP^w(\vec{s}_k) - \vec{d}_k \parallel^2 . \quad (4)$$

In this case, optimal input weights of an Adaline (and therefore control point coordinates for TPBPs) can be obtained efficiently by using the well known *Method of Least Squares* (MLS). Although the smallest possible approximation errors $MSE(\mathcal{A})$ can be achieved in this way, that approach has a major drawback: Especially with increasing degrees of the TPBP, resulting approximations tend to oscillate heavily and overshoot between the samples of set $\mathcal{A}$. Therefore, the generalization capabilities are usually only poor.

3 Evolution Strategies

Evolution strategies, which were introduced by H.-P. Schwefel [14] and improved by several other researchers (see e.g. [1, 2, 6]), are generally known to be robust optimization methods. They are especially useful in finding nearly optimal parameters for complex numerical functions. An ES explores the search

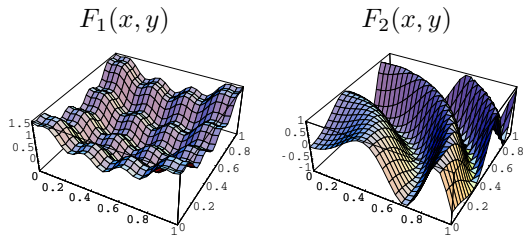
μ (# of parents):	15
λ (# of offsprings):	100
Initial $C_{\vec{i},p}$:	random values $\in [\min\{d_{k,p}\}, \max\{d_{k,p}\}]$, $(p = 1, \ldots, n)$
Initial $\sigma_{\vec{i},p}$:	$\frac{\max\{d_{k,p}\}-\min\{d_{k,p}\}}{2\cdot\sqrt{(w+1)^m}}$ (according to [1])
Fitness function:	$MSE(\mathcal{A})$
Recombination:	$C'_{\vec{i},p} = C^{\mathrm{A}}_{\vec{i},p}$ or $C^{\mathrm{B}}_{\vec{i},p}$ $\sigma'_{\vec{i},p} = (\sigma^{\mathrm{A}}_{\vec{i},p} + \sigma^{\mathrm{B}}_{\vec{i},p})/2$
Mutation:	$C''_{\vec{i},p} = C'_{\vec{i},p} + \mathcal{N}(0, \sigma''_{\vec{i},p})$ $\sigma''_{\vec{i},p} = \sigma'_{\vec{i},p} \cdot e^{\mathcal{N}(0,\tau)}$
Termination cond.:	all $\sigma_{\vec{i},p}$ of best individual $\leq 1\%$ of initial $\sigma_{\vec{i},p}$

Table 1: Some implementation details of the ES.

space efficiently by evaluating samples chosen in a partly stochastic way. This process usually converges to highly fit regions of the search space. For a detailed introduction to ESs see [1, 6].

Already in [10, 11, 18], (μ, λ)-ESs including self-adaption of strategy parameters were used successfully to find suitable control points for three-dimensional space deformations (Free-Form Deformations). In [12] this approach was generalized to TPBPs of arbitrary input and output space dimensions.

Basically, the ES starts with a population consisting of μ randomly generated solutions. Each solution is represented by all its control point coordinates $C_{\vec{i},p}$ $(p = 1, \ldots, n)$, which are randomly chosen out of $[\min\{d_{k,p}\}, \max\{d_{k,p}\}]$. In the next step, these initial solutions are evaluated using $MSE(\mathcal{A})$ as evaluation function. The initial population is then improved in a loop over many generations via the processes of selection, recombination, and mutation until the ES converges and all solutions are close to each other. The creation of a new population for the next generation works as follows: λ offsprings are generated by recombining randomly selected solutions from the parent population two by two. Then these new solutions are slightly modified via a mutation operator, which perturbs each control point coordinate by adding a normally distributed offset with mean 0 and standard deviation $\sigma_{\vec{i},p}$. Note that these strategy parameters $\sigma_{\vec{i},p}$ are also optimized by the ES using the mechanism of self-adaption. For a more detailed discussion of the recombination and mutation operators



$F_1(x,y) = 4\left(\left(x-\frac{1}{2}\right)^2 + \left(y-\frac{1}{2}\right)^2 - \frac{\cos(8x\pi)+\cos(8y\pi)}{5}\right)$
$F_2(x,y) = \sin\left(10x^2 + 10y^2\right)$

Figure 2: Test functions F_1 and F_2 $(\mathbb{E}^2 \to \mathbb{R})$.

including self-adaption see [12]. Finally the newly generated offsprings are evaluated, and the best μ offsprings are selected to form the parent population for the next generation. This process continues until all $\sigma_{\vec{i},p}$ of the best solution so far fall below a certain limit, which indicates that the ES has converged to a small region in the search space.

Applying the ES to the problem of finding control point coordinates for the discussed TPBPs results in solutions with slightly larger $MSE(\mathcal{A})$, but much better generalization capabilities than when using the MLS. Examples in the following section document this behavior. A reason for this effect seems to be that also slightly different solutions in the surroundings of the one obtained by the ES are of very high quality. The global optimum, however, is often located in a way so that solutions which only slightly differ from the optimal solution often approximate significantly worse.

4 Experimental Results

The TPBP/ES approach has been implemented with the characteristics shown in Table 1 and was then compared to the TPBP/MLS and the BPN techniques using various sets of test data derived from several functions. As an example the results of the experiments are discussed here for two functions F_1 and F_2 $(\mathbb{E}^2 \to \mathbb{R})$, depicted in Fig. 2.

A set $\mathcal{A}$ of 100 randomly chosen pairs of values known in source and in destination space was used for approximating each test function with the above mentioned methods. Generalization abilities of obtained final solutions were measured by using a different set $\mathcal{G}$ also made up of 100 samples. The mean square error $MSE(\mathcal{G})$ for sample set $\mathcal{G}$ is, therefore, called generalization error.

TPBP/ES				
	F_1		F_2	
Degree	$MSE(\mathcal{A})$	$MSE(\mathcal{G})$	$MSE(\mathcal{A})$	$MSE(\mathcal{G})$
1	19.36	21.50	34.58	53.65
2	4.28	4.72	34.22	52.64
3	4.17	4.57	26.12	37.24
4	3.89	4.24	22.89	31.24
5	3.39	3.84	11.22	27.17
6	2.27	2.57	6.57	24.36
7	1.96	3.26	3.31	26.20
8	1.82	2.88	2.98	22.84
9	1.51	3.38	3.08	24.74

Table 2: TPBP/ES approach: Results for test functions F_1 and F_2.

TPBP/MLS				
	F_1		F_2	
Deg.	$MSE(\mathcal{A})$	$MSE(\mathcal{G})$	$MSE(\mathcal{A})$	$MSE(\mathcal{G})$
1	19.36	21.42	34.58	53.70
2	4.28	4.86	34.22	52.41
3	4.13	4.92	25.44	77.32
4	3.81	4.42	16.92	2.15e+02
5	3.12	99.89	4.13	4.51e+02
6	1.67	9.81e+02	0.84	7.26e+02
7	1.43	4.77e+03	0.20	8.49e+03
8	0.45	1.61e+05	0.01	5.73e+03
9	7.18e-19	1.39e+08	3.51e-18	2.41e+06

Table 3: TPBP/MLS approach: Results for test functions F_1 and F_2.

Typical approximation errors $MSE(\mathcal{A})$ and generalization errors $MSE(\mathcal{G})$ for the ES and the MLS techniques are listed in Tables 2 and 3 and are graphically depicted for F_1 in Fig. 3. Usually, $MSE(\mathcal{A})$ of a solution obtained by the ES is only slightly worse than that of the MLS, while $MSE(\mathcal{G})$ of the ES-solution is significantly smaller, particularly when solutions of higher degrees are compared.

For the purpose of comparison, traditional backpropagation networks with one and two hidden layers and varying numbers of hidden nodes were implemented using the software package `nn/xnn`, see [9]. Because of the sigmoidal output functions of neurons, output values of the BPNs were linearly scaled so that the interval $[0.2, 0.8]$ was mapped to $[\min\{d_k\}, \max\{d_k\}]$.

The resulting approximation errors $MSE(\mathcal{A})$ and generalization errors $MSE(\mathcal{G})$ for both test functions are shown in Tables 4 and 5 and are for F_1 also depicted in Fig. 4. Both, $MSE(\mathcal{A})$ and $MSE(\mathcal{G})$ obtained by the BPN with only one hidden layer are comparable to those of the TPBP/ES approach.

BPN - one hidden layer				
	F_1		F_2	
H1	$MSE(\mathcal{A})$	$MSE(\mathcal{G})$	$MSE(\mathcal{A})$	$MSE(\mathcal{G})$
1	16.26	17.47	36.97	61.19
2	11.41	11.20	27.75	48.29
3	4.14	3.95	28.41	50.90
4	4.16	4.10	25.23	52.45
5	4.19	4.18	22.65	46.24
6	4.10	3.81	28.51	57.08
7	2.81	3.10	27.02	52.60
8	3.86	4.17	22.88	47.29
9	4.36	4.35	23.37	50.15
10	4.88	4.86	23.16	44.81
11	2.59	3.06	13.13	38.84
12	4.42	4.50	19.26	46.54
13	2.34	2.94	6.39	20.43
14	2.41	2.59	19.92	45.53
15	4.31	4.29	23.74	47.51
16	5.03	4.73	4.85	20.42
17	5.14	4.78	8.92	22.33
18	2.01	2.13	21.36	53.58
19	2.20	2.47	8.41	22.90
20	2.12	2.44	2.04	15.43

Table 4: BPNs containing one hidden layer with varying numbers of nodes (H1): Results for test functions F_1 and F_2.

BPN - two hidden layers				
	F_1		F_2	
H1/H2	$MSE(\mathcal{A})$	$MSE(\mathcal{G})$	$MSE(\mathcal{A})$	$MSE(\mathcal{G})$
2/2	20.26	20.86	40.10	67.46
2/3	20.33	21.07	42.25	71.54
2/4	20.49	21.39	44.58	74.97
2/5	8.17	10.87	21.78	54.59
3/2	20.26	20.86	37.05	59.02
3/3	4.28	4.04	27.67	45.72
3/4	13.06	15.17	44.62	75.12
3/5	20.74	21.81	46.35	78.37
4/2	4.88	5.19	27.59	53.11
4/3	20.33	21.07	19.63	39.50
4/4	4.38	5.01	37.87	65.41
4/5	20.74	21.81	44.01	72.93
5/2	3.60	4.19	16.04	37.96
5/3	20.33	21.07	42.29	72.99
5/4	20.49	21.39	15.06	39.43
5/5	18.45	17.54	31.25	50.18

Table 5: BPNs containing two hidden layers with varying numbers of nodes (H1, H2): Results for test functions F_1 and F_2.

Sometimes the BPN approximates and generalizes slightly better, but especially when using two layers of hidden nodes the results are not as good de-

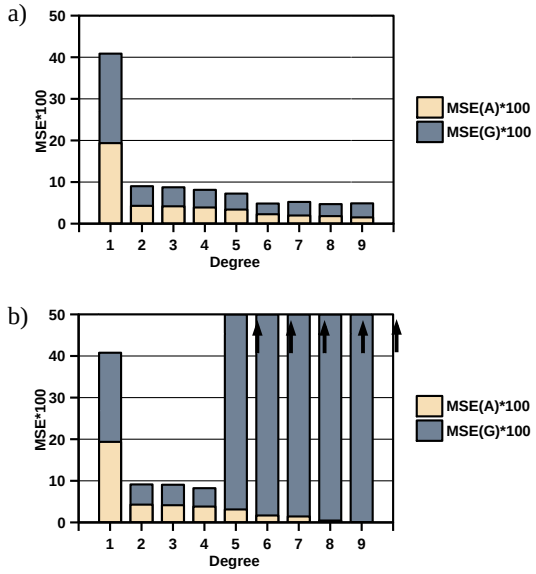


Figure 3: Results for test function F_1: (a) TPBP/ES and (b) TPBP/MLS.

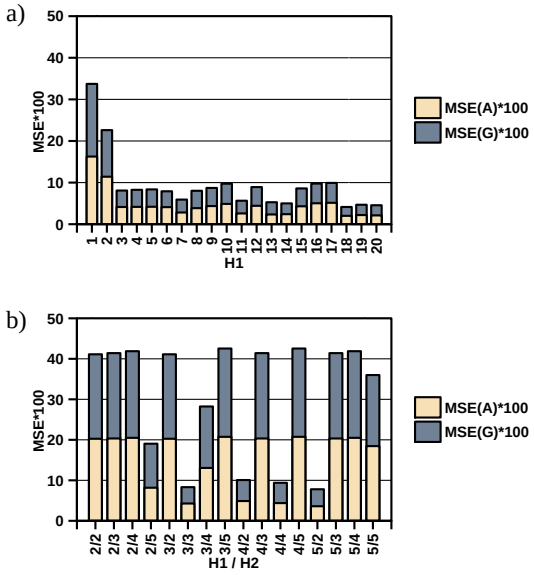


Figure 4: Results for test function F_1: BPNs containing (a) one hidden layer and (b) two hidden layers with varying numbers of nodes (H1, H2).

spite the large numbers of performed iterations (from 18000 to 60000 depending on the number of nodes in the hidden layers).

A drawback of the TPBP/ES approach is its higher computational effort. The performed experiments using the new method took between a few seconds up to 2.5 hours[1] depending on the degree of the TPBP. The backpropagation learning rule needed up to 35 minutes for the most complex neural nets, but may even be sped up by using more sophisticated techniques as e.g. Quickprop, see [4]. Also, the probability for finding well approximating solutions with BPNs containing two layers might be increased in various ways.

5 Conclusion

Many experiments indicate that the TPBP/ES approach is a method well suited for complicated approximation problems involving only few input and output variables. Although the ES does not usually converge to the globally optimal solution as it can be determined by the MLS when using the mean square error, the ES solutions are much better concerning generalization. The approximation and generalization errors are comparable to those of various BPNs. The major advantage of the TPBP/ES approach over many other approximation techniques as BPNs is the possibility for humans to understand the meaning and effects of found control point positions and, therefore, to adapt or improve solutions manually in a very intuitive way. Another advantage is that any error function as e.g. the general Minkowski metric might be used instead of only the mean square error.

[1] A Pentium 133MHz PC, Linux, and GNU C++ were used.

References

[1] Bäck, T.: *Evolutionary Algorithms in Theory and Practice*, Oxford University Press, New York, 1996.

[2] Bäck, T., Fogel, D. B., Michalewicz, Z.: *Handbook of Evolutionary Computation*, Oxford University Press, New York, 1997.

[3] Coquillart S., Jancene, P.: Animated Free-Form Deformation: An Interactive Animation Technique, *Computer Graphics (SIGGRAPH '91 Proceedings) 25*, pp. 23–26, 1991.

[4] Fahlman, S. E.: Faster-Learning Variations on Back-Propagation: An Empirical Study, *Proceedings of the 1988 Connectionist Models Sum-*

mer School, Morgan Kaufmann, pp. 38–51, 1989.

[5] Farin, G.: *Curves and Surfaces for Computer Aided Geometric Design*, Academic Press, CA, 1990.

[6] Fogel, D. B.: *Evolutionary Computation – Toward a New Philosophy of Machine Intelligence*, IEEE Press, NJ, 1995.

[7] Freeman, J. A., Skapura, D. M.: *Neural Networks – Algorithms, Applications, and Programming Techniques*, Addison Wesley, MA, 1991.

[8] Lampinen, J.: Choosing a Shape Representation Method for Optimization of 2D Shapes by Genetic Algorithms, *Proceedings of the 3rd Nordic Workshop on Genetic Algorithms and their Applications*, Helsinki, Finland, pp. 305–319, August 1997.

[9] Neureka Artificial Neural Systems: *nn/xnn – Developing and Simulating Artificial Neural Networks*, see `http://www.bgif.no/neureka` (free software), 1993–1997.

[10] Raidl, G., Tastl, I.: Finding a Perceptual Uniform Color Space with Evolution Strategies, *Proceedings of the 4th IEEE Conference on Evolutionary Computation*, Indianapolis, IN, pp. 513–517, April 1997.

[11] Raidl, G., Tastl, I., Wurm, C.: Automated Generation of Free-Form Deformations by Using Evolution Strategies, *Proceedings of the 6th International Workshop on Digital Image Processing and Computer Graphics*, Vienna, Austria, pp. 177–184, October 1997.

[12] Raidl, G., Wurm, C.: Approximation with Evolutionary Optimized Tensor Product Bernstein Polynomials, *Proceedings of the International Conference on Artificial Intelligence in Industry*, High Tatras, Slovakia, pp. 247–256, April 1998.

[13] Rumelhart, D. E., Hinton, G. E., Williams, R. J.: Learning Internal Representations by Error Propagation, *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, Vol. 1, Cambridge, The MIT Press, pp. 318–362, 1986.

[14] Schwefel, H.-P.: *Numerical Optimization of Computer Models*, Wiley, UK, 1981.

[15] Sederberg, T. W., Parry, S. R.: Free-Form Deformation of Solid Geometric Models, *Computer Graphics (SIGGRAPH '86 Proceedings) 20*, pp. 151–160, 1986.

[16] Watt, A., Watt, M.: *Advanced Animation and Rendering Techniques*, ACM press, Addison-Wesley, UK, 1992.

[17] Widrow, B., Stearns, S. D.: *Adaptive Signal Processing*, Signal Processing Series, Prentice-Hall, NJ, 1985.

[18] Wurm, C.: *Suche von Farbraumtransformationen zwischen Scanner-, Bildschirm- und Drukker-Farbräumen*, MSc Thesis, Institute of Computer Graphics, Vienna University of Technology, Austria, 1997.