Further Accelerating the Inversion of the Yule-Nielson Modified Neugebauer Model

Changjun Li and M Ronnier Luo; University of Leeds, Leeds LS2 9JT, UK)

Abstract

Urban and Grigat [*Color research and Application 31: pages 229-238*] *proposed an iterative method for the inversion of the Cellular Yule-Nielson modified Neugebauer model. The method involves matrix and vector multiplications per iteration. Urban et al [CIC15, pages 178-183, 2007] reported that the computational costs per iteration could be reduced by introducing the singular value decomposition of the matrix. In this paper, "QR" decomposition for the matrix is introduced. The matrix "Q" is an orthogonal matrix and "R" represents an upper triangular matrix. Using this kind of decomposition, an upper triangular matrix is involved per iteration; hence the computational cost can be further reduced comparing with the work of Urban et al (2007).*

Introduction

The Neugebauer model [1] is one of the most basic tools for modelling colour printing systems. For using this model for 3-ink (Cyan (c), Magenta (m), and Yellow (y)) printer, eight Neugebauer primaries must be measured. Let $R_i(\lambda)$, $i = 1,2,\cdots,8$, be the reflectances measured from the corresponding primary colours. The names of the eight primary colours, combination of inks, and the reflectance functions are listed in the first, second, and third columns respectively of Table 1. For any combination of inks c , m , and y , the resulting colour's reflectance on the printer is given by equation (1):

$$R(\lambda) = \sum_{i=1}^{8} a_i R_i(\lambda) \qquad (1)$$

which is the classical Neugebauer equation. The coefficients a_i are defined in the last column of Table 1.

One of the most improvements over the classical Neugebauer model is the Cellular Neugebauer model proposed by Heuberger et al [2]. A three-level Cellular Neugebauer model is shown in Figure 1. There are eight cells and 27 measurements must be made. Suppose we divide the whole space into N levels in each ink direction and let the corresponding ink densities denote by

$$c_0, c_1, \cdots, c_N \,,\ \ m_0, m_1, \cdots, m_N \,,\ \ y_0, y_1, \cdots, y_N$$

in ascending order for each ink. For each of the given cyan, magenta, and yellow ink values: c, m, and y , the cell containing c , m , and y must be found. Suppose the cell is represented by:

$$[c_i, c_{i+1}; m_j, m_{j+1}; y_k, y_{k+1}]$$

Table 1: Eight primaries, ink combinations, reflectances of the primaries, and coefficients for predicting reflectance using Neugebauer model for a given ink combination (c,m,y)

Primary	Ink Combination (*c,m,y*)	Reflectance	Coefficient
White	(*0,0,0*)	$R_1(\lambda)$	$a_1 = (1-c)(1-m)(1-y)$
Cyan	(*1,0,0*)	$R_2(\lambda)$	$a_2 = c(1-m)(1-y)$
Magenta	(*0,1,0*)	$R_3(\lambda)$	$a_3 = (1-c)m(1-y)$
Yellow	(*0,0,1*)	$R_4(\lambda)$	$a_4 = (1-c)(1-m)y$
Red	(*0,1,1*)	$R_5(\lambda)$	$a_5 = (1-c)my$
Green	(*1,0,1*)	$R_6(\lambda)$	$a_6 = c(1-m)y$
Blue	(*1,1,0*)	$R_7(\lambda)$	$a_7 = cm(1-y)$
Black	(*1,1,1*)	$R_8(\lambda)$	$a_8 = cmy$

Then, c' , m' , and y' can be obtained by using the following transformations

$$c' = \frac{c - c_i}{c_{i+1} - c_i}, \quad m' = \frac{m - m_j}{m_{j+1} - m_j}, \quad y' = \frac{y - y_k}{y_{k+1} - y_k} \qquad (2)$$

Now the eight coefficients can be computed using the formulae in Table 1 with c' , m' , and y' values. The eight reflectance functions of the primary colours now correspond to the corners of the cell. As soon as the eight coefficients and reflectance functions are found, the predicted reflectance $R(\lambda)$ corresponding to the given cyan, magenta, and yellow ink values: c , m , and y can be computed using equation (1).

The Yule-Nielsen effect [3] can be incorporated into the Cellular Neugebauer Model to reflect the scattering behavior of photons penetrating into the substrate, which can be simply realized by adding one parameter n into equation (1). The new equation now becomes:

$$R(\lambda) = [\sum_{i=1}^{8} a_i (R_i(\lambda))^{1/n}]^n \qquad (3)$$

The Cellular Neugebauer Model with equation (3) rather than equation (1) is called the Cellular Yule-Nielson modified spectral Neugebauer (CYNSN) model in references [4,5]. The parameter n will also affect the performance of the CYNSN model, which can be obtained using optimization with certain training data.

Theoretical consideration can be fund from other publications [6-8].

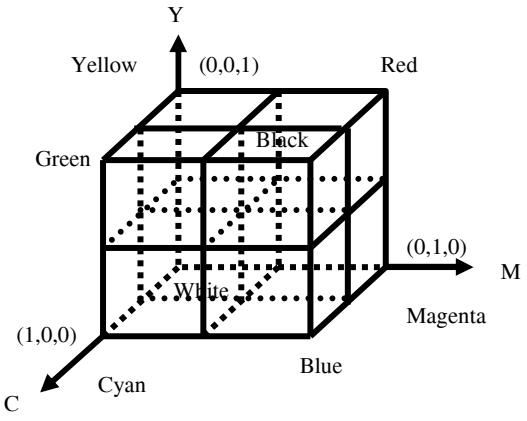

***Figure 1**. Cell illustration for three-level Cellular Neugebauer Model*

The Inversion of the CYNSN Model

Urban and Grigat [4] used the cellular Yule-Nielson modified Neugebauer (CYNSN) model for the spectral separation of multispectral images. Let A be defined by:

$$A=[(R_1)^{1/n},(R_2)^{1/n},(R_3)^{1/n},(R_4)^{1/n},(R_5)^{1/n},(R_6)^{1/n},(R_7)^{1/n},(R_8)^{1/n}] \quad (4)$$

where R_i are the reflectance functions (column vectors) of the eight corners of a cell and $(R_i)^{(1/n)}$ is evaluated at each of the components of column vector R_i. Note that if the reflectance function is sampled at 10nm interval between 400nm and 700nm, then A is a matrix of size 31×8. Let also all the coefficients in equation (3) form a column vector *a*, i.e.,

$$a^T=(a_1,a_2,a_3,a_4,a_5,a_6,a_7,a_8) \quad (5)$$

Here T is the transpose of a vector or matrix. If we let R be a reflectance function (column vector), then the inverse CYNSN model is expressed by minimising the function $f(c,m,y)$ defined by equation (6):

$$f(c,m,y)=\| Aa-r \|_2^2 \quad \text{with} \quad r=(R)^{(1/n)} \quad (6)$$

Here, for any column vector p

$$\| p \|_2^2 = p^T p = \sum_{i=1}^{n} p_i^2$$

Urban and Grigat [4] developed an algorithm for the inversion of the CYNSN. The algorithm can be summarised below:

Urban and Grigat (UG) Algorithm (2006): (applied to $f(c,m,y)=\| Aa-r \|_2^2$)

Choose initial: $x^{(0)}=(c^{(0)},m^{(0)},y^{(0)})^T=(0.5,0.5,0.5)^T$, the error tolerance τ (a small positive number, $\tau=10^{-5}$ for example), and a large integer $k_{\max}$

For k=1,2, … till $k_{\max}$ do the following two steps:

Step 1:

$$c^{(k)}=\frac{(A\frac{\partial a(c^{(k-1)},m^{(k-1)},y^{(k-1)})}{\partial c})^T(r-Aa_{my}(c^{(k-1)},m^{(k-1)},y^{(k-1)}))}{(A\frac{\partial a(c^{(k-1)},m^{(k-1)},y^{(k-1)})}{\partial c})^T(A\frac{\partial a(c^{(k-1)},m^{(k-1)},y^{(k-1)})}{\partial c})}$$

$$m^{(k)}=\frac{(A\frac{\partial a(c^{(k)},m^{(k-1)},y^{(k-1)})}{\partial m})^T(r-Aa_{cy}(c^{(k)},m^{(k-1)},y^{(k-1)}))}{(A\frac{\partial a(c^{(k)},m^{(k-1)},y^{(k-1)})}{\partial m})^T(A\frac{\partial a(c^{(k)},m^{(k-1)},y^{(k-1)})}{\partial m})}$$

$$y^{(k)}=\frac{(A\frac{\partial a(c^{(k)},m^{(k)},y^{(k-1)})}{\partial y})^T(r-Aa_{cm}(c^{(k)},m^{(k)},y^{(k-1)}))}{(A\frac{\partial a(c^{(k)},m^{(k)},y^{(k-1)})}{\partial y})^T(A\frac{\partial a(c^{(k)},m^{(k)},y^{(k-1)})}{\partial y})}$$

Step 2:

If $|f(x^{(k)})-f(x^{(k-1)})|<\tau(1+f(x^{(k)}))$ and $\| x^{(k)}-x^{(k-1)} \|_2\le\sqrt{\tau}(1+\| x^{(k)} \|_2)$ stop and accept $x^{(k)}=(c^{(k)},m^{(k)},y^{(k)})^T$ as the approximate solution.

The above algorithm deserves some notes.

Note 1: $\frac{\partial a(c^{(k-1)},m^{(k-1)},y^{(k-1)})}{\partial c}$ is $\frac{\partial a}{\partial c}$ evaluated at $(c^{(k-1)},m^{(k-1)},y^{(k-1)})$, and others have the similar meanings.

Besides, $a_{my}=a-c\frac{\partial a}{\partial c}$, $a_{cy}=a-m\frac{\partial a}{\partial m}$, $a_{cm}=a-y\frac{\partial a}{\partial y}$

In addition, it can be found that a_{my} is part of $\frac{\partial a}{\partial c}$, a_{cy} is part of $\frac{\partial a}{\partial m}$, and a_{cm} is part of $\frac{\partial a}{\partial y}$. Hence, computing a_{my}, a_{cy} and a_{cm} need no extra cost.

Note 2: when computing $c^{(k)}$ in Step 1, if it is greater than 1, it is set to 1, and if it is less then 0, it is set to 0. This is true for computing $m^{(k)}$ and $y^{(k)}$ as well.

Note 3: computing the matrix and vector product $Aa_{my}(c^{(k-1)},m^{(k-1)},y^{(k-1)})$ needs no extra computational work since it is part of the computational work for computing $A\frac{\partial a(c^{(k-1)},m^{(k-1)},y^{(k-1)})}{\partial c}$. Thus, for computing $c^{(k)}$, one vector evaluation ($\frac{\partial a(c^{(k-1)},m^{(k-1)},y^{(k-1)})}{\partial c}$), one matrix and vector multiplication and two inner products of vectors are involved. Computing $m^{(k)}$ or $y^{(k)}$ has the exact same cost as computing $c^{(k)}$. Thus, if we let A be size of L by 8, then number of multiplications per iteration are:

M1=3[4 (vector evaluation) + 8L (matrix*vector) + 2L (inner product) +1] (7)

Note 4: Urban et al [5] further accelerated the above algorithm by using the singular value decomposition [9] of the matrix A. In fact, Let

$$A = UDV^T$$

where U (or V) is an orthogonal matrix and D is a matrix having the same size as A and being zero everywhere except at diagonals which are the singular values of the matrix A. Thus,

$$f(c,m,y) = \| Aa - r \|_2^2 = \| DV^T a - U^T r \|_2^2$$

Let

$$Q^T r = \begin{pmatrix} b_q \\ p \end{pmatrix}, \; DV^T = \begin{pmatrix} B_q \\ B \end{pmatrix}$$

where b_q is a column vector having q components and B_q is a matrix having q rows. Urban et al [5] pointed out that normally matrix B is close to a zero matrix, hence we have

$$\begin{aligned} f(c,m,y) &= \| Aa - r \|_2^2 = \| B_q a - b_q \|_2^2 + \| Ba - p \|_2^2 \\ &\approx \| B_q a - b_q \|_2^2 + \| p \|_2^2 \end{aligned} \quad (8)$$

Thus, minimising function $f(c,m,y) = \| Aa - r \|_2^2$ is approximately equivalent to minimise

$$\phi(c,m,y) = \| B_q a - b_q \|_2^2 \quad (9)$$

since $\| p \|_2^2$ is independent of the inks . Hence applying the UG algorithm to $\phi(c,m,y) = \| B_q a - b_q \|_2^2$ will decrease significantly computational costs per iteration compared with applying the UG algorithm to $f(c,m,y) = \| Aa - r \|_2^2$ if q is set to be small compared with L. Detail analysis for more than 3 inks was given in [5]. Note also that for 3-inks, q should not be larger than 8, and for 4-inks, q should not be larger than 16. For 3-inks, the amount of multiplications per iteration in this case is given by:

M2=3[4 (vector evaluation) + q*L (matrix*vector) + 2q (inner product) +1] (10)

Further Acceleration of UG Algorithm by QR Decomposition

As noted above, Urban et al [5] demonstrated that the inversion of the CYNSN model using the UG algorithm with the singular value decomposition of the matrix A can be accelerated. Now we want to show if we use 'QR' decomposition for the matrix A, further gain can be obtained. The so-called QR decomposition [9] is that the matrix A can be expressed as: A=QS, where Q is an orthogonal matrix of size L by L and S is an 'upper triangular' matrix of size L by 8 having the following form:

$$S = \begin{pmatrix} U \\ O \end{pmatrix} \text{ with } U = \begin{pmatrix} x & x & x & x & x & x & x & x \\ & x & x & x & x & x & x & x \\ & & x & x & x & x & x & x \\ & & & x & x & x & x & x \\ & & & & x & x & x & x \\ & & & & & x & x & x \\ & & & & & & x & x \\ & & & & & & & x \end{pmatrix} \quad (11)$$

Here U is an 8 by 8 upper triangular matrix, x represents non-zero elements, and O is an (L–8) by 8 matrix.

Note that with 4-ink printer, the above matrix U is a size of 16 by 16 and O is a size of (L–16) by 16. When a printer has more than 4 primary inks, see for example has k inks, then 2^k is normally greater than L. In this case, matrix S is simply the matrix U which is a size of L by 2^k , and has the following form:

$$U = \begin{pmatrix} x & x & \cdots & \cdots & x \\ & x & \text{x} & \cdots & x \\ & & \ddots \; \ddots & \cdots & x \\ & & x \;\; \text{x} \; \cdots & x \end{pmatrix}$$

Setting

$$Q^T r = \begin{pmatrix} b \\ p \end{pmatrix}, \quad (12)$$

with b being a column vector of 8 components results in:

$$\begin{aligned} f(c,m,y) &= \| Aa - r \|_2^2 = \| \begin{pmatrix} U \\ O \end{pmatrix} a - Q^T r \|_2^2 \\ &= \| Ua - b \|_2^2 + \| p \|_2^2 \end{aligned} . \quad (13)$$

Note that p is independent of *c*, *m*, and *y*. Thus, minimisation of the function $f(c,m,y)$ is equivalent to the minimisation of

$$\varphi(c,m,y) = \| Ua - b \|_2^2 \quad (14)$$

Therefore, applying the UG algorithm to $\varphi(c,m,y) = \| Ua - b \|_2^2$ will decrease significantly computational costs per iteration compared with applying the UG algorithm to $f(c,m,y) = \| Aa - r \|_2^2$. In this case, the amount of multiplications per iteration is given by

M3=3[4 (vector evaluation) + 4*9 (matrix*vector) + 2*8 (inner product) +1] (15)

If we let L=31, and q=8. Thus, the UG algorithm applied to $f(c,m,y) = \| Aa - r \|_2^2$ needs M1=945 multiplications, the UG algorithm applied to $\phi(c,m,y) = \| B_q a - b_q \|_2^2$ (Urban et al [5]) needs M2=255 multiplications, and the UG algorithm applied to $\varphi(c,m,y) = \| Ua - b \|_2^2$ needs M3=171 multiplications. Thus, the current approach saves 82 percent compared with the original UG algorithm, and 33 percent compared with the approach of Urban et al [5].

One may argue that the UG algorithm with SVD decomposition may have less computational costs per iteration by choosing smaller q. Yes, this is true if q is small. Table 2 shows the amounts of multiplications (M2) for this approach with q=1,2, till 8. As noted before, there is no point to choose q greater than 8 for 3-ink printer. M1 represents the amount of multiplications for the original UG algorithm and M3 for the UG algorithm with QR decomposition of A. However, they are not affected by the choices of q. It can be seen that from Table 2, when q is less than or equal to 5, the UG algorithm with SVD decomposition will be faster than the current approach. However, we have to note that this is at the expense of loosing accuracy. In fact, with our 3-ink printer, it is fund that for a typical cell, if we let $A = UDV^T$ and $H = DV^T$, the elements in the first row of H have magnitudes of x.xxxx, here x represents non zero value; elements in the second

to fourth rows have magnitudes of 0.xxxx; and elements in the fifth to eighth rows having magnitudes of 0.0xxx. Thus, using q less than 8 will definitely lose accuracy. But the current approach does not loss accuracy. On the other hand, the elements of the matrix U in equation (11) for the QR decomposition have magnitudes less than those of the matrix $H = DV^T$. Hence, the q rows of U (hence b in equation (14)) can be used as well if the accuracy is not important.

Table 2 also shows the ratios: $M3/M1$ and $M3/M2$. When q=8, the ratios are 0.18 and 0.67 respectively, which show the current approach only takes 18 percent of the original UG algorithm and 67 percent of the UG algorithm with SVD decomposition.

Table 2: Amounts of multiplications for the original UG algorithm (M1), the UG algorithm with SVD decomposition (M2), and with QR decomposition (M3) for q=1 to 8 respectively for 3-ink printer, and the ratios: $M3/M1$ and $M3/M2$.

q	M1	M2	M3	M3/M1	M3/M2
1	945	45	171	0.18	3.8
2	945	75	171	0.18	2.28
3	945	105	171	0.18	1.63
4	945	135	171	0.18	1.27
5	945	165	171	0.18	1.04
6	945	195	171	0.18	0.88
7	945	225	171	0.18	0.76
8	945	255	171	0.18	0.67

Table 3: Amounts of multiplications for the original UG algorithm (M1), the UG algorithm with SVD decomposition (M2), and with QR decomposition (M3) for q=1 to 16 respectively for 4-ink printer, and the ratios: $M3/M1$ and $M3/M2$.

q	M1	M2	M3	M3/M1	M3/M2
1	1725	105	555	0.32	5.29
2	1725	159	555	0.32	3.49
3	1725	213	555	0.32	2.61
4	1725	267	555	0.32	2.08
5	1725	321	555	0.32	1.73
6	1725	375	555	0.32	1.48
7	1725	429	555	0.32	1.29
8	1725	483	555	0.32	1.15
9	1725	537	555	0.32	1.03
10	1725	591	555	0.32	0.94
11	1725	645	555	0.32	0.86
12	1725	699	555	0.32	0.79
13	1725	753	555	0.32	0.74
14	1725	807	555	0.32	0.69
15	1725	861	555	0.32	0.64
16	1725	915	555	0.32	0.61

Table 3 shows details about 4-ink printer. The current approach will save 68 percent computational work per iteration compared with the original UG algorithm. For q =16, the current approach will save 39 percent computational work per iteration compared with the UG algorithm with SVD decomposition. For q less than or equal to 9, the UG algorithm with SVD decomposition will be faster than the current approach at the expense of losing accuracy. On the other hand, the current approach can also use q rows when accuracy is not an important factor.

The current approach only uses 58% of the amount of computational work of the original UG algorithm for 5-ink printer and 79% for the 6-ink printer.

Conclusions

The original UG algorithm [4] for the inversion of the CYNSN model was improved using the QR decomposition. It is shown that the current approach only uses 18, 32, 58, and 79 percents of the original UG computational work for 3-, 4-, 5- and 6-ink printers respectively. Unlike the approach of Urban et al [5] using SVD decomposition, the current approach does not sacrifice any accuracy for achieving the acceleration.

When comparing the current approach with that of Urban et al [5], it is slightly more complicated since the latter achieves the acceleration at the expense of losing accuracy. Even at the expense

of losing accuracy the current approach is still better than their approach in certain cases. However, the current approach can also use fewer rows from corresponding matrix and vector so that further saving can be achieved. Thus if both methods use the same number of rows, the current approach is always better.

References

[1] HEJ Neugebauer, Die theoretischen Grundlagen des Mehrfarbenbuchdrucks. Zeit wissenschaft. Photogr. Photophys. Photochem. [reprinted in 27, translated in 28] 1937; 36: 73-89.

[2] KJ Heuberger, ZM Jing, and S Persiev, Color transformation and lookup tables, TAGA/ISCC Proc; 1992, pages 863-881.

[3] JAC Yule and WJ Nielson, The penetration of light into paper and its effect on halftone reproduction, J Tech Assoc Graphic Arts 1951; 4: 65-76.

[4] P Urban and RR Grigat, Spectral based color separation using linear regression iteration, Color Research and Application; 31, 229-238, 2005.

[5] P Urban, MR Rosen and RS Berns, Fast spectral based separation of multispectral images, IS&T/SID: CIC15, pages 178-183, Albuquerque, New Mexico, USA, 2007.

[6] JS Arney, A probability description of the Yule-Nielson effect, J Imaging Science and Technology 1997; 41: 633-636.

[7] JS Arney and M Katsube, A probability description of the Yule-Nielson effect, II, J Imaging Science and Technology 1997; 41: 637-642.

[8] JS Arney and S Yamaguchi, The physics behind the Yule-Nielson equation, Savannah, GA: IS&T: PICS; 1999, pages 381-385.

[9] GH Golub and CF Van Loan, Matrix Computations, 3rd ed , Johns Hopkins University Press, 1996.