Bilinear wavefront transformation

Keith Dillon

Formulens, LLC, 13149 Calle Caballeros, San Diego, California 92129, USA
(kdillon@formulens.com)

Received January 8, 2009; revised May 4, 2009; accepted June 16, 2009;
posted June 26, 2009 (Doc. ID 105930); published July 29, 2009

Truncated expansions such as Zernike polynomials provide a powerful approach for describing wavefront data. However, many simple calculations with data in this form can require significant computational effort. Important examples include recentering, renormalizing, and translating the wavefront data. This paper describes a technique whereby these operations and many others can be performed with a simple matrix approach using monomials. The technique may be applied to other expansions by reordering the data and applying transformations. The key is the use of the vectorization operator to convert data between vector and matrix descriptions. With this conversion, one-dimensional polynomial techniques may be employed to perform separable operations. Examples are also given for differentiation and integration of wavefronts. © 2009 Optical Society of America

OCIS codes: 330.4460, 000.3870, 220.1010, 010.1080, 010.1290, 010.7350.

1. INTRODUCTION

The motivation for the technique at hand is the complexity in processing wavefront data described with polynomial expansions. Zernike polynomials in particular are very useful due to their intuitive description and increasingly standardized usage in optometric and ophthalmic care [1]. But performing simple manipulations such as recentering or scaling the coordinates can require rather sophisticated algorithms, and a growing body of publications address how to perform these and similar tasks using a variety of approaches [2–11]. Noll [2] provide a matrix method to take derivatives of Zernike expansions. Guirao *et al.* [3] describe translation of Zernike expansions with Taylor expansions, and a similar algorithm is included in the 2004 ANSI standard for ophthalmic data [1]. Schwiegerling [4], Campbell [5], Dai [6], Janssen and Dirksen [7], and Shu *et al.* [8] provide algorithms for the scaling of Zernike expansions. Bara *et al.* [9] give a matrix-based approach to achieve scaling, rotation, and displacement for Zernike expansions. Dai in [10] devotes an entire chapter to algorithms for scaling, rotation, and translation of Zernike and other polynomial expansions. Lundström and Unsbo [11] provide perhaps the most direct and unified treatment yet, employing complex matrix descriptions to translate, scale, and rotate data. In this paper we will show how these operations, with the exception of rotation, may all be performed easily through conversion to monomials in a bilinear form. Further, to the list we will add integration, which is extremely useful in modal reconstruction problems, plus essentially any other separable transformation.

Ophthalmic data are usually described with a fairly low number of polynomial terms. Kreuger *et al.* [12] suggest that an eighth-order description (roughly 45 terms) be used for refractive surgical correction. Therefore a matrix approach for manipulation of these terms, perhaps in the form of 45-element vectors and using 45-by-45 element matrices, would still take a negligible amount of processing time on modern computers. Further as we will show, for an expansion in monomials—the atomic terms that make up Zernike and other polynomials—computations that are separable in Cartesian coordinates become very simple.

In Section 2 we will summarize the treatment of expansions as vectors and their conversion in matrix equations. Throughout this paper we will adhere to the following conventions: italic letters refer to scalar variables and scalar functions. Bold lowercase letters denote vectors and vector functions. All vectors are column vectors. Bold uppercase letters denote matrices. Also, because it will help minimize confusion, we will use zero-based indexing of the vector and matrix elements, as is often done in software data structures.

2. BACKGROUND

A truncated polynomial expansion of a wavefront in Cartesian coordinates may be written

$$w(x,y) = \sum_{k=0}^{K} c_k v_k(x,y), \tag{1}$$

where $v_k(x,y)$ is the kth polynomial (or monomial), K is $\frac{1}{2}(N+1)(N+2)$ for an Nth-order polynomial representation of a two-dimensional wavefront, and c_k is the scalar coefficient of the kth monomial. Orthonormality of the set of functions $\{v_k(x,y)\}$ over some domain is convenient for some tasks but not required here. Note that the ordering of expansion terms in k is arbitrary and set by convention.

If the expansion is composed of Zernike polynomials, then the polynomial $v_k(x,y)$ is $Z_k(x,y)$, usually given in polar coordinates as $Z_k(r,\theta)$. And one common ordering convention is to choose $k=\frac{1}{2}\{n(n+2)+m\}$, with Z_k also called $Z_{n,m}$ [1].

If, on the other hand, the expansion is composed of monomials, sometimes described as Taylor monomials [13], or in the image processing literature as geometric moments [14], individual terms are

$$v_k(x,y) = x^i y^j, \tag{2}$$

where i and j are integers. Clearly some $k(i,j)$ ordering must be chosen, and there will be a different set of coefficients from the Zernike expansion. Also note that for an Nth-order polynomial, $i+j \leqslant N$ for all terms. For example, we would refer to the monomial x^2y^2 as fourth-order in total, or simply fourth-order, since its value along the line $x=y$ increases with the fourth power of the radius. There is no standard ordering convention for monomials, but one approach would be to list the terms in order of increasing (total) order. Multiple terms with the same value $i+j$ are ordered with increasing j and decreasing i. For example,

$$w(x,y) = (c_0x^0y^0) + (c_1x^1y^0 + c_2x^0y^1) + (c_3x^2y^0 + c_4x^1y^1 + c_5x^0y^2) + \ldots, \tag{3}$$

where we have all terms where the order $i+j$ equals zero, followed by the terms where $i+j$ equals one, and so on. For multiple terms with the same order, the terms with higher powers of x are first.

This results in an ordering with c_k as the coefficient for the monomial x^iy^j where

$$k = \tfrac{1}{2}(i+j)(i+j+1) + j. \tag{4}$$

In ophthalmic applications, the coordinates are usually normalized to the pupil radius, as Zernikes are defined on the unit disk. It is also worthwhile to do this with monomials for numerical precision reasons for some tasks; if we have a reasonably large image, such that x and y are large at the edge, then the Nth power of these values will be much larger than the Nth power at small values of x and y, and a computer implementation may result in significant errors for applications involving a matrix inversion (such as fitting a polynomial to a surface). While such situations are beyond the scope of this paper, we will assume normalization here as it is also useful to remain consistent with the normalization used in Zernike polynomials. One may explicitly normalize the terms as in

$$v_k(x,y) = \left(\frac{x}{r}\right)^i \left(\frac{y}{r}\right)^j, \tag{5}$$

where r is the pupil radius. Or, as is commonly done, one may assume the coordinates themselves are given in units normalized to the pupil radius, which we will do in this paper.

The set of all polynomials of degree N or less can be treated as a vector space with the particular set of coefficients for a given expansion $\{c_k\}$ described with the column vector $\mathbf{c}$, i.e., $[c_0, c_1, \ldots, c_K]^T$, where T denotes the transpose. Then, Eq. (1) can be written as

$$w(x,y) = \mathbf{c}^T\mathbf{v}(x,y), \tag{6}$$

where $\mathbf{v}(x,y)$ is $[v_0(x,y), v_1(x,y), \ldots, v_K(x,y)]^T$. Zero-based indexing of vector components allows us to be consistent between the exponents and vector indices we use, so that the kth element of $\mathbf{c}$ is simply c_k. Additionally, we must fill in zeros for coefficients that are equal to zero, as the position in the vector must correspond to the correct monomial to which the coefficient applies. For example, the polynomial

$$w(x,y) = 2(x^2+y^2) - 1 \tag{7}$$

is to be viewed as

$$w(x,y) = (-1)x^0y^0 + 0x^1y^0 + 0x^0y^1 + 2x^2y^0 + 0x^1y^1 + 2x^0y^2, \tag{8}$$

corresponding to the coefficient vector $[c_0,c_1,c_2,c_3,c_4,c_5]^T = [-1,0,0,2,0,2]^T$.

Since a Zernike polynomial is simply a linear combination of monomials, a linear transformation exists to convert between the Zernike and monomial expansions as

$$\mathbf{c}_m = \mathbf{T}\mathbf{c}_z, \qquad \mathbf{c}_z = \mathbf{T}^{-1}\mathbf{c}_m, \tag{9}$$

where $\mathbf{c}_z$ is the vector of Zernike coefficients and $\mathbf{c}_m$ is the vector of monomial coefficients.

In fact, any algorithm that can generate Zernike polynomials in Cartesian form is inherently generating the conversion matrix elements. Assuming we want our monomials normalized to the pupil radius, each Zernike polynomial in Cartesian coordinates directly gives a column of $\mathbf{T}$. The conversion matrix for second-order Zernikes would be

$$\mathbf{T} = \begin{bmatrix} 1 & 0 & 0 & 0 & -\sqrt{3} & 0 \\ 0 & 0 & 2 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 2\sqrt{3} & \sqrt{6} \\ 0 & 0 & 0 & 2\sqrt{6} & 0 & 0 \\ 0 & 0 & 0 & 0 & 2\sqrt{3} & -\sqrt{6} \end{bmatrix}, \tag{10}$$

the columns of which may be recognized as the scalars multiplying the monomial terms in each Zernike polynomial. Then given a Zernike expansion of some wavefront, we arrange it into a corresponding column vector and apply $\mathbf{T}$. For example, consider the wavefront consisting of 2.5 μm of sphere. We have $w(x,y) = 2.5Z_4(x,y)$, which corresponds to the polynomial $2.5\sqrt{3}(2x^2+2y^2-1)$. To verify the matrix approach gives this result, we form a vector of the Zernike coefficients which gives $\mathbf{c}_z = [0,0,0,0,2.5,0]^T$. Then we compute $\mathbf{c}_m = \mathbf{T}\mathbf{c}_z$, which equals $[-2.5\sqrt{3},0,0,5\sqrt{3},0,5\sqrt{3}]^T$, the expected answer in vector form. Matlab code to generate this conversion matrix is provided in Appendix A, and the algorithm is derived in Appendix B.

3. BILINEAR FORM

Now we reconsider the ordering of monomial expansions. Similar to how Zernikes are enumerated with n and m, we may enumerate the monomial coefficients with the exponents i and j of the coordinates, so Eq. (3) becomes

$$w(x,y) = c_{0,0}x^0y^0 + c_{1,0}x^1y^0 + c_{0,1}x^0y^1 + c_{2,0}x^2y^0 + c_{1,1}x^1y^1 + c_{0,2}x^0y^2 + \ldots, \quad (11)$$

which, if truncated to include only terms of order $i+j \leq N$, may be written as

$$w(x,y) = \sum_{i=0}^{N} \sum_{j=0}^{N-i} c_{i,j}x^iy^j. \quad (12)$$

Now we define vector functions $\mathbf{x}$ and $\mathbf{y}$ with elements $\mathbf{x}_i = x^i$ and $\mathbf{y}_i = y^i$, and we define $\mathbf{C}$ as a matrix with $\mathbf{C}_{i,j} = c_{i,j}$ for $i+j \leq N$ and zero otherwise. We are essentially zero-padding the coefficients for terms with order $i+j > N$. Recall we are using zero-based indexing of the vector and matrix elements. Now Eq. (12) can be written as

$$w(x,y) = \mathbf{x}^T\mathbf{C}\mathbf{y}. \quad (13)$$

This is the bilinear form of the polynomial expansion. For a second order polynomial, $\mathbf{x} = (1, x, x^2)^T$, $\mathbf{y} = (1, y, y^2)^T$, and

$$\mathbf{C} = \begin{bmatrix} c_{0,0} & c_{0,1} & c_{0,2} \\ c_{1,0} & c_{1,1} & 0 \\ c_{2,0} & 0 & 0 \end{bmatrix}. \quad (14)$$

If we multiply out $\mathbf{x}^T\mathbf{C}\mathbf{y}$ we will get the terms up to second order from Eq. (11).

Now, while the wavefront itself will not be separable in general, its coordinates can be transformed separately in this form. Any linear operator that works exclusively on the space of x or y can be described by a matrix in the basis $\{x^i\}$ or $\{y^i\}$ and be used to derive new coefficients. For example, if we wish to scale the x coordinate, we would produce a matrix that appropriately scaled each member of the vector $\mathbf{x}$, then apply this to the coefficient matrix to get new coefficients in the scaled coordinate system. We compute new coordinates via transformations

$$\mathbf{x}' = \mathbf{D}_x\mathbf{x},$$

$$\mathbf{y}' = \mathbf{D}_y\mathbf{y}, \quad (15)$$

where $\mathbf{D}_\mathbf{x}$ and $\mathbf{D}_\mathbf{y}$ are the x and y transformation matrices. The full expansion in the new coordinates would be

$$w'(x',y') = (\mathbf{x}')^T\mathbf{C}\mathbf{y}' = (\mathbf{D}_x\mathbf{x})^T\mathbf{C}(\mathbf{D}_y\mathbf{y}) = \mathbf{x}^T\mathbf{C}'\mathbf{y}, \quad (16)$$

where we have incorporated the transforms into the coefficients. The expression for the new set of expansion coefficients is therefore

$$\mathbf{C}' = \mathbf{D}_x^T\mathbf{C}\mathbf{D}_y. \quad (17)$$

The elements of $\mathbf{D}$ can be derived from the basic algebra and calculus of one-dimensional polynomials. Let $\mathbf{u} = (u^0, u^1, u^2, \ldots, u^N)^T$ refer to either $\mathbf{x}$ or $\mathbf{y}$. Then for example the rule for differentiation of one-dimensional monomials is simply $\mathrm{d}/\mathrm{d}u\, u^i = iu^{i-1}$. For the vector $\mathbf{u}$ this yields $\mathbf{u}' = (0, 1, 2u, \ldots, Nu^{N-1})^T$, which can be written in matrix form as

$$\mathbf{u}' = \begin{pmatrix} 0 \\ 1 \\ 2u \\ 3u^2 \end{pmatrix} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 \\ 0 & 0 & 3 & 0 \end{bmatrix} \begin{pmatrix} 1 \\ u \\ u^2 \\ u^3 \end{pmatrix} \quad (18)$$

for $N=3$. So we can describe the elements of this matrix as

$$\mathbf{D}_{i,j} = \begin{cases} i, & \text{if } j = i-1 \\ 0, & \text{otherwise} \end{cases}. \quad (19)$$

Matrix elements for translation, scaling, differentiation, and integration matrices are given in Table 1. In all cases the matrices are of size $N+1$ by $N+1$. Matlab implementations of the matrices are given in Appendix A.

Since integration increases the order of the polynomial, the result must be one order higher. Hence we should zero-pad the coefficients up to the next order to avoid potential loss of data, then apply the integration matrix at the higher order. To integrate a second-order coefficient matrix, we zero-pad the coefficients up to third order, so Eq. (14) would be made into

$$\mathbf{C} = \begin{bmatrix} c_{0,0} & c_{0,1} & c_{0,2} & 0 \\ c_{1,0} & c_{1,1} & 0 & 0 \\ c_{2,0} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}. \quad (20)$$

Then the transformation matrix that performs integration on this polynomial is given by

$$\mathbf{D} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & \frac{1}{2} & 0 \\ 0 & 0 & 0 & \frac{1}{3} \\ 0 & 0 & 0 & 0 \end{bmatrix}. \quad (21)$$

For differentiation the result will be one lower in total order, requiring no additional consideration.

Given the example from the previous section of 2.5 μm of sphere, we have as our coefficient matrix

Table 1. Bilinear Transformation Matrices for *N*th Order[a]

Transformation	Matrix Elements (Zero-Based Indexing)
Scaling by α	$\mathbf{D}_{i,j} = \begin{cases} \alpha^i, & \text{if } j=i, \\ 0, & \text{otherwise.} \end{cases}$
Differentiation	$\mathbf{D}_{i,j} = \begin{cases} i, & \text{if } j=i-1, \\ 0, & \text{otherwise.} \end{cases}$
Integration	$\mathbf{D}_{i,j} = \begin{cases} \frac{1}{i}, & \text{if } j=i+1, \\ 0, & \text{otherwise.} \end{cases}$
Translation by a	$\mathbf{D}_{i,j} = \begin{cases} \binom{i}{j}a^{i-j}, & \text{if } i \geq j, \\ 0, & \text{otherwise.} \end{cases}$

[a] i and j run from 0 to N.

$$\mathbf{C} = \begin{bmatrix} -2.5\sqrt{3} & 0 & 5\sqrt{3} \\ 0 & 0 & 0 \\ 5\sqrt{3} & 0 & 0 \end{bmatrix}. \tag{22}$$

If we desire differentiation in the x-coordinate, then we form a differentiation matrix and apply its transpose on the left to get the new coefficient matrix, as in Eq. (17). As there is no transformation for the y coordinate, we simply assume $\mathbf{D}_y$ is the identity matrix in Eq. (17). So we compute $\mathbf{D}^T\mathbf{C}$ as

$$\mathbf{C}' = \begin{bmatrix} 0 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 2 & 0 \end{bmatrix}^T \begin{bmatrix} -2.5\sqrt{3} & 0 & 5\sqrt{3} \\ 0 & 0 & 0 \\ 5\sqrt{3} & 0 & 0 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 \\ 10\sqrt{3} & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \tag{23}$$

which corresponds to $w(x,y)=10\sqrt{3}x$, just as would be expected from taking the derivative with respect to x of 2.5 μm of sphere. To perform differentiation in the y-coordinate, we apply the differentiation matrix on the right to form $\mathbf{CD}$, which is

$$\mathbf{C}' = \begin{bmatrix} -2.5\sqrt{3} & 0 & 5\sqrt{3} \\ 0 & 0 & 0 \\ 5\sqrt{3} & 0 & 0 \end{bmatrix} \begin{bmatrix} 0 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 2 & 0 \end{bmatrix} = \begin{bmatrix} 0 & 10\sqrt{3} & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \tag{24}$$

and which corresponds to $w(x,y)=10\sqrt{3}y$.

Application to wavefront transformations are obvious; translation can be used to recenter the pupil by applying appropriate translation matrices for x and y. Scaling can be used to renormalize the radius by scaling x and y equally. Differentiation can be used to produce synthetic wavefront sensor gradient data, or perhaps in an extremum-finding algorithm. Integration can be used for reconstruction in a sensor that measures wavefront gradients by separately integrating each gradient and appropriately combining the results.

However to actually perform these applications at this stage would require multiple steps as data are converted from Zernike to monomial vector to monomial matrix and back again. Next we will show how to combine these into a single step.

4. VECTORIZATION

In Section 3, we showed how various manipulations may be applied to the coefficient data once it has been manually assembled into the matrix form $\mathbf{C}$. Next we present the use of standard mathematical methods that may handle this bookkeeping in a straightforward fashion. Strictly speaking, interconversion between vectors and matrices, such as the conversion between $\mathbf{c}$ from Eq. (6) and $\mathbf{C}$ from Eq. (13), requires tensor methods [15]. But while that may sound intimidating, there is a well-known function that implements the specific operation we need here, and provides a useful identity, as we shall see.

First, we consider the issue of ordering the matrix elements $\mathbf{C}_{i,j}$ into vector elements $\mathbf{c}_k$. One approach would be to order the elements in terms of increasing total order $(i+j)$ as in Eq. (3). But if instead we simply append the $N+1$ columns of the matrix to form a $(N+1)^2$ long vector, we can employ the *vectorization* function Vec [16]. The coefficient vector resulting from the vectorized coefficient matrix is written as

$$\mathbf{c}^{(0)} = \mathrm{Vec}(\mathbf{C}). \tag{25}$$

For example with a second-order coefficient matrix

$$\mathrm{Vec}\left\{\begin{bmatrix} c_{0,0} & c_{0,1} & c_{0,2} \\ c_{1,0} & c_{1,1} & 0 \\ c_{2,0} & 0 & 0 \end{bmatrix}\right\} = \begin{pmatrix} c_{0,0} \\ c_{1,0} \\ c_{2,0} \\ c_{0,1} \\ c_{1,1} \\ 0 \\ c_{0,2} \\ 0 \\ 0 \end{pmatrix}. \tag{26}$$

The superscript on $\mathbf{c}^{(0)}$ refers to the inclusion of the higher-order terms (as zeros) corresponding to monomials with $i+j>N$, where N is 2 in this example. As a result this vector will be roughly twice as long as the c vector of Eq. (6), as it is $(N+1)^2$ long instead of $\frac{1}{2}(N+1)(N+2)$. We can easily relate $\mathbf{c}^{(0)}$ to $\mathbf{c}$ from Eq. (6) via a matrix $\mathbf{P}$ that discards these unwanted zeros,

$$\mathbf{c} = \mathbf{P}\mathbf{c}^{(0)}. \tag{27}$$

To form $\mathbf{P}$ we simply start with an $(N+1)^2$ by $(N+1)^2$ identity matrix and remove the rows corresponding to the higher-order terms in $\mathbf{c}^{(0)}$. Hence $\mathbf{P}$ will be $\frac{1}{2}(N+1)(N+2)$ by $(N+1)^2$. Further, by permuting the columns of $\mathbf{P}$, we may adapt the method to any ordering of monomials desired, say, based on the ordering our Zernike-to-monomial conversion uses. We note that the ordering of the coefficients in the vector in Eq. (25) can be written as simply

$$k^{(0)} = i + (N+1)j, \tag{28}$$

where $k^{(0)}$ gives the location of the coefficient for monomial x^iy^j, starting from zero. Given the desired ordering of powers i and j for each term, such as given by Eq. (4), we can directly generate $\mathbf{P}$ by making a matrix with ones at locations corresponding to $(k,k^{(0)})$ for each monomial, and zero otherwise. An algorithm to generate $\mathbf{P}$ is given in Appendix A.

In the second-order case, the matrix to discard padded zeros and rearrange coefficients into the increasing-order sequence of Eq. (3) would be

$$\mathbf{P} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix}. \tag{29}$$

Applying this matrix to the vector from Eq. (26) gives $(c_{0,0},c_{1,0},c_{0,1},c_{2,0},c_{1,1},c_{0,2})^T$ as desired. To go in the opposite direction, we would apply $\mathbf{P}^T$. Combining Eqs. (25) and (27) we get the direct interconversion between matrix and vector forms of coefficients,

$$\mathbf{c} = \mathbf{P}\mathrm{Vec}(\mathbf{C}),$$

$$\mathrm{Vec}(\mathbf{C}) = \mathbf{P}^T\mathbf{c}. \tag{30}$$

The key advantage of using the Vec function is that we may easily convert a transform matrix operating on the bilinear form of coefficients into a transform matrix operating on the vector form of coefficients. The matrix representing multiplication in the equivalent vectorized situation can be generated using the well-known Kronecker product $\otimes$. We use the mathematical identity [16]

$$\mathrm{Vec}(\mathbf{QRS}) = (\mathbf{S}^T \otimes \mathbf{Q})\mathrm{Vec}(\mathbf{R}), \tag{31}$$

where $\mathbf{Q}$, $\mathbf{R}$, and $\mathbf{S}$ are matrices. So when we vectorize the result $\mathbf{C}'$ from Eq. (17), the identity gives

$$\mathrm{Vec}(\mathbf{D}_x^T\mathbf{C}\mathbf{D}_y) = (\mathbf{D}_y^T \otimes \mathbf{D}_x^T)\mathrm{Vec}(\mathbf{C}). \tag{32}$$

We may perform multiple transformations at once by defining $\mathbf{D}_x$ and $\mathbf{D}_y$ as the products of the individual transformations. Or we may replace $\mathbf{D}_x$ or $\mathbf{D}_y$ by the $(N+1)\times(N+1)$ identity matrix if that coordinate is not to be transformed at all. Examples are given in Appendix A.

As the left-hand side of Eq. (32) equals $\mathrm{Vec}(\mathbf{C}')$, we may use Eqs. (30) to write the direct transformation for vector coefficients as

$$\mathbf{c}' = \mathbf{P}\mathrm{Vec}(\mathbf{C}') = \mathbf{P}(\mathbf{D}_y^T \otimes \mathbf{D}_x^T)\mathbf{P}^T\mathbf{c} = \mathbf{M}_m\mathbf{c}, \tag{33}$$

where $\mathbf{c}'$ is our transformed coefficient vector. $\mathbf{M}_m$ is the transformation matrix for monomial coefficients in vector form:

$$\mathbf{M}_m = \mathbf{P}(\mathbf{D}_y^T \otimes \mathbf{D}_x^T)\mathbf{P}^T. \tag{34}$$

For example, suppose we want to form the matrix corresponding to a translation of a in x and b in y, for a second-order expansion. The transformation matrix $\mathbf{D}$ from Table 1 is the same form for both coordinates and gives

$$\mathbf{D}_x = \begin{bmatrix} 1 & 0 & 0 \\ a & 1 & 0 \\ a^2 & 2a & 1 \end{bmatrix},$$

$$\mathbf{D}_y = \begin{bmatrix} 1 & 0 & 0 \\ b & 1 & 0 \\ b^2 & 2b & 1 \end{bmatrix}. \tag{35}$$

Since we are transforming both x and y, we would form the matrix

$$\mathbf{D}_y^T \otimes \mathbf{D}_x^T = \begin{bmatrix} \mathbf{D}_x^T & b\mathbf{D}_x^T & b^2\mathbf{D}_x^T \\ 0 & \mathbf{D}_x^T & 2b\mathbf{D}_x^T \\ 0 & 0 & \mathbf{D}_x^T \end{bmatrix}$$

$$= \begin{bmatrix} 1 & a & a^2 & b & ab & a^2b & b^2 & ab^2 & a^2b^2 \\ 0 & 1 & 2a & 0 & b & 2ab & 0 & b^2 & 2ab^2 \\ 0 & 0 & 1 & 0 & 0 & b & 0 & 0 & b^2 \\ 0 & 0 & 0 & 1 & a & a^2 & 2b & 2ab & 2a^2b \\ 0 & 0 & 0 & 0 & 1 & 2a & 0 & 2b & 4ab \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 2b \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & a & a^2 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 2a \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}. \tag{36}$$

The matrix that we would apply to a vector of monomial coefficients is given by Eq. (34). Using Eqs. (36) and (29), this product is

$$\mathbf{M}_m = \begin{bmatrix} 1 & a & b & a^2 & ab & b^2 \\ 0 & 1 & 0 & 2a & b & 0 \\ 0 & 0 & 1 & 0 & a & 2b \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}. \tag{37}$$

To apply this to Zernike coefficients, we would first convert the Zernike coefficients to monomial coefficients via Eq. (9), next apply the transformation of Eq. (37), then finally convert the result back to Zernike coefficients with Eq. (9). So in total we would compute

$$\mathbf{c}_z' = \mathbf{T}^{-1}\mathbf{M}_m\mathbf{T}\mathbf{c}_z = \mathbf{M}_z\mathbf{c}_z. \tag{38}$$

The transformation matrix for a vector of Zernike coefficients therefore, is

$$\mathbf{M}_z = \mathbf{T}^{-1}\mathbf{P}(\mathbf{D}_y^T \otimes \mathbf{D}_x^T)\mathbf{P}^T\mathbf{T}. \tag{39}$$

For the translation of second-order Zernikes we apply $\mathbf{T}^{-1}$ and $\mathbf{T}$ from Eq. (10) to the respective sides of the matrix in Eq. (37) to get

$$\mathbf{M}_z = \begin{bmatrix} 1 & 2b & 2a & 2\sqrt{6}ab & 2\sqrt{3}(a^2+b^2) & \sqrt{6}(a^2-b^2) \\ 0 & 1 & 0 & \sqrt{6}a & 2\sqrt{3}b & -\sqrt{6}b \\ 0 & 0 & 1 & \sqrt{6}b & 2\sqrt{3}a & \sqrt{6}a \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}. \tag{40}$$

If $w(x,y)=c_4Z_4(x,y)$ then the coefficient vector would be $\mathbf{c}_z=[0,0,0,0,c_4,0]^T$. Computing $\mathbf{M}_z\mathbf{c}_z$ yields the transformed coefficient vector

$$\mathbf{c}_z' = \begin{pmatrix} 2\sqrt{3}(a^2+b^2)c_4 \\ 2\sqrt{3}bc_4 \\ 2\sqrt{3}ac_4 \\ 0 \\ c_4 \\ 0 \end{pmatrix}. \tag{41}$$

Performing the transformation by hand, meanwhile, yields

$$\begin{aligned} w'(x,y) &= c_4\{\sqrt{3}[2(x+a)^2+2(y+b)^2-1]\} \\ &= c_4 2\sqrt{3}(a^2+b^2) + c_4 2\sqrt{3}b(2y) + c_4 2\sqrt{3}a(2x) \\ &\quad + c_4\sqrt{3}(2x^2+2y^2-1) \\ &= c_4 2\sqrt{3}(a^2+b^2)Z_0(x,y) + c_4 2\sqrt{3}aZ_1(x,y) \\ &\quad + c_4 2\sqrt{3}bZ_2(x,y) + c_4 Z_4(x,y), \end{aligned} \tag{42}$$

which agrees with Eq. (41).

5. CONCLUSION

A matrix approach to perform linear transforms on an expansion of monomials was described which is very useful in the transformation of wavefront expansions. Examples for scaling, translation, differentiation, and integration were given. The use of permutation matrices and the Kronecker product allows the algorithm to be described in a compact mathematical form, plus provides a means to generate single-step matrices to perform the transformations. Perhaps the most obvious task missing from this approach is rotation of wavefronts. But while rotation is not separable in Cartesian coordinates, it is separable in polar coordinates, suggesting a similar method employing polar monomials $r^n\theta^m$. However rotation is already relatively simple with Zernike polynomials, as it requires only linear combination of the azimuthal frequency terms within each order. Methods to interconvert between polar monomials, Cartesian monomials, and Zernike and other polynomials allow the application to other expansions. Examples using Zernike polynomials were described in detail, and all necessary algorithms are provided in the appendices.

We also note that another use for the bilinear form description of polynomials could be multiplication of polynomials. Just as one-dimensional polynomials may be multiplied via convolution of their coefficient vectors, two-dimensional polynomials may be multiplied via a two-dimensional convolution of their coefficient matrices. Vectorization of this operation is possible, but would require a different approach than used for the linear transformations, so is beyond the scope of this paper.

APPENDIX A

Here we give Matlab code that may perform the included techniques. As Matlab uses typical mathematical indexing (i.e., one-based) of matrix and vector elements, this will also serve to describe the equations in that form. N is used as the order throughout.

Code to generate permutation matrices:

```
P=zeros(1/2*(N+1)*(N+2),(N+1)^2);
for i=0:N,
  for j=0:N-i,
    k=1/2*(i+j)*(i+j+1)+j;
    k0=i+(N+1)*j;
    P(k+1,k0+1)=1;
  end;
end;
```

Code to generate matrix for scaling by alpha:

```
Ds=diag(alpha.^(0:N));
```

Code to generate matrix for translation by a:

```
Dt=zeros(N+1,N+1);
for r=1:N+1,
  for c=1:r,
    Dt(r,c)=a^(r-c)*factorial(r-1)/factorial
    (c-1)/factorial(r-c);
  end;
end;
```

Code to generate differentiation matrix:

```
Dd=diag((1:N),-1);
```

Code to generate integration matrix:

```
Di=diag((1:N).^(-1),1);
```

To pad a coefficient vector (Zernike or monomial) up to the next higher order for integration, we simply append zeros to it as in the following:

```
c=[c;zeros(N+2,1)];
```

To form the vectorized transformation matrix, we first form the matrices for each coordinate. For example if we want to scale then translate by different amounts in both coordinates we would form:

```
Dx=Dt1*Ds1;
Dy=Dt2*Ds2;
```

With only a single-coordinate transformation, such as if we want to integrate in the x direction, we form:

```
Dx=Di;
Dy=eye(N+1);
```

and vice versa for the integration in y. Differentiation is treated similarly. Finally the vectorized transformation matrix can be computed as:

```
Mm=P*kron(Dy.',Dx.')*P.';
```

Given a matrix T that converts monomials to Zernike coefficients, we may form the Zernike transformation matrix as:

```
Mz=inv(T)*P*kron(Dy.',Dx.')*P.'*T;
```

An algorithm for computing T is:

```
T=zeros(1/2*(N+1)*(N+2),1/2*(N+1)*(N+2));
for n=0:N,
  for m=-n:2:n,
    k_z=1/2*(n*(n+2)+m);
    d=m<0;
if m==0, a=sqrt((2*n+2)/2); else a=sqrt(2*n+2);
end;
  for l=0:(abs(m)-d)/2,
      for s=0:(n-abs(m))/2,
        for t=0:(n-abs(m))/2-s,
    b=(-1)*(s+l)*factorial(n-s)/factorial(s) ...
```

```
            /factorial((n+abs(m))/2
-s)/factorial((n-abs(m))/2-s) ...
            *nchoosek(abs(m),2*l
+d)*nchoosek((n-abs(m))/2-s,t);
      i=n-d-2*(s+t+l);
      j=2*l+d+2*t;
      k_m=1/2*(i+j)*(i+j+1)+j
      T(k_m+1,k_z+1)=T(k_m+1,k_z+1)+a*b;
        end;
      end;
    end;
  end;
end;
```

APPENDIX B

We briefly derive the algorithm to produce the Zernike-to-monomial transformation matrix **T**, so that we may demonstrate how to incorporate ordering of the terms.

Starting from the expression for Zernike polynomials in polar coordinates, which we write as

$$Z_{n,m}(x,y) = a_{n,m} u_m \sum_{s=0}^{(n-|m|)/2} (-1)^s \times \frac{(n-s)!}{s!\left(\frac{n+|m|}{2}-s\right)!\left(\frac{n-|m|}{2}-s\right)!} r^{n-2s}, \tag{B1}$$

$$a_{n,m} = \begin{cases} \sqrt{2n+2}, & m \neq 0 \\ \sqrt{\dfrac{2n+2}{2}}, & m = 0 \end{cases}, \tag{B2}$$

$$u_m = \begin{cases} \cos(m\theta), & m > 0 \\ \sin(m\theta), & m < 0 \\ 1, & m = 0 \end{cases}. \tag{B3}$$

We convert to Cartesian coordinates by making the substitutions $r=\sqrt{x^2+y^2}$, $x=r\cos\theta$, $y=r\sin\theta$, and using the expansions

$$\begin{aligned} \sin(m\theta) &= \sum_{l=0}^{(m-1)/2} (-1)^l \frac{m!}{(2l+1)!(m-2l-1)!} \times \cos^{m-2l-1}\theta \sin^{2l+1}\theta \\ &= \sum_{l=0}^{(m-1)/2} (-1)^l \frac{m!}{(2l+1)!(m-2l-1)!} \frac{x^{m-2l-1} y^{2l+1}}{(x^2+y^2)^{m/2}}, \end{aligned}$$

$$\begin{aligned} \cos(m\theta) &= \sum_{l=0}^{m/2} (-1)^l \frac{m!}{(2l)!(m-2l)!} \cos^{m-2l}\theta \sin^{2l}\theta \\ &= \sum_{l=0}^{m/2} (-1)^l \frac{m!}{(2l)!(m-2l)!} \frac{x^{m-2l} y^{2l}}{(x^2+y^2)^{m/2}}, \end{aligned} \tag{B4}$$

which yields

$$Z_{n,m}(x,y) = a_{m,n} \begin{cases} \displaystyle\sum_{l=0}^{|m|/2} (-1)^l \frac{|m|!}{(2l)!(|m|-2l)!} \frac{x^{|m|-2l} y^{2l}}{(x^2+y^2)^{|m|/2}}, & m > 0 \\ \displaystyle\sum_{l=0}^{(|m|-1)/2} (-1)^l \frac{|m|!}{(2l+1)!(|m|-2l-1)!} \frac{x^{|m|-2l-1} y^{2l+1}}{(x^2+y^2)^{|m|/2}}, & m < 0 \\ 1, & m = 0 \end{cases} \times \sum_{s=0}^{(n-|m|)/2} (-1)^s \frac{(n-s)!}{s!\left(\frac{n+|m|}{2}-s\right)!\left(\frac{n-|m|}{2}-s\right)!} (x^2+y^2)^{n-2s/2}. \tag{B5}$$

After algebraic manipulation we may combine terms together into the following expression for generating Zernike polynomials in Cartesian coordinates:

$$Z_{n,m}(x,y) = a \sum_{l=0}^{(|m|-d)/2} \sum_{s=0}^{(n-|m|)/2} \sum_{t=0}^{(n-|m|)/2-s} b x^i y^j,$$

$$a = \begin{cases} \sqrt{2n+2}, & m \neq 0 \\ \sqrt{\dfrac{2n+2}{2}}, & m = 0 \end{cases},$$

$$b = (-1)^{l+s}(n-s)! \left\{ s! \left[\frac{1}{2}(n+|m|)-s\right]! \times \left[\frac{1}{2}(n-|m|)-s\right]! \right\}^{-1} \binom{|m|}{2l+d} \binom{\frac{1}{2}(n-|m|)-s}{t},$$

$$d = \begin{cases} 0, & m \geq 0 \\ 1, & m < 0 \end{cases},$$

$$i = |m| - 2l - d + 2[(n-|m|)/2 - s - t],$$

$$j = 2l + d + 2[(n - |m|)/2 - s]. \quad \text{(B6)}$$

It is evident from Eq. (B6) as we have written it that the Zernike polynomial is simply a sum of monomials. Note that the scalars a, b, d, i, and j are functions.

To form the transformation matrix, we must accumulate the products ab appropriately for the correct powers, for each Zernike polynomial. The indexes of the transformation matrix are $k_z(n,m)$ and $k_m(i,j)$, which are the orderings of Zernike and monomial coefficients discussed in the paper. Given a max order N, we start with an empty matrix $\mathbf{T}$ of size $\frac{1}{2}(N+1)(N+2)$ by $\frac{1}{2}(N+1)(N+2)$ containing zeros, and accumulate the coefficients into the appropriate location, using zero-based indexing of the elements. So in total, we perform the following algorithm, given in pseudocode:

```
for n=0 to N, step=1
  for m=-n to n, step=2
    Compute k_z = (1/2)[n(n+2)+m].
    Compute d from Eq. (B6).
    Compute a from Eq. (B6).
    for l=0 to (|m|-d)/2, step=1
      for s=0 to (n-|m|)/2, step=1
        for t=0 to (n-|m|)/2-s, step=1
          Compute b from Eq. (B6).
          Compute i from Eq. (B6).
          Compute j from Eq. (B6).
          Compute k_m = (1/2)(i+j)(i+j+1)+j.
          Accumulate T_{k_m,k_z} = T_{k_m,k_z} + ab.
        end for t
      end for s
      end for l
  end for m
end for n
```

The computation of the index k_m above incorporates the ordering of the monomial coefficients based on their powers i and j. To use a different ordering simply replace this step. A different Zernike ordering can similarly be incorporated by replacing the computation of k_z.

REFERENCES

1. Z80.28-2004 Methods for reporting optical aberrations of eyes (American National Standards Institute, 2004).
2. R. R. Noll, "Zernike polynomials and atmospheric turbulence," J. Opt. Soc. Am. **66**, 207–211 (1976).
3. A. Guirao, D. R. Williams, and I. G. Cox, "Effect of rotation and translation on an ideal method to correct the eye's higher-order aberrations," J. Opt. Soc. Am. A **18**, 1003–1015 (2001).
4. J. Schwiegerling, "Scaling Zernike expansion coefficients to different pupil sizes," J. Opt. Soc. Am. A **19**, 1937–1945 (2002).
5. C. E. Campbell, "Matrix method to find a new set of Zernike coefficients from an original set when the aperture radius is changed," J. Opt. Soc. Am. A **20**, 209–217 (2003).
6. G.-M. Dai, "Scaling Zernike expansion coefficients to smaller pupil sizes: a simpler formula," J. Opt. Soc. Am. A **23**, 539–543 (2006).
7. A. J. E. M. Janssen and P. Dirksen, "Concise formula for the Zernike coefficients of scaled pupils," J. Microlith. Microfab. Microsyst. Letters **5**, 030501 (2006).
8. H. Shu, L. Luo, G. Han, and J.-L. Coatrieux, "General method to derive the relationship between two sets of Zernike coefficients corresponding to different aperture sizes," J. Opt. Soc. Am. A **23**, 1960–1966 (2006).
9. S. Bara, J. Arines, J. Ares, and P. Prado, "Direct transformation of Zernike eye aberration coefficients between scaled, rotated, and/or displaced pupils," J. Opt. Soc. Am. A **23**, 2061–2066 (2006).
10. G.-M. Dai, *Wavefront Optics for Vision Correction* (SPIE Press, 2008).
11. L. Lundström and P. Unsbo, "Transformation of Zernike coefficients: scaled, translated, and rotated wavefronts with circular and elliptical pupils," J. Opt. Soc. Am. A **24**, 569–577 (2007).
12. R. R. Kreuger, R. A. Applegate, and S. M. McRae, *Wavefront Customized Visual Correction: The Quest for Super-vision II* (Slack, 2004).
13. G.-M. Dai, "Wavefront expansion basis functions and their relationships," J. Opt. Soc. Am. A **23**, 1657–1668 (2006).
14. C.-H. Teh, "On image analysis by the methods of moments," IEEE Trans. Pattern Anal. Mach. Intell. **10**, 496–513 (1988).
15. S. Roman, *Advanced Linear Algebra*, 3rd ed. (Springer, 2008).
16. W.-H. Steeb, *Matrix Calculus and Kronecker Product with Applications and C++ Programs* (World Scientific, 1997).