Calculation of Scalars in Neugebauer-Like Models. I: Refactoring the Calculations

J A Stephen Viggiano
School of Photographic Arts and Sciences
Rochester Institute of Technology

Abstract

This paper discusses a new way to compute weights (scalars) for Neugebauer-like models that is more flexible than existing methods, simplifying the insertion of a new model for scalar calculation. Specifically, the scalar computing task is refactored into two independent components. In one, the specific dot overlap behavior is specified in a single, often simple, expression. This expression may be implemented for each the three main overlap modalities in a function with a single-statement body. The other component actually computes the scalars, calling the other component as necessary. This second component has been described algorithmically, and open-source software to demonstrate it has been made available.

Introduction

Models of halftone reflectance have many applications, including construction of profiles in color management, analysis of sensitivity of color attributes to variables including ink amount, process control, colorant and substrate selection, colorant level optimization, Pareto analysis of error sources, simulation, and, among others, simply "for the science of it."

Some background is provided in the remainder of this section and the following section. Motivation for the study is discussed in Section . Specialized definitions are provided in Section . Sections and disclose the new computation method for the two simplest cases of two and three colorants. In Section , companion software to demonstrate the new technique is introduced; details of numerical tests are provided. Finally, conclusions and a description of ongoing and future work are offered in Section .

What is this paper about?

In order to implement most models[1] for halftone color prediction with i inks or colorants, one must compute the weights (mathematically, *scalars* in a vector space) for each of the 2^i primaries. This computation depends on the specific manner the dots of different colorants overlap each other.

While there are many different ways the dots may overlap each other, there are only three that have received more than cursory attention:

1. Independent overlap, as for traditionally angled clustered dots and independent random screening, as modeled by the Demichel equations; [3, 4, 5]

2. Dot-on-dot, where overlap is maximized; and

3. Dot-off-dot, where overlap is minimized, and dots of two colorants do not overlap until their combined relative area coverage is unity.

Calculating the scalars for any of these (or other models) under the current state of the art requires coding of $p = 2^i$ different expressions; the number of expressions literally grows exponentially with the number of colorants. This paper shows how the implementation can be split into two different tasks, the first dependent solely on the number of colorants, the second containing the specifics of the dot overlap behavior. Companion software is provided that generates code to implement the first component; the second may be, for any of the three models enumerated above, a simple function with a one-line body. For the dot-on-dot model, the function in Python may be:

```
def dot_on_dot (colorant_amounts):
    return min (colorant_amounts)
```

This contrasts sharply with the 8! = 40320 cases that need to be considered in a state-of-the-art implementation for eight colorants.

The dramatic reduction in complexity is meritorious in its own right; papers to follow this one will discuss the class of functions that may be used for dot overlap models, permitting greater halftone color model flexibility, and, as a consequence, greater accuracy and robustness.

Objectives

The primary goal of this investigation, as a whole, is to enable greater accuracy and robustness (scenarios for which the assumptions are satisfied to a practical extent) to models for halftone color. Specifically, the aim is to both enable models for dot overlap other than those traditionally used, and to identify an entire category of functions that may be used for this purpose. Combined, these can open an all-but-neglected dimension for improvement of model accuracy.

This paper, the first part of a three-part series, presents a novel method for calculating the scalars ("area fractions") common to many models of halftone color. In software engineering, *refactoring* denotes…

> …the process of changing a software system in such a way that it does not alter the external behavior of the code yet it improves its internal structure. It is a disciplined way to clean up code that minimizes the chances of introducing bugs. [6]

The refactoring described in this paper permits the scalars to be computed as simple sums and differences of a single function, enabling, in turn, a fast, uncomplicated, and sure way to account

[1] A counter-example is Gustavson.[1, 2].

for different modes of dot overlap beyond the traditional angled screens (Demichel), dot-on-dot, and dot-off-dot, all of whose conditions may easily be violated in practice. When changing the model for dot overlap, only a single, simple, function need be changed, rather than an entire block of sometimes complex code.

One may wonder why the refactoring is being offered prior to discussing the class of functions that govern halftone dot overlap. Software engineering expert Fowler offers the following wisdom:

> When you find you have to add a new feature to a program, and the program's code is not structured in a convenient way to add the new feature, first refactor the program to make it easier to add the new feature, then add the new feature. [7]

The refactoring is described in this paper. The later papers will describe the "new features" (alternate models for dot overlap) and strategies for using them.

Neugebauer-Like Models

In this paper, a *Neugebauer-like model* for the color produced by a hard-copy process may be written:

$$\phi\left(\vec{\beta}\right) = \sum_{j=1}^{p} a_j \cdot \phi\left(\vec{\beta}_j\right) \tag{1}$$

where $\phi : \mathbb{R}^+ \rightarrow \mathbb{R}$ is a *bijection*, an invertible function; it maps non-negative real numbers to real numbers;

the $\vec{\beta}_j$ are radiance factor spectra of *Neugebauer primaries* (henceforth, simply "primaries");

the a_j are the area fractions occupied by each primary;

p is the number of primaries; and

$\vec{\beta}$ is the spatially-averaged radiance factor spectrum of the resulting halftone pattern.

Because ϕ is invertible, the radiance factor spectrum of the colored pattern may be written explicitly:

$$\vec{\beta} = \phi^{-1}\left[\sum_{j=1}^{p} a_j \cdot \phi\left(\vec{\beta}_j\right)\right] \tag{2}$$

Mathematically, Eq (1) may be described as a convex sum in a vector space, with the role of the vectors played by each $\phi(\vec{\beta})$, and the scalars by the area fractions a_j. The scalars are computed from the colorant amounts; these calculations depend on the specific manner the dots of different colorants overlap each other.

Note that Eq (2) reduces to Neugebauer's [8] when the bijection ϕ is the identity function and the scalars are computed according to Demichel's [3, 4, 5] formulae. Another special case for (2) occurs when ϕ is a power function (the reciprocal of the exponent is referred to as the *Yule-Nielsen* [9] n *value*) and the Demichel equations are used. [10, 11, 12]

A third special case is contained in Balasubramanian, [13] who provided a model for halftone color when using so-called *dot-on-dot* screening by changing the calculation of the scalars. Specifics of scalar calculation under Demichel's and Balasubramanian's models for two colorants will be covered below.

Neugebauer-type models have been applied to varied applications, including printing on ceramics, [14] two-sided printing for backlit displays, [15, 16] and printing with metallic inks. [17]

Generalizations of Neugebauer-like models

Heuberger, et alii, [18] suggested the *cellular* model, a popular generalization of Neugebauer's original model ($\phi(\beta) = \beta$, the identity function). Each colorant axis is partitioned into two or more sub-axes (so the unit cube (or hypercube) is partitioned into a plurality of subspaces, each a rectangular (hyper) parallelepiped); the colorant amounts are normalized to [0, 1] within each partition, and Neugebauer's model is applied within each. This requires additional primaries; for example, if each of $i = 4$ colorant axes are partitioned into two sub-axes, there will be 81 primaries ($(2+1)^4$) rather than the 16 required by the non-cellular approach with the same number of colorants. Heuberger and his co-authors suggested printing a characterization target with these additional "virtual" primaries.

Balasubramanian [19] suggested a further generalization of this idea, applying the Yule-Nielsen correction to Heuberger's cellular model. He further suggested performing a characterization of this model based on conventionally angled screens, another for dot-on-dot screening, and accounting for the inevitable misregistration by using a convex sum of the reflectance spectra predicted by each model. Balasubramanian presented empirical evidence that, while the use of this linear blending provided a significant increase in accuracy for the non-cellular approach, the cellular approach, even with modest sub-partitioning, produced not only superior accuracy, but results that were relatively invariant with respect to the parameter of the convex combination; refer to his Figure 8 on page 164. Pjanic and Hersch recently used this model to predict specular reflectance on printed metallic substrate. [20]

While promising, that insertion of a minimal level of partitioning performs as well as or better than blending reflectance for two overlap modes suggests that at least some of the utility and appeal of the cellular approach lies in its ability to compensate for (or adapt to) departures of the actual dot overlap behavior from that assumed by one of the classic overlap models of Demichel, dot-on-dot, or dot-off-dot.

Hersch and Crètè [21] found evidence that, when using Eq (1), colorant amounts appear to depend on the amount of previously-printed colorants. In other words, if magenta is deposited after cyan, the same requested amount of magenta colorant may result in a declining amount of magenta colorant as the amount of cyan colorant increases. They proposed a solution for this; their revised model fits within the framework of Neugebauer-type models as described above with some inter-channel pre-processing is performed on the input colorant amounts. Again, it is possible that a component of this improvement may be attributable to less-than-perfect modeling of dot overlap behavior.

Probability interpretation of scalars

The scalars act as weights for the primaries, so it is intuitive to think of them as the fraction of area occupied by each primary. Extending this intuitive picture further, Hersch articulated a probability interpretation of the scalars. [5] While Hersch considered only the case introduced by Demichel (the subject of Hersch's article), applicable to certain types of screening (when the presence or absence of a colorant is independent of the presence or absence of any of the other colorants, as in conventionally angled screening),

Motivation for this study

The main restrictions on the function ϕ is that it be defined on the unit interval, and that it possess an inverse to permit computa-

tion of the result. Thus, varied families of Neugebauer-like models may be easily generated. On the other hand, the restrictions on the scalars seem more intimidating; not only must they be non-negative, but they must also sum to unity. Naturally, they must reflect the actual overlap behavior with reasonable accuracy. Rather than a single function, there are typically eight (for three-ink printing) or 16 (for four-ink printing). Under the current paradigm, the number of equations grows exponentially with the number of colorants. For inkjet printers having six, seven, or ten inks, the complexity increases dramatically, complicating coding and testing and leaving many more opportunities for error.

When modeling dot-on-dot printing, it is unclear how to account for departure from perfect concentric dots caused by inevitable misregistration. Even if one derived a set of equations (one for each primary) for the scalars as functions of the colorant amounts, one equation for each primary would need to be coded and tested. To offer another example, one may employ dot-on-dot screening for cyan, magenta, and yellow, and, with an eye towards maximizing gamut, use the dot-off-dot arrangement for the black relative to the others. How should the scalars be calculated under these circumstances? Finally, in six- and seven-ink printing, common screen angles may be shared by pairs of inks. Again, it is unclear under the current state of the art how to proceed, and would require extensive coding and testing if and when a solution became available.

Clearly, it would be advantageous, should dot overlap behavior change, or should one wish to simulate a different type of dot overlap behavior, to have the following options:

- Specify a single, relatively simple, equation, easy to code and test;
- Have a rich parametric family in which a few parameters could be changed or tuned;
- Build a complete overlap model for many inks easily by combining/leveraging very simple models.

It will be shown that the new approach for computing the scalars, or area fractions of each primary, has many advantages over the the existing method. These include:

Simpler coding. The calculation of scalars is factored into two independent tasks, simplifying coding.

Greater flexibility. The model for colorant overlap is provided in a single function, easily changed, permitting run-time selection of this component of the model.

Easier testing. The separation of the calculation into two independent tasks also simplifies testing of the code for new overlap models.

Ability to handle complex overlap models. Models for complex colorant overlap behavior may often be simple to write in terms of simpler models. An example of this will be provided later.

Many families of existing models. In the next paper in this series, it will be shown how to leverage models from statistical literature for not just for traditional colorant overlap situations, but for many others, as well.

Tuning opportunities. Rich families of functions with one or two parameters may be employed, enabling an opportunity for optimizing accuracy through judicious parameter selection.

Some definitions

Some specialized notation and nomenclature will be used in this paper.

Primaries as power set of colorants

The distinction must be made between colorants (inks) and primaries. The symbol i will be used for the number of inks or colorants, and p for the number of primaries. Mathematically, the primaries are the *power set* [23] of the colorants, with $p = 2^i$.

If a set of three inks ($i = 3$) are represented with the single letters "c," "m," and "y," respectively, there will be $p = 8$ primaries: the plain paper, denoted in set notation as {} (none of the inks), the individual inks {c}, {m}, and {y}, the two-colorant overprints {c, m}, {c, y}, and {c, m}, and the three-colorant overprint {c, m, y}.

Context will generally permit omission of the set-denoting braces and commas when naming the primaries. For example, the cyan and magenta overprint primary, {c, m}, may be abbreviated as "cm" with the understanding that this denotes the primary itself, rather than the product of the two colorant amounts. The plain paper primary, {}, may be written as "w" (for "white"). Thus, the scalar ("area fraction") for the cyan and magenta overprint will be written as a_{cm}, and reference will be made to the "cm" primary.

Cardinality and parity of a primary

The *cardinality* of a primary A, denoted $\#A$, is the number of colorants that make up that primary. For example, the primaries c, m, and y each have cardinality 1; cm, cy, and my all have cardinality 2, and so on. The cardinality of the plain paper is 0 ($\#w = 0$).

A primary will be said to have *even parity* if its cardinality is even, and similarly *odd parity* if its cardinality is odd. For example, the primaries cm, cy, and my all have even parity, while the single inks and the three-colorant overprint all have odd parity. Note that the unprinted substrate has zero colorants, hence it has even parity.

Scalar for final primary

The scalar for the primary composed of all colorants, which shall be referred to as the *final primary*, characterizes the entire overlap behavior, insofar as calculation of the scalars is concerned. In the next two sections, it will be shown that all scalars may be algorithmically computed in terms of this function. Because of this key role, the notation $F(c, m, \ldots)$ for this function will be introduced.

Under the probability interpretation of the scalars, the final scalar is a *joint probability*, and the function $F(c, m, \ldots)$ is a *(cumulative) joint probability distribution function*. More specifically, because all colorant amounts are restricted to the unit interval $[0, 1]$, and the probability that a light ray is incident on a region covered by a colorant is equal to the amount of that colorant (i.e., $P(c) = c$, for $c \in [0, 1]$), F may be further characterized as a joint distribution function with *unit uniform marginal distributions*.

Scalar Computation for Two Colorants

In this section, the colorants (and colorant amounts) will be referred to using the symbols c and m.

Recall that the function that computes the area fraction of the i-color overprint from the colorant amounts is denoted F.

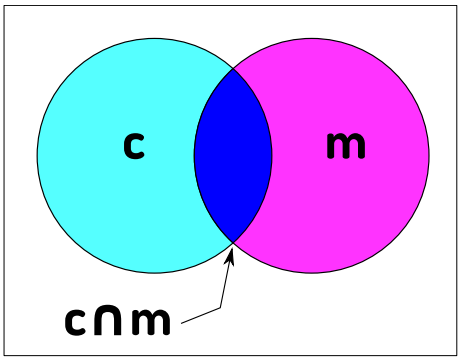

Figure 1. *Venn diagram for two colorants*

The event "c" corresponds to the entire circle on the left; the event "m" is indicated by the entire circle on the right; their intersection, "c ∩ m," is the lens-shaped region they share; their union, $c \cup m$, is the region occupied by one, the other, or both. The regions are not to scale; in particular, the intersection could have zero area, or consume all of both c and m, or be anywhere between these extremes, depending on the relative colorant amounts and the dot overlap modality.

For 2-ink conventionally-angled screens, $F(c,m) = c \cdot m$, for 2-colorant dot-on-dot, $F(c,m) = \min(c,m)$, and for 2-colorant dot-off-dot, $F(c,m) = \max(0, c+m-1)$. (While many other overlap functions exist, these three are the most familiar. The method described here applies for any overprint for binary printing processes.)

Make note of a few simple identities that apply to all three cases:

1. $F(1,1) = 1$
2. $F(c,1) = c$
3. $F(1,m) = m$

Probability model for scalars, two colorant case

The symbol "∩" denotes the logical intersection, or "and;" the symbol "∪" is used to denote the logical union, or "or." These events are illustrated for two colorants in Figure 1. In addition, the overbar, as in $\overline{e}$, denotes the logical negation of the event e ("not"). $P(e)$ is the probability of the event e. The scalars are then:

$$\begin{aligned} a_{cm} &= P(c \cap m) = F(c,m) \\ a_c &= P(c) - P(c \cap m) = c - F(c,m) \\ a_m &= P(m) - P(c \cap m) = m - F(c,m) \\ a_w &= P(\overline{c \cup m}) = 1 - P(c) - [P(m) - P(c \cap m)] \\ &= 1 - c - m + F(c,m) \end{aligned} \tag{3}$$

Building the coefficient matrix

While this case is rather simple, a matrix solution will nevertheless be provided to help elucidate the solutions for higher-order cases. This matrix method may be applied to an arbitrarily large colorant set. The matrix may be constructed by first writing two header rows and columns. The first header row and column contain the primaries. While any order will do, Yates's *standard order* [24] will be used in this investigation (it has several attractive properties, including an algorithmic definition and a fast test for inclusion). It is recommended to use the same order for the row and column header. The second header row and column are the parities of each primary. Using "e" for even parity and "o" for odd parity, and keeping in mind that the primary w has zero colorants:

		w	c	m	cm
		e	o	o	e
w	e				
c	o				
m	o				
cm	e				

Build the body of the table row by row, using the following rules:

1. If the row primary not a subset of the column primary, enter a zero.

   Otherwise,

2. If both primaries have the same parity, enter a one at that position.

3. If the primaries have different parities, enter the value "-1" at that position.[2]

For the first row, only the second and third rules apply, because the plain paper is a subset of all primaries. The single-ink primaries c and m have opposite parity, while w and cm have the same parity as w, so the matrix, after filling in the first row is:

		w	c	m	cm
		e	o	o	e
w	e	1	-1	-1	1
c	o				
m	o				
cm	e				

Applying the rules to the remaining rows yields:

		w	c	m	cm
		e	o	o	e
w	e	1	-1	-1	1
c	o	0	1	0	-1
m	o	0	0	1	-1
cm	e	0	0	0	1

Checking the matrix

One may perform some checks on the entries; the number on non-zero entries in each row will be p for the primary w, $p \div 2$ for the single-colorant primaries, and $p \div 4$ for the two-colorant primary. The coefficient sum for each row, except for the row for the primary with all colorants, should be zero.

More generally, the number of non-zero elements in the row corresponding to an arbitrary primary A will be $p \div 2^{\#A}$.

[2] This is recommended for hand computation. For machine computation, the second "header" row and column likely to be in a single sequence such as a list, array, or similar container, and are populated with the cardinality of each primary. Rules 2 and 3 may be revised as follows: 2a. If the sum of the row and column cardinalities is even, a one is entered at that position; 3a. If the sum of the cardinalities of the row and column is odd, assign a "-1" at that position. Testing for inclusion of row primary in the column primary is still performed. This is the procedure used by the demonstration software.

Left- and right side vectors

Form the vector for the left side of the matrix equation from the first header column, attaching each primary to a scalar:

$$\begin{bmatrix} a_w \\ a_c \\ a_m \\ a_{cm} \end{bmatrix}$$

Form the right-hand vector from the first header column, using each as arguments to the 2-colorant overlap function, F, replacing empty positions with 1s in the argument list, and substituting the identities $F(1,1) = 1$, $F(c,1) = c$, and $F(1,m) = m$:

$$\begin{bmatrix} F(1,1) \\ F(c,1) \\ F(1,m) \\ F(c,m) \end{bmatrix}, \text{ equivalent to } \begin{bmatrix} 1 \\ c \\ m \\ F(c,m) \end{bmatrix}$$

The matrix equation is then written as:

$$\begin{bmatrix} a_w \\ a_c \\ a_m \\ a_{cm} \end{bmatrix} = \begin{bmatrix} 1 & -1 & -1 & 1 \\ 0 & 1 & 0 & -1 \\ 0 & 0 & 1 & -1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ c \\ m \\ F(c,m) \end{bmatrix} \tag{4}$$

or, in non-matrix notation:

$$\begin{aligned} a_w &= 1 - c - m + F(c,m) \\ a_c &= c - F(c,m) \\ a_m &= m - F(c,m) \\ a_{cm} &= F(c,m) \end{aligned} \tag{4a}$$

Verification

The inverse of the coefficient matrix in Eq (4) is nearly identical; simply replace all -1s with +1s. The linear system written in the other direction (values of F as functions of the scalars) is:

$$\begin{aligned} 1 &= a_w + a_c + a_m + a_{cm} \\ c &= a_c + a_{cm} \\ m &= a_m + a_{cm} \\ F(c,m) &= a_{cm} \end{aligned} \tag{5}$$

The first equation in (5) is consistent with Eq (1) being a convex sum (coefficients sum to unity), while the last is true by definition of F. The remaining two are easily verified for all three overlap modes presented earlier (conventionally angled screen using Demichel, dot-on-dot, and dot-off-dot).

Scalar Computation for Three Colorants

If the probability argument from the two-colorant case is extended to three colorants, one is led to the same solution presented below. In the interest of practicality and brevity, only the algorithmic solution is presented here.

A third colorant (here, "y") is appended to the colorant list. The list of primaries in Yates's standard order is obtained by repeating the list of primaries from the two-colorant case and including this third colorant in all the repeated members. The list of primaries is again used as a header row and column to build the coefficient matrix, as is the parity of each primary. The same three rules are applied to complete the body of the matrix. The coefficient matrix with the header rows and columns appears below:

		w	c	m	cm	y	cy	my	cmy
		e	o	o	e	o	e	e	o
w	e	1	-1	-1	1	-1	1	1	-1
c	o	0	1	0	-1	0	-1	0	1
m	o	0	0	1	-1	0	0	-1	1
cm	e	0	0	0	1	0	0	0	-1
y	o	0	0	0	0	1	-1	-1	1
cy	e	0	0	0	0	0	1	0	-1
my	e	0	0	0	0	0	0	1	-1
cmy	o	0	0	0	0	0	0	0	1

Left- and right-side vectors

As in the two-colorant case, the argument lists for the overlap function F are built from the primaries by inserting a "1" in any positions where a colorant is missing, and making use of the identities $F(c,1,1) = c$, $F(1,m,1) = m$, $F(1,1,y) = y$, and $F(1,1,1) = 1$. The vectors on the left and right sides of the matrix equation are, respectively:

$$\begin{bmatrix} a_w \\ a_c \\ a_m \\ a_{cm} \\ a_y \\ a_{cy} \\ a_{my} \\ a_{cmy} \end{bmatrix} \text{ and } \begin{bmatrix} 1 \\ c \\ m \\ F(c,m,1) \\ y \\ F(c,1,y) \\ F(1,m,y) \\ F(c,m,y) \end{bmatrix}$$

The complete set of linear equations, in matrix form, is:

$$\begin{bmatrix} a_w \\ a_c \\ a_m \\ a_{cm} \\ a_y \\ a_{cy} \\ a_{my} \\ a_{cmy} \end{bmatrix} = \begin{bmatrix} 1 & -1 & -1 & 1 & -1 & 1 & 1 & -1 \\ 0 & 1 & 0 & -1 & 0 & -1 & 0 & 1 \\ 0 & 0 & 1 & -1 & 0 & 0 & -1 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & -1 \\ 0 & 0 & 0 & 0 & 1 & -1 & -1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & -1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ c \\ m \\ F(c,m,1) \\ y \\ F(c,1,y) \\ F(1,m,y) \\ F(c,m,y) \end{bmatrix} \tag{6}$$

Arbitrarily-sized Colorant Sets

This procedure is easily extended to colorant sets of arbitrary size. Software written in the Python computer programming language is available online[3] under the GNU General Public License, Version 3.

Utilities helpful in implementing the new method are supplied in a Python module. The functions provided can start with a single string containing one letter for each colorant and generate the set of primaries in Yates's standard order. An example session follows; the ">>>" is the prompt from the Python interactive shell:

```
>>> from neugebauer_scalar_utils import *
>>> inks = 'cmy'
```

[3] https://sourceforge.net/projects/neugebauerscalar/

```
>>> primaries = yates (inks)
>>> print primaries
['', 'c', 'm', 'cm', 'y', 'cy', 'my', 'cmy']
>>> matrix = build_coef_matrix (primaries)
>>> for row in matrix:
print row

[1, -1, -1, 1, -1, 1, 1, -1]
[0, 1, 0, -1, 0, -1, 0, 1]
[0, 0, 1, -1, 0, 0, -1, 1]
[0, 0, 0, 1, 0, 0, 0, -1]
[0, 0, 0, 0, 1, -1, -1, 1]
[0, 0, 0, 0, 0, 1, 0, -1]
[0, 0, 0, 0, 0, 0, 1, -1]
[0, 0, 0, 0, 0, 0, 0, 1]
>>> var, stmts = cache_function_calls (primaries)
>>> code = build_code (var, stmts, matrix)
>>> for statement in code:
if len (statement):
print statement

def scalars (F, c, m, y):
    f_cm = F (c,m,1)
    f_cy = F (c,1,y)
    f_my = F (1,m,y)
    f_cmy = F (c,m,y)
    return (
        1 - c - m + f_cm - y + f_cy + f_my - f_cmy,
        c - f_cm - f_cy + f_cmy,
        m - f_cm - f_my + f_cmy,
        f_cm - f_cmy,
        y - f_cy - f_my + f_cmy,
        f_cy - f_cmy,
        f_my - f_cmy,
        f_cmy,
    )
>>>
```

In the first line, the contents of the library (called a module in Python) are imported into the current namespace. Next, the colorant set is specified; one character is used for each colorant. The module-supplied functions `yates`, `build_coef_matrix`, `cache_function_calls`, and `build_code` are called. The return value from this last function contains a list of Python statements; they constitute the Python code to compute the scalars independent of the specific overlap model being used.

If the code generated in the example above were put in a Python source file, and the following two functions also defined in that file, the first for the Demichel equations, the second for dot-on-dot:

```
def independence (c, m, y):
    """Used for angled screens and independent
       random-dot patterns;
       implements the Demichel equations.
    """
    return c * m * y

def dot_on_dot (c, m, y):
    """Used for dot-on-dot screening patterns.
    """
    return min (c, m, y)
```

and the source file imported into an interactive session, one could do the following:

```
>>> scalars (independence, 0.2, 0.4, 0.6)
(0.19200000000000003, 0.04800000000000001,
0.12800000000000003, 0.03200000000000001,
0.288, 0.07199999999999998,
0.19199999999999998, 0.04800000000000001)

>>>scalars (dot_on_dot, 0.2, 0.4, 0.6)
(0.4000000000000001, 0.0, 0.0, 0.0,
0.19999999999999996, 0.0, 0.2, 0.2)
```

The reader's attention is drawn to three points: First, the function bodies (the part coming after the function declaration and the triple-quoted documentation string) specific to the overlap models are a single, simple line of code. Secondly, exercising either model involved a simple call to the function provided by the demonstration software. Thirdly, switching from one overlap model to the other required only changing a single argument in this function call.

More specific examples for using this code are included in the demonstration suite.

Testing

Several tests were performed on the software suite. Results for two, three, and four colorants were checked and agreed with hand calculations.

In order to provide some validation of both the software and the underlying theory, numerical tests were performed. Code was written in Python to compute scalars using both the old and new methods. (The test programs are also available for download from the project site.) One million random colorant amount vectors were generated, and the scalars for each method were compared.

The results of this testing are summarized in Table 1. For two colorants, the old and new methods were compared for dot-off-dot, dot-on-dot, and conventionally-angled screens governed by the Demichel equations. For three colorants, the comparisons were performed for dot-on-dot and angled screens; dot-off-dot is limited to two colorants. Because the computer code for the traditional computation method became difficult to write for dot-on-dot, it was not compared for more than three colorants. (It could easily have been exercised for six colorants, for example; doing so requires only substituting the one-line function used for six-colorant angled screens for the one-line function that governs six-colorant dot-on-dot.) Comparisons for four, five, six, and eight colorants were performed for angled screens only.

The magnitude of the scalars are comparable to one, so it is valid to compare them directly to the computing environment's floating point epsilon, here, approximately 2.210^{-16}. In all cases, the absolute difference between the scalars computed using the old and new methods was no more than two orders of magnitude greater than the floating point epsilon, indicating primarily rounding error. In any case, the differences have no practical impact.

Conclusions and Future Work

It has been shown that the computation of scalars in Neugebauer-type models may be factored into two independent components. One is indifferent to overlap modality and contains an expression for each scalar to be computed; the other encapsulates the specific overlap behavior and may be very compact, consisting, perhaps, of a single expression. An algorithm for generating a matrix for the first component has been described, and demonstration software to

Table 1. Results of testing.

Scalars computed using traditional and new methods agree to within 10^{-14} *or better for one million runs for 2, 3, 4, 5, 6, and 8 colorants.*

overlap mode	# of inks	# of primaries	# of comparisons	Largest Error
Dot off Dot	2	4	4 000 000	$< 10^{-16}$
Dot on Dot	2	4	4 000 000	$< 10^{-16}$
Angled	2	4	4 000 000	$< 10^{-15}$
Dot on Dot	3	8	8 000 000	$< 10^{-16}$
Angled	3	8	8 000 000	$< 10^{-15}$
Angled	4	16	16 000 000	$< 10^{-15}$
Angled	5	32	32 000 000	$< 10^{-15}$
Angled	6	64	64 000 000	$< 10^{-15}$
Angled	8	256	256 000 000	$< 10^{-14}$

perform this has been made available under an open-source license.

The refactoring process has been verified for the three main overlap models using two colorants, two of them (dot-on-dot and angled screening) for three colorants, and for one of them (angled screens) additionally for four, five, six, and eight colorants. In nearly 400 million scalar comparisons, no difference larger than 10^{-14} was encountered.

The next paper in this series will enumerate the characteristics of the function that characterizes a dot overlap modality, identify the class of function, and provide examples. In the third paper, strategies for fitting these functions to data will be discussed.

Other future work may include revision of the software library so it generates code in other languages, such as C, C++, or Java.

References

[1] Stefan Gustavson and Björn Kruse. "Evaluation of a light diffusion model for dot gain." In *TAGA Proceedings* (Technical Association of the Graphic Arts, 1996), p. 58–68.

[2] Stefan Gustavson. *Dot Gain in Colour Halftones*. Ph.D. thesis (Linköping, SE: Linköping Institute of Technology, 1997).

[3] *Procèdè*, **26** (3):17–21 (1924).

[4] *Procèdè*, **26** (4):26–27 (1924).

[5] Roger D Hersch. "Demichel equations." In *Encyclopedia of Color Science and Technology*, p 572–575 (New York: Springer, 2106).

[6] Martin Fowler. *Refactoring: Improving the design of existing code*. Reading, MA: Addison-Wesley (1999), p xvi.

[7] Fowler, p 7.

[8] Hans E J Neugebauer. "Die theoretischen Grundlagen des Mehrfarbenbuchdrucks." *Zeitschrift für wissenschaftliche Photographie Photophysik und Photochemie*, **36** (4):73–89 (1937).

[9] J A C Yule and W J Neilsen [sic]. "The penetration of light into paper and its effect on halftone reproduction." In *TAGA Proceedings* (Technical Association of the Graphic Arts, 1951), p. 65–76.

[10] J A Stephen Viggiano. "The color of halftone tints." In *TAGA Proceedings* (Technical Association of the Graphic Arts, 1985), p. 647–661.

[11] J A Stephen Viggiano. "Modeling the color of multi-color halftones." In *TAGA Proceedings* (Technical Association of the Graphic Arts, 1990), p. 44–62.

[12] J A Stephen Viggiano. *Models for the Prediction of Color in Graphic Reproduction Technology*. MSc thesis (Rochester, NY: Rochester Institute of Technology, 1987).

[13] Raja Balasubramanian. "Printer model for dot-on-dot halftone screens." In *Proceedings of SPIE Volume 2413 Color Hard Copy and Graphic Arts IV* (International Society for Optical Engineering, 1995), p. 356–364.

[14] Achim Lewandowski, Marcus Ludl, Gerald Byrne, and Georg Dorffner. "Applying the Yule-Nielsen equation with negative *n*." *Journal of the Optical Society of America* A, **23** (8):1827–1834 (2006).

[15] Mathieu Hébert and Roger David Hersch. "Reflectance and transmittance model for recto-verso halftone prints: spectral predictions with multi-ink halftones." *Journal of the Optical Society of America* A, **26** (2):356–364 (2009).

[16] Mathieu Hébert and Roger D. Hersch. "Yule–Nielsen based recto–verso color halftone transmittance prediction model." *Applied Optics*, **50** (4):519–525 (2011).

[17] Vahid Babaei and Roger D Hersch. "Color reproduction of metallic-ink images." *Journal of Imaging Science and Technology*, **60** (3):30503-1 – 10 (2016).

[18] Karl J. Heuberger, Zhou Mo Jing, and Serdar Persiev. "Color transformations and lookup tables." In *TAGA/ISCC Proceedings*. (Technical Association of the Graphic Arts, 1992), p. 863–881.

[19] Raja Balasubramanian. "Optimization of the spectral Neugebauer model for printer characterization." *Journal of Electronic Imaging*, 8 (2):156–166 (1999).

[20] Petar Pjanic and Roger D Hersch. "Specular color imaging on a metallic substrate." In *21st Color and Imaging Conference Final Program and Proceedings*. (IS&T, 2013), p. 61 – 68.

[21] Roger David Hersch and Frèdèrique Crètè. "Improving the Yule-Nielsen modified spectral Neugebauer model by dot surface coverages depending on the ink superposition conditions." In *SPIE vol. 5667: Color Imaging* X (International Society for Optical Engineering, 2005), p. 434–445.

[22] F R Clapper and J A C Yule. "The effect of multiple internal reflections on the densities of half-tone prints on paper." *Journal of the Optical Society of America*, **43** (7):600–603 (1953).

[23] Kam-tim Leung and Doris Lai-chue Chen. *Elementary Set Theory, Part 1* (Hong Kong: Hong Kong University Press, 1979), p 36–37.

[24] Samuel Kotz and Normal L Johnson, editors. "Yates' method," in *Encyclopedia of Statistical Sciences*, volume 9 (New York: John Wiley and Sons, 1982), p 659–662.