Exact Computation of Scalar, 2D Aerial Imagery

R.L. Gordon
IBM Corp., 1580 Rt. 52, Hopewell Jct., NY 12533

ABSTRACT

An exact formulation of the problem of imaging a 2D object through a Köhler illumination system is presented; the accurate simulation of a real layout is then not time-limited, but memory-limited. The main idea behind the algorithm is that the boundary of the region that comprise a typical TCC is made up of circular arcs, and therefore the area – which determines the value of the TCC – should be exactly computable in terms of elementary analytical functions. A change to integration around the boundary leads to an expression with minimal dependence on expensive functions such as arctangents and square roots. Numerical comparisons are made for a simple 2D structure.

Keywords: lithography, aerial image simulation, Hopkins transmission cross-coefficients, PROLITH, SPLAT

1. INTRODUCTION

The computation of the image intensity in air of a transilluminated object in a Köhler illumination system is, in principle, a relatively straightforward problem that was effectively solved by the inimitable H.H. Hopkins about 50 years ago[1]. The setup of the Köhler illumination system itself facilitates mathematical models of the image intensity: each point in the source is a plane wave incident on the object, and the telecentricity of the system implies that the subsequent diffraction of the light by the objective is governed by the simple Debye integral for the electric field in the image plane[2]. The challenging part of the computation in most cases is in the integration of the intensities resulting from the plane waves incident on the mask from various angles, especially if the source – each point of which produces one of those plane waves - is an irregular shape. Nevertheless, no new, groundbreaking physics surfaces in attacking this problem. This then begs the question of why so much interest and resources are being dedicated to precisely this seemingly simple problem by lithographers today.

An answer to this question lies in the sheer demands now being placed on lithography simulation, where aerial image simulation is the most complex and resource-intensive piece in many cases. Not only is aerial image simulation used to explore how the image and corresponding figures of merit behave as a function of stepper and/or mask parameters, but it has also become an integral piece of the design-for-manufacture infrastructure that has been built around the concept of resolution enhancement technologies such as optical proximity correction and phase-shift masks. The consequence of such a utility change is that the per-session computational need has moved from, say, dozens to low-hundreds aerial image simulations to the order of thousands to tens of thousands. It is therefore essential that the computation of the aerial image become as fast as possible, sacrificing little accuracy along the way.

Many aerial image simulators employ what has become known as the "Hopkins Model"[3]. Consider a typical microlithographic imaging system, comprised of a (finite) source, condenser optics, a semitransparent mask to be imaged, objective optics, and a silicon wafer in the image plane. In an ideal system, the source is imaged onto the entrance pupil of the objective lens by the condenser, and the mask is in the Fourier transform (focal) plane of the condenser. The idea here is that each point of the source is then mapped onto a unique plane wave, incident at some angle upon the mask. Further, a uniform source is mapped onto a uniform distribution of plane wave amplitudes upon the mask. The resulting expression for the intensity I at wafer position $\mathbf{x}$ is

$$I(\mathbf{x}) = \int_{\Re^2} d^2\mathbf{u}' \int_{\Re^2} d^2\mathbf{u}'' \; M(\mathbf{u}')T(\mathbf{u}',\mathbf{u}'')M^*(\mathbf{u}'')\exp\left[-i2\pi\frac{NA}{\lambda}(\mathbf{u}'-\mathbf{u}'')\cdot\mathbf{x}\right], \tag{1}$$

where the transmission cross-coefficient T is given by

$$T(\mathbf{u}',\mathbf{u}'') = \int_{\Re^2} d^2\boldsymbol{\sigma}\, S(\boldsymbol{\sigma})P(\mathbf{u}'+\boldsymbol{\sigma})P^*(\mathbf{u}''+\boldsymbol{\sigma}), \tag{2}$$

and where M represents the Fourier transform of the mask transmittance function, or frequency response of the mask, P represents the pupil function, or frequency response of the optical system, S represents the intensity distribution on the source, NA is the numerical aperture of the objective lens, and λ is the illumination wavelength. The vector quantity $\mathbf{u}$ represents a scaled frequency coordinate in that the pupil function is zero when $|\mathbf{u}| > 1$. The vector quantity $NA\cdot\boldsymbol{\sigma}$ represents

The author may be reached via email (rlgordon@us.ibm.com) or phone (845) 892 2432.

the projection of a point on the unit sphere onto the mask plane; the notation is intentionally suggestive for a Köhler illumination system. That is, as *NA* denotes the numerical aperture, $\boldsymbol{\sigma}$ represents a point in the illuminator.

Whether considering relatively small areas in a repeating pattern, or larger areas in a nonrepeating one, the specter of computing thousands of unwieldy transmission cross-coefficient (TCC) double integrals is enough to hinder the development of an accurate set of chip design corrections. Clearly, if one is interested in reducing the computational burden from the aerial image simulation alone without affecting the accuracy, then one of two routes must be chosen: decrease the number of TCC's needed to describe the system through symmetry, or find a faster way to compute the TCC's. This work will address the latter issue, although the former method is by all means desirable. Section 2 will examine the Hopkins' model for periodic objects in more detail. Clearly, such a scheme is also the basis of the "fast" schemes currently in use in OPC design tools, and the interested reader is referred to the several excellent publications on the subject[4,5].

Section 3 contains the thrust of the new results presented in this work. Ideally, the name of the game is to avoid the double integration in Eq. (2) with as little approximation as possible. In 1977, Kintner outlined in a groundbreaking paper a methodology for computing the in-focus aerial image of a 1D object using exact analytical expressions for the TCC's. While there is still much to be desired for defocused images, this work will concentrate on the in-focus case as Kintner did[*]. The difference here is, of course, that attention is now turned to the general 2D images of interest to modern simulators; Kintner's result is, in fact, only of practical interest in benchmarking 1D simulators. Finally, in Section 4, computational results using these new precise descriptions of the TCC's are shown and are compared with standard results that simply grid frequency space in their TCC evaluations.

2. PERIODIC OBJECTS: HOPKINS' INTEGRATION

We now turn our attention to Eqs. (1) and (2). Consider a semitransparent, periodic object with x- and y-periods p_x and p_y, respectively. Such an object is represented mathematically as

$$M(\mathbf{u}) = \sum_{m=-\infty}^{\infty}\sum_{n=-\infty}^{\infty} c_{mn}\,\delta\!\left(u_x - \frac{m}{p_x}\frac{\lambda}{NA}\right)\delta\!\left(u_y - \frac{n}{p_y}\frac{\lambda}{NA}\right), \tag{3}$$

where the c_{mn} are the mask Fourier coefficients. (Of course, the entire spectrum is not used in determining the image, as the objective lens filters out the higher-order coefficients.) When Eq. (3) is substituted into Eq. (2), a series for the intensity results:

$$I(\mathbf{x}) = \sum_{m'=-\infty}^{\infty}\sum_{m''=-\infty}^{\infty}\sum_{n'=-\infty}^{\infty}\sum_{n''=-\infty}^{\infty} c_{m'n'}c^{*}_{m''n''}T_{m'n'm''n''}\exp\left\{-i2\pi\left[(m'-m'')\frac{x}{p_x}+(n'-n'')\frac{y}{p_y}\right]\right\}, \tag{4}$$

where

$$T_{m'n'm''n''} = T\!\left(\frac{m'}{q_x},\frac{n'}{q_y},\frac{m''}{q_x},\frac{n''}{q_y}\right), \tag{5}$$

and the quantity $q = pNA/\lambda$. The form for the intensity in Eq. (5) does not seem to offer any clear advantage over the ray-based method. In principle, one could sum Eq. (4) directly, as the quantity $T_{m'n'm''n''}$ is bound to vanish when the indices are large enough, making the sum finite. To see how this works, we take a closer look at the definition of this coefficient in Eq. (4). The pupil function and source distribution function has finite support; that is, $|\mathbf{u}|>1 \Rightarrow P(\mathbf{u})=0$ and $|\mathbf{u}|>\sigma_{\max} \Rightarrow S(\mathbf{u})=0$. This then reduces the infinite integration interval specified in Eq. (3) into a finite interval, and this is illustrated in Figure 1 below.

[*] The paraxial, defocused case was treated by S. Subramanian, "Rapid Calculation of Defocused Partially Coherent Images", *Appl. Opt.* **20**, 1854-1857 (1981). There, he was able to reduce the TCCs to one-dimensional integrals over the regions defined by Kintner.

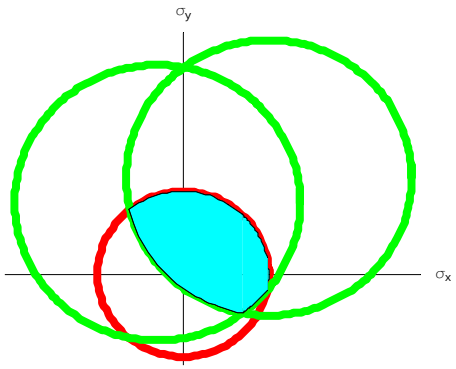


Figure 1 Geometry for computing the transmission cross-coefficient. The region of nonzero pupil values is represented by either of the larger circles, while that of nonzero source distribution is the smaller. The presence of two larger circles offset from both the origin and each other is a consequence of the convolution form of the integral in Eq. (2); the offsets are equal to the spatial frequencies. The transmission cross-coefficient is then equal to the integral over the intersection of these 3 regions and is represented by filled region.

The summation limits in Eq. (4) are now deduced from the bounds of the transmission cross-coefficients. Two geometrical conditions must be met for nonzero values: first, the separation between the two offset pupils must be less than twice their radius, or 2 in this coordinate scale, and second, both pupils must touch the source. Letting $k = m' - m''$ and $l = n' - n''$, these conditions translate into the following:

$$\left|\mathbf{u}' - \mathbf{u}''\right| \le 2 \Rightarrow \frac{k^2}{q_x^2} + \frac{l^2}{q_y^2} \le 4, \tag{6}$$

$$\max\left\{\left|\mathbf{u}'\right|, \left|\mathbf{u}''\right|\right\} \le 1 + \sigma \Rightarrow \max\left\{\frac{m'^2}{q_x^2} + \frac{n'^2}{q_y^2}, \frac{m''^2}{q_x^2} + \frac{n''^2}{q_y^2}\right\} \le (1+\sigma)^2. \tag{7}$$

The conditions in Eqs. (6) and (7), and the change of variable suggested above, lead directly to the following exact finite sum for the intensity. Denoting $\lfloor X \rfloor$ as the greatest integer less than X:

$$I(x,y) = \sum_{k=-M_x}^{M_x} \sum_{l=-M_y(k)}^{M_y(k)} C_{kl} \exp\left[-i2\pi\left(k\frac{x}{p_x} + l\frac{y}{p_y}\right)\right], \tag{8}$$

where

$$M_x = \lfloor 2q_x \rfloor, \tag{9}$$

$$M_y(k) = \left\lfloor q_y \sqrt{4 - \frac{k^2}{q_x^2}} \right\rfloor, \tag{10}$$

and

$$C_{kl} = \sum_{m=-k\theta(-k)-N_x}^{N_x - k\theta(k)} \sum_{n=-l\theta(-l)-N_y(m,k)}^{N_y(m,k)-l\theta(l)} c_{m+k,n+l} c_{mn}^* T_{m+k,n+l,m,n}, \tag{11}$$

$$N_x = \lfloor (1+\sigma) q_x \rfloor, \tag{12}$$

$$N_y(m,k)=\left\lfloor q_y\sqrt{(1+\sigma)^2-\frac{(m+k)^2}{q_x^2}}\right\rfloor, \tag{13}$$

and θ is the Heaviside step function:

$$\theta(x)=\begin{cases}1, & x>0\\ 0, & x<0\end{cases}. \tag{14}$$

Note that all of the information pertaining to the optical system is contained in the TCC's. Because the TCC's are computed via the integrations over the source, they can be stored in an array for a very efficient mask perturbation scheme. That is, as long as one only varies the mask features (without changing the pitch), then the TCC's need only be computed once; once they are known, then the rest of the computation is done in a small fraction of the time it took to compute the TCC's. Therefore, this frequency-space approach is ideal for optical proximity correction, in which layouts are optimized for optical diffraction and lithography process conditions.

From this explicit expression for the intensity, the size of the computational task is easily estimated due to the fact that the computation time is dominated by the calculation of the TCC's. The number of TCC's needed to compute the intensity is then the quantity we seek here. (The time to compute a single TCC is also important, but this is dependent on the integration scheme employed.) Eqs. (6) and (7) describe integer lattices bounded by ellipses, and therefore the number of points bounded by the ellipses is roughly equal to the product of the ellipses areas. An approximation to the number of TCC's $\Im$, in the limit of a large unit cell area, is

$$\Im\approx\pi^2\left\lfloor\left(\frac{2\,NA}{\lambda}\right)^2 p_x\,p_y\right\rfloor\left\lfloor\left[\frac{(1+\sigma)NA}{\lambda}\right]^2 p_x p_y\right\rfloor. \tag{15}$$

Note that this number is roughly proportional to the square of the period cell area, making this method unwieldy for large areas. Clearly, the reality of the intensity can reduce this number a factor of 2, and mask symmetries can be harnessed for further reductions. One can imagine a patching scheme, whereby a larger area is split into smaller areas in order to reduce the computational burden; such a scheme, however, is beyond the scope of this paper.

3. ANALYTICAL EVALUATION OF THE TCC'S IN FOCUS

3.1. Generalization of Kintner's Algorithm

The computation of the in-focus TCC is reduced to simply computing the common area of 3 circles, 2 of which are offset from the origin and have unit radius (the pupils), while the other is centered at the origin and has radius of σ (the source), as pictured in Figure 1. Kintner showed that, for the 1D case where the centers of the pupils are constrained to lie on the horizontal (or vertical) axis, and where σ<1, the TCC geometry takes the form of one of 4 distinct configurations.[6] If we denote the intersection of the offset pupils by D and the source by S, then these configurations are as follows:

1. S is completely contained within D.
2. S protrudes out of one side of D.
3. S protrudes out of both sides of D.
4. D is completely contained within S.

Clearly, the situation for the general 2D mask is considerably more complicated. Up until now, no published work even enumerated the number of distinct geometrical configurations available; the typical approach to computing the 2D TCC has been to employ the minimum grid that contains the pupils and the source. It will be shown below, however, that, taking into account all possible symmetries, there are 18 distinct geometrical configurations to consider.

We begin by observing that the coordinate system can always be defined such that the symmetry axis of the intersection region D is vertical. By performing this rotation, the myriad of possible geometrical combinations is limited, and only a single vertical coordinate is necessary to specify the TCC. Organizing the different cases is also simplified, as the 2D TCC can now be viewed as a perturbation of the 1D TCC. The various geometries will then be organized by their configuration with the y-components of the frequencies set to zero.

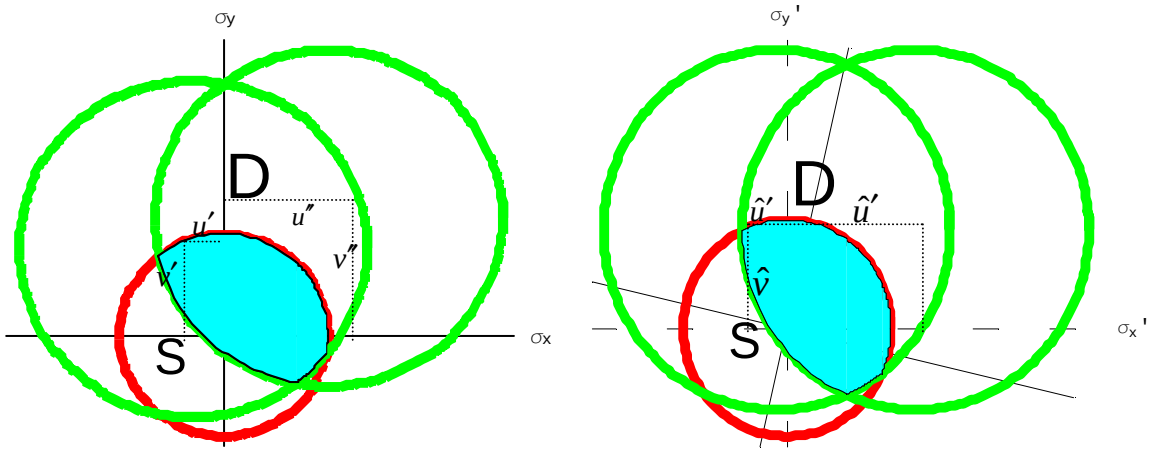


Figure 2 Illustrating the rotation of coordinates in order to produce an intersection region D with a vertical axis of symmetry.

Before enumerating the different cases, the necessary change of coordinates is stated. Let the tangent of the rotation angle be denoted as

$$r = \frac{v' - v''}{u' - u''}, \tag{16}$$

where $u = u_x$ and $v = u_y$. Then the horizontal offsets and the common vertical offset are as follows:

$$\hat{u}' = \frac{u' + rv'}{\sqrt{1+r^2}}, \hat{u}'' = \frac{u'' + rv''}{\sqrt{1+r^2}}, \tag{17}$$

$$\hat{v} = \frac{-ru' + v'}{\sqrt{1+r^2}}. \tag{18}$$

Also, denote $u_1 = \max\{\hat{u}', \hat{u}''\}$, $u_2 = \min\{\hat{u}', \hat{u}''\}$, and $v = \hat{v}$. We consider the changes in geometry as the region D is moved vertically. It is assumed that $|u_1| \geq |u_2|$; the complementary case results in the mirror image about the vertical.

Basically, one must track the vertical motion of the region D through the source, taking note of all critical points beyond which the nature of the integration region changes. The number of logical conditions easily numbers in the hundreds; therefore, an exhaustive description of the algorithm is not possible here. To simplify the programming, it is essential that the unique regions be identified and mapped to the correct set of logical conditions. A listing of the unique domains, and an example of a logical condition that would produce that region, is given in the Appendix, as well as the critical points that define the transitions between regions.

3.2. Simplification via contour integration

The main result of the previous subsection was to identify all of the possible geometrical forms for the region of integrations for the TCCs, and the conditions under which each particular form would occur. The in-focus TCCs are simply the areas of their respective regions. Given that each of the bounding contours of these regions is a circular arc, the determination of the areas exactly in terms of analytic functions is straightforward. All one needs are the locations of the intersection points in order to determine where a change in the integration limits is warranted.

On the other hand, the resulting functions that describe the TCC are, from a programming perspective, rather unwieldy. Even though orders of magnitude efficiency over straightforward numerical integration is still possible, one gets the feeling that further improvements are possible. Such an improvement would result if there were a simple, universal expression that could act as a building block for the computation of the area.

Such a building block is generated by integrating over the contour rather than the area. The mathematical tool for converting between contour and area integrals is Green's Theorem[7]. A particularly simple consequence of Green's Theorem is an expression of the area *A* of a closed region *D* in terms of its bounding contour:

$$A(D)=\int_D d^2\boldsymbol{\sigma}=\frac{1}{2}\oint_{\partial D}\left(\sigma_x d\sigma_y-\sigma_y d\sigma_x\right), \tag{19}$$

Because the closed contour consists of nothing more than a set of circular arcs, the parametrization of the above contour integral is trivial; the final result takes the form

$$A(D)=\frac{1}{2}\sum_{k=1}^{N}\left\{\rho_k^2\left(\phi_2^{(k)}-\phi_1^{(k)}\right)+\rho_k\left[u_0^{(k)}\left(s_2^{(k)}-s_1^{(k)}\right)-v_0^{(k)}\left(c_2^{(k)}-c_1^{(k)}\right)\right]\right\}, \tag{20}$$

where the following notation applies:

- ρ_k is the radius, and $\left(u_0^{(k)},v_0^{(k)}\right)$ is the center of the circle containing the k^{th} arc.
- $\phi_2^{(k)}$ and $\phi_1^{(k)}$ are the upper and lower angles, respectively, subtended by the k^{th} arc at its circle's center.
- $s_j^{(k)}=\sin\phi_j^{(k)},\ c_j^{(k)}=\cos\phi_j^{(k)},\ j\in\{1,2\}.$
- N is the number of arcs in the contour.

See Fig. 3 below for an illustration of the meaning of these symbols. There are two immediate advantages to the contour integrations method:

- The number of unique geometrical configurations shrinks from 18 to 9, simplifying the computer code describing the algorithm.
- The final expression in Eq. (20) is evaluated with a single arctangent and a few square roots per arc, resulting in a more efficient algorithm.

3.3. Implementation in software

The base aerial image algorithm was implemented in C on a 700 MHz Pentium III with 256 MB RAM. This allowed for the storage of slightly fewer than 10^7 TCCs in memory[*]. One must keep in mind, however, that, unless one has a truly periodic mask to simulate, it is prudent to place a background layer around the boundary of the mask region in order to avoid edge effects associated with the periodic assumption. Such a region is typically a few times λ/NA and can easily double the number of TCCs needed in memory[†].

For ease of use, the C program was linked to Matlab[8] via MEX, which is the internal C and FORTRAN compiler Matlab uses to form Matlab functions from the C or FORTRAN code. In principle, one could write the code in Matlab's interpretive language, but the execution time is much faster if the looping is done inside the C code.

The computation of the TCCs is only one of the two components needed to compute aerial images. One also must have a means by which the TCCs, along with parameters describing the mask, are input. Matlab provides the ideal environment in which to read in various mask files describing polygons in a layout and pass them into a structure for storage. A separate program was written, then, in the same fashion as the TCC computation piece[9], that takes in mask data and TCCs and computes the intensity from Eq.(11). The details of this particular computation lie outside the scope of this work and need not be discussed further here[‡].

---

[*] The TCCs were stored as complex values, as they are also used in more general cases involving defocus and aberrations.

[†] Modern simulators built into an EDA engine typically compute intensities at preselected points in a layout, rather than at densely sampled, evenly spaced points throughout the layout. They then select some window around each point of interest and compute the intensity there. The size of the window is typically on the order of 1 μm^2, so that the number of coefficients needed is relatively small.

[‡] The intensity computation contains no approximations and, like SPLAT before it, is $O\left(N^2\right)$. The typical "fast" aerial image algorithm finds the equivalent linearized pupil functions; this means that the intensity is expressed in terms of a truncated series of convolutions, each of which is, worst case, an $O(N\log N)$ computation. By exploiting specific nuances of IC geometries, the intensity computation is even faster. See Y.C. Pati et al, Ref. 5.

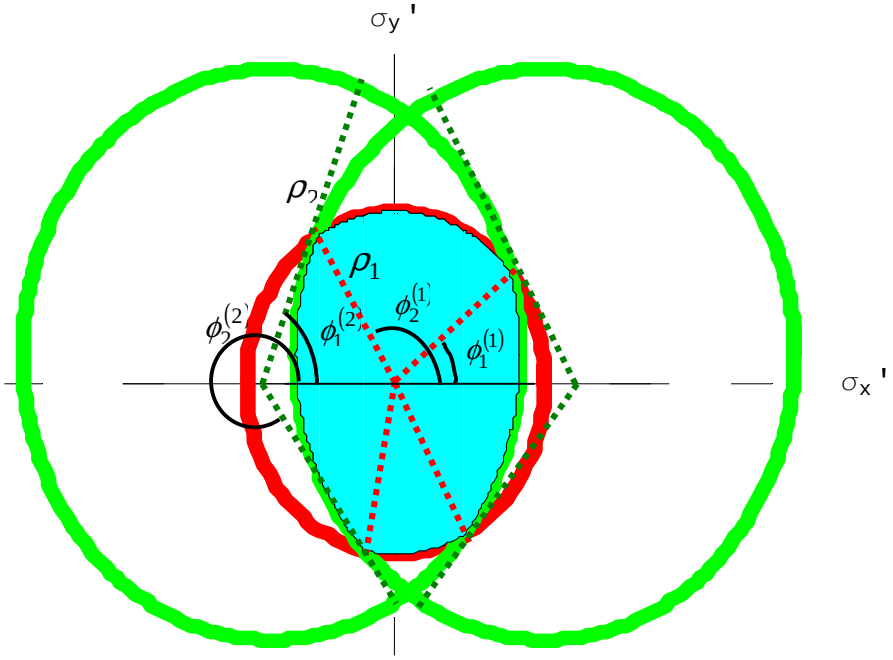


Figure 3 Illustrating the meaning of Eq. (20).

4. NUMERICAL RESULTS

The results discussed here will focus on accuracy, as no absolute test of accuracy for 2D simulation has been offered in the literature, to the author's knowledge. The simplest, nontrivial 2D object is simply a single line of finite length, centered in some unit cell. Consider such an opaque line, 0.15μm X 3μm, centered in a clear periodic unit cell of dimensions 0.3μm X 6μm. We illuminate with wavelength λ=0.19337 μm at σ=0.6 and *NA*=0.68 and 0.75. We ignore the obliquity factor from the reduction lens for the purposes of this calculation[*]. The aerial image generated with the algorithm described in Sec. 3.2 is shown below in Fig. 4.

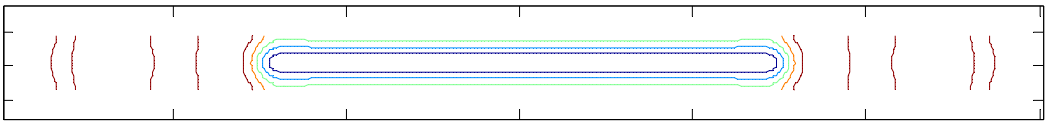


Figure 4 In-focus aerial image due to a single, opaque line.

The computation of this aerial image is then compared with any number of simulation packages currently used in the lithography community today. The error metric ε is given by

$$\varepsilon = \frac{\sum_k \left| I_{\text{exact}}(\mathbf{x}_k) - I_{\text{numerical}}(\mathbf{x}_k) \right|}{\sum_k I_{\text{exact}}(\mathbf{x}_k)}. \tag{21}$$

A simulation tool that computes the aerial image according to the prescription in Section 3.1 above also exists. It employs a 2D adaptive integration scheme that runs until some relative tolerance is achieved, and can take in obliquity, defocus, and aberration parameters. When this scheme was run against the exact scheme with the tolerance set to 0.01, it was noted that ε=0.002 for both *NA*=0.68 and 0.75. This value of the metric will be denoted as the nominal error.

The exact algorithm was then compared with SPLAT[10] and PROLITH[11]. With SPLAT, the error was over an order of magnitude less than the nominal error for both values of the *NA* (0.00010 and 0.00011, respectively). In the case of

[*] This is justified when one considers the angles that are actually seen by the photoresist. Thanks to Tim Brunner of IBM for pointing this fact out.

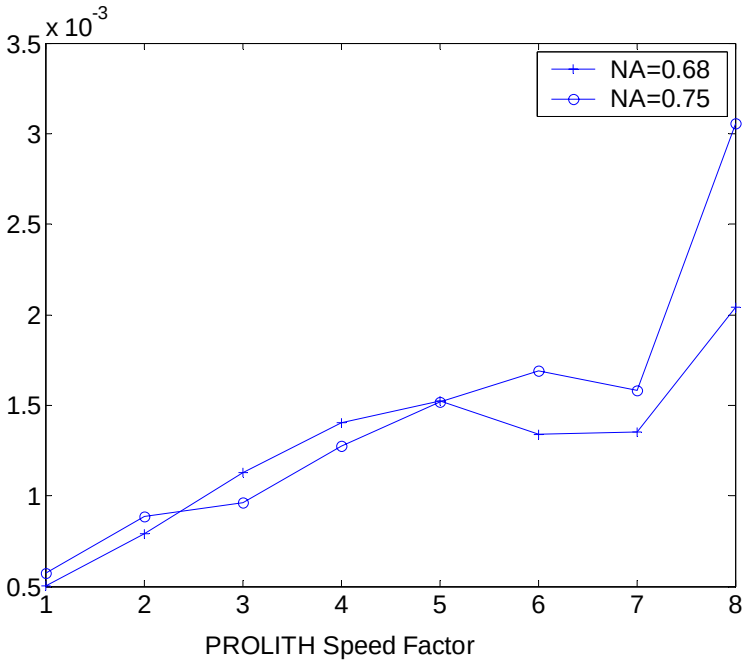


Figure 5 Error metric in Eq. (21) for PROLITH through speed factor for the above simulation. The "+" and "o" annotated lines represent the nominal errors for the respective NAs, which roughly correspond to a worst-case 1% error.

PROLITH, there are 8 speed factors to consider, with "1" being the most accurate and "8" the least. Fig. 5 contains a plot of the error metric, for both *NA* values, through the PROLITH speed factor. As can be seen from this figure, the numerical error here is reasonable for all values of the speed factor. For *NA*=0.68, all speed factors provide a lower error than the nominal case, while for *NA*=0.75, all but the "8" case do.

5. CONCLUDING REMARKS

An exact formulation of the problem of imaging a 2D object through a Kohler illumination system was presented. The main limitation to doing any real layout exactly is not patience in waiting for an integrator to churn through hundreds of thousands of different cases, but is simply memory. The main idea behind the algorithm is that the boundaries of the regions that comprise a typical TCC is made up of circular arcs, and therefore the area – which determines the value of the TCC – should be exactly computable. A change in integration scheme via Green's Theorem leads to expressions with minimal dependence on expensive functions such as arctangents and square roots. The difficult piece of the algorithm is determining the regions of integration; this has to be done brute force, but only once. Rotating the coordinate system provide a crucial simplification at a negligible computational price. In the end, 18 unique regions were found, but this was halved when Green's Theorem was used.

This work serves as a complement to the fast aerial simulators, now in most OPC tools. Where those tools provide the ability to evaluate intensities quickly via a small series of convolutions of specially defined objects at selected points, they still must compute the TCCs to build their libraries. While the coefficients do get stored, a change in any optical or numerical parameter: illumination, pitch/sample region, and, most importantly, defocus necessitates a recomputation of all of the TCCs. While this work does not address defocus, the ability to do so is a viable extension of the formulation proposed here. Such a simulator could provide a powerful means of including multiple defocus criteria in a real OPC design.

ACKNOWLEDGMENTS

Thanks to Martin Burkhardt, Lars Liebmann, Santo Crendedino and Ed Pell (Mentor Graphics Corp.) for help with some simulations. Thanks also to Tim Brunner, Alan Rosenbluth, Gregg Gallatin, and Carlos Fonseca for interesting discussions, to Chris Progler (now with Photronics) for initiating the project that led to this work, and to George Gomba for managerial support.

APPENDIX. THE DISTINCT GEOMETRICAL CONFIGURATIONS OF THE HOPKINS' TRANSMISSION CROSS-COEFFICIENT FOR A CIRCULAR SOURCE AND EXIT PUPIL.

We define the following quantities:

$$v_1 = \sqrt{(1-\sigma)^2 - u_1^2}\ ,\ v_2 = \sqrt{1-(\sigma+u_1)^2}\ , \qquad (22),\ (23)$$

$$v_3 = \sqrt{(1-\sigma)^2 - u_2^2}\ ,\ v_4 = \sqrt{1-(\sigma-u_2)^2}\ , \qquad (24),\ (25)$$

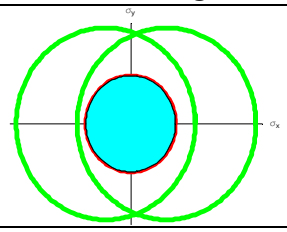
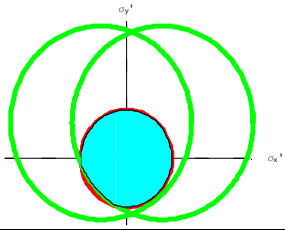
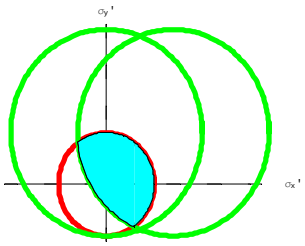
$$v_\alpha = \sqrt{1-\frac{1}{4}(u_1-u_2)^2}\ ,\ v_\beta = \sqrt{\sigma^2-\frac{1}{4}(u_1+u_2)^2}\ , v_5 = v_\alpha - v_\beta\ ,\ v_6 = v_\alpha + v_\beta\ , (26),\ (27),\ (28),\ (29)$$

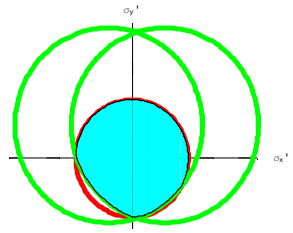
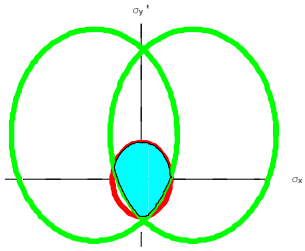
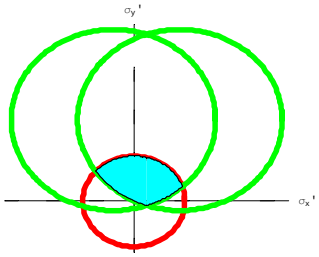
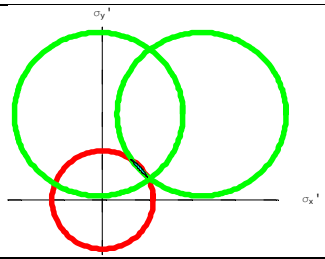
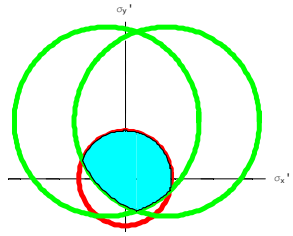
$$v_7 = \sqrt{(1+\sigma)^2 - u_1^2}\ ,\ v_8 = \sqrt{(1+\sigma)^2 - u_2^2}\ . \qquad (30),\ (31)$$

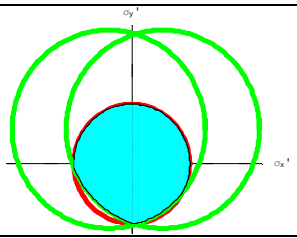
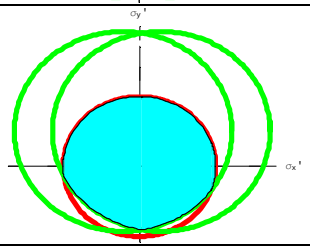
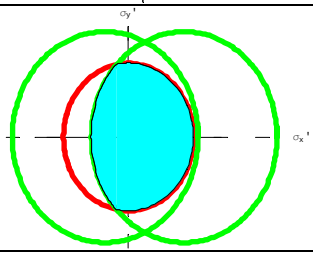
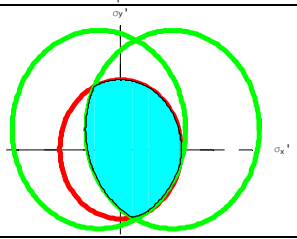
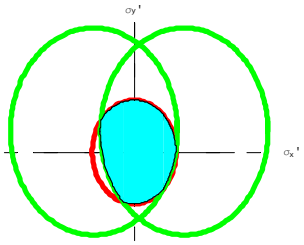
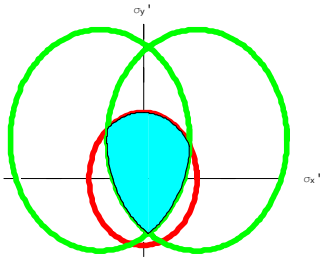
$$v_9 = \sqrt{\sigma^2-(1-u_1)^2}\ ,\ v_{10} = \sqrt{\sigma^2-(1-u_2)^2}\ , v_{11} = \sqrt{\sigma^2-(1+u_1)^2}\ ,\ v_{12} = \sqrt{\sigma^2-(1+u_2)^2}\ \ (32),(33),(34),(35)$$

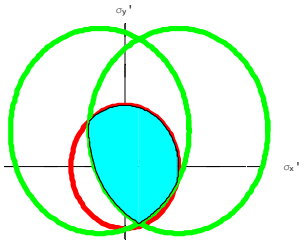
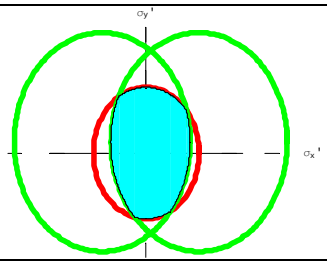
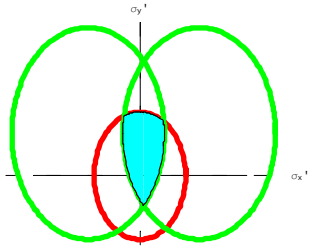
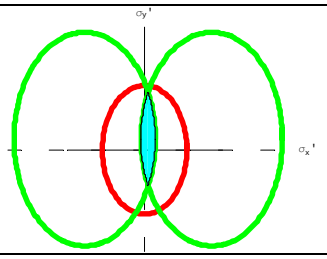
$$v_{13} = \sqrt{1-(\sigma-|u_1|)^2}\ , v_{14} = \sqrt{1-(\sigma-|u_2|)^2} \qquad (36),\ (37)$$

Then the distinct geometrical cases are determined under the following conditions:

#	Geometric Configuration	Kintner (1D) Classification	Example of 2D constraint
1		$u_1 - 1 < -\sigma$ and $u_2 + 1 > \sigma$	$\lvert v\rvert < v_1$
2		$u_1 - 1 < -\sigma$ and $u_2 + 1 > \sigma$,	$u_2 < 0$, $v_2 < v_3$ and $v_1 < \lvert v\rvert < v_2$.
3		$u_1 - 1 < -\sigma$ and $u_2 + 1 > \sigma$,	$u_2 < 0$, $v_2 < v_3$ and $v_2 < \lvert v\rvert < v_3$.
		$u_1 - 1 > -\sigma$ and $u_2 + 1 > \sigma$,	$u_2 < -\frac{1-\sigma}{1+\sigma}u_1$, $v_9 < v_3$, $v_9 < \lvert v\rvert < v_3$.

4		$u_1 - 1 < -\sigma$ and $u_2 + 1 > \sigma$,	$u_2 < 0$, $v_2 < v_3$, $v_4 < v_5$, $v_3 < \lvert v \rvert < v_4$.
		$u_1 - 1 > -\sigma$ and $u_2 + 1 > \sigma$,	$u_2 < -\frac{1-\sigma}{1+\sigma} u_1, v_9 < v_3, v_{10} < v_5$, $v_3 < \lvert v \rvert < v_{10}$.
5		$u_1 - 1 < -\sigma$ and $u_2 + 1 > \sigma$,	$u_2 < 0$, $v_4 < v_5$, and $v_4 < \lvert v \rvert < v_5$.
		$u_1 - 1 > -\sigma$ and $u_2 + 1 > \sigma$,	$u_2 < -\frac{1-\sigma}{1+\sigma} u_1, v_9 < v_3, v_{10} < v_5$, $v_{10} < \lvert v \rvert < v_5$.
		$u_1 - 1 > -\sigma$, $u_2 + 1 < \sigma$, and $\sigma^2 < u_1 u_2 + 1$,	$v_5 > v_{14}, v_5 > v_{13}, v_{13} < \lvert v \rvert < v_5$
6		$u_1 - 1 < -\sigma$ and $u_2 + 1 > \sigma$,	$u_2 < 0, v_4 < v_5$, and $v_5 < \lvert v \rvert < v_6$.
		$u_1 - 1 > -\sigma$ and $u_2 + 1 > \sigma$,	$u_2 < -\frac{1-\sigma}{1+\sigma} u_1, v_9 < v_3$ $, v_{10} < v_5, v_5 < \lvert v \rvert < v_6$.
		$u_1 - 1 > -\sigma$, $u_2 + 1 < \sigma$, and $\sigma^2 < u_1 u_2 + 1$,	$v_5 < v_{14}$, $v_{13} < \lvert v \rvert < v_6$
		$u_1 - 1 > -\sigma$, $u_2 + 1 < \sigma$, and $\sigma^2 > u_1 u_2 + 1$,	$v_{13} < \lvert v \rvert < v_6$
7		$u_1 - 1 < -\sigma$ and $u_2 + 1 > \sigma$,	$u_2 < -\frac{1-\sigma}{1+\sigma} u_1$, and $v_6 < \lvert v \rvert < v_7$.
		$u_1 - 1 > -\sigma$ and $u_2 + 1 > \sigma$,	$u_2 < -\frac{1-\sigma}{1+\sigma} u_1$, $v_9 > v_3$, $v_9 < v_{10}$, $v_5 < v_9$, $v_9 < \lvert v \rvert < v_{10}$.
8		$u_1 - 1 < -\sigma$ and $u_2 + 1 > \sigma$,	$u_2 < 0$, $v_4 > v_5$, and $v_5 < \lvert v \rvert < v_4$.
		$u_1 - 1 > -\sigma$ and $u_2 + 1 > \sigma$,	$u_2 < -\frac{1-\sigma}{1+\sigma} u_1$, $v_9 < v_3$, $v_{10} < v_5$, $v_{10} < \lvert v \rvert < v_5$.

9		$u_1-1<-\sigma$ and $u_2+1>\sigma$,	$u_2>0$, $v_2>v_5$ and $v_5<\lvert v\rvert<v_2$.
10		$u_1-1<-\sigma$ and $u_2+1>\sigma$,	$u_2<0$, $v_2>v_3$, and $v_3<\lvert v\rvert<v_2$.
11		$u_1-1>-\sigma$ and $u_2+1>\sigma$,	$u_2<-\dfrac{1-\sigma}{1+\sigma}u_1$, $v_9<v_3$, $\lvert v\rvert<v_9$.
12		$u_1-1>-\sigma$ and $u_2+1>\sigma$,	$u_2<-\dfrac{1-\sigma}{1+\sigma}u_1$, $v_9>v_3$, $v_9<v_{10}$, $v_5<v_9$, $v_3<\lvert v\rvert<v_5$.
13		$u_1-1>-\sigma$ and $u_2+1>\sigma$,	$u_2<-\dfrac{1-\sigma}{1+\sigma}u_1$, $v_9>v_3$, $v_{10}<v_9$, $v_5>v_{10}$, $v_5>v_9$, $v_3<\lvert v\rvert<v_9$.
		$u_1-1>-\sigma$, $u_2+1<\sigma$, and $\sigma^2<u_1u_2+1$,	$v_5>v_{14}$, $v_5<v_{13}$, $v_{14}<\lvert v\rvert<v_5$
14		$u_1-1>-\sigma$ and $u_2+1>\sigma$,	$u_2<-\dfrac{1-\sigma}{1+\sigma}u_1$, $v_9>v_3$, $v_{10}<v_9$, $v_5>v_{10}$, $v_5<v_9$, $v_5<\lvert v\rvert<v_9$.
		$u_1-1>-\sigma$, $u_2+1<\sigma$, and $\sigma^2<u_1u_2+1$,	$v_5<v_{14}$, $v_{14}<\lvert v\rvert<v_{13}$
		$u_1-1>-\sigma$, $u_2+1<\sigma$, and $\sigma^2>u_1u_2+1$.	$v_{14}<\lvert v\rvert<v_{13}$

15		$u_1 - 1 > -\sigma$ and $u_2 + 1 > \sigma$,	$u_2 < -\frac{1-\sigma}{1+\sigma} u_1$, $v_9 > v_3$, $v_9 < v_{10}$, $v_5 < v_9$, $v_5 < \lvert v \rvert < v_9$. .
16		$u_1 - 1 > -\sigma$, $u_2 + 1 < \sigma$, and $\sigma^2 < u_1 u_2 + 1$,	$v_5 < v_{14}$, $\lvert v \rvert < v_5$.
17		$u_1 - 1 > -\sigma$, $u_2 + 1 < \sigma$, and $\sigma^2 < u_1 u_2 + 1$,	$v_5 < v_{14}$, $v_5 < \lvert v \rvert < v_{14}$
		$u_1 - 1 > -\sigma$, $u_2 + 1 < \sigma$, and $\sigma^2 > u_1 u_2 + 1$.	$v_5 < \lvert v \rvert < v_{14}$
18		$u_1 - 1 > -\sigma$, $u_2 + 1 < \sigma$, and $\sigma^2 > u_1 u_2 + 1$.	$\lvert v \rvert < v_5$

REFERENCES

1. H.H. Hopkins, "On the diffraction theory of optical images", *Proc. Roy. Soc.A* **217**, 408-432 (1953).
2. M. Born and E. Wolf, *Principles of Optics*, 7th Ed. Cambridge: Cambridge Univ. Press (1999), Sec. 8.8.
3. K.K.H. Toh and A.R. Neureuther, "Identifying and Monitoring Effects of Lens Aberrations in Projection Printing", *Proc. SPIE* **772**, 202-209 (1987).
4. N. Cobb and A. Zakhor, "Fast low-complexity mask design", in *Proc. SPIE* **3334**, 313-327 (1995).
5. Y.C. Pati, A.A. Ghazanfarian, and R.F. Pease, "Exploiting Structure in Fast Aerial Image Computation for Integrated Circuit Patterns", *IEEE Trans. Semi. Man.***10**, 62-74 (1997).
6. E.C. Kintner, "Method for the calculation of partially coherent imagery", *Applied Optics* **17**, 2747-2753 (1978).
7. J.E. Marsden and A.J. Tromba, *Vector Calculus*, 3rd Ed. New York: W.H. Freeman and Co. (1988), p. 496.
8. Matlab R12, version 6.0, from The Mathworks, Natick, MA.
9. Alan Rosenbluth, private communication.
10. D. Lee, A.R. Neureuther et al., *SPLAT v5.0 User's Guide*. Berkeley: U. Cal. Press (1995).
11. PROLITH v7.1, KLA-Tencor, San Jose, CA.