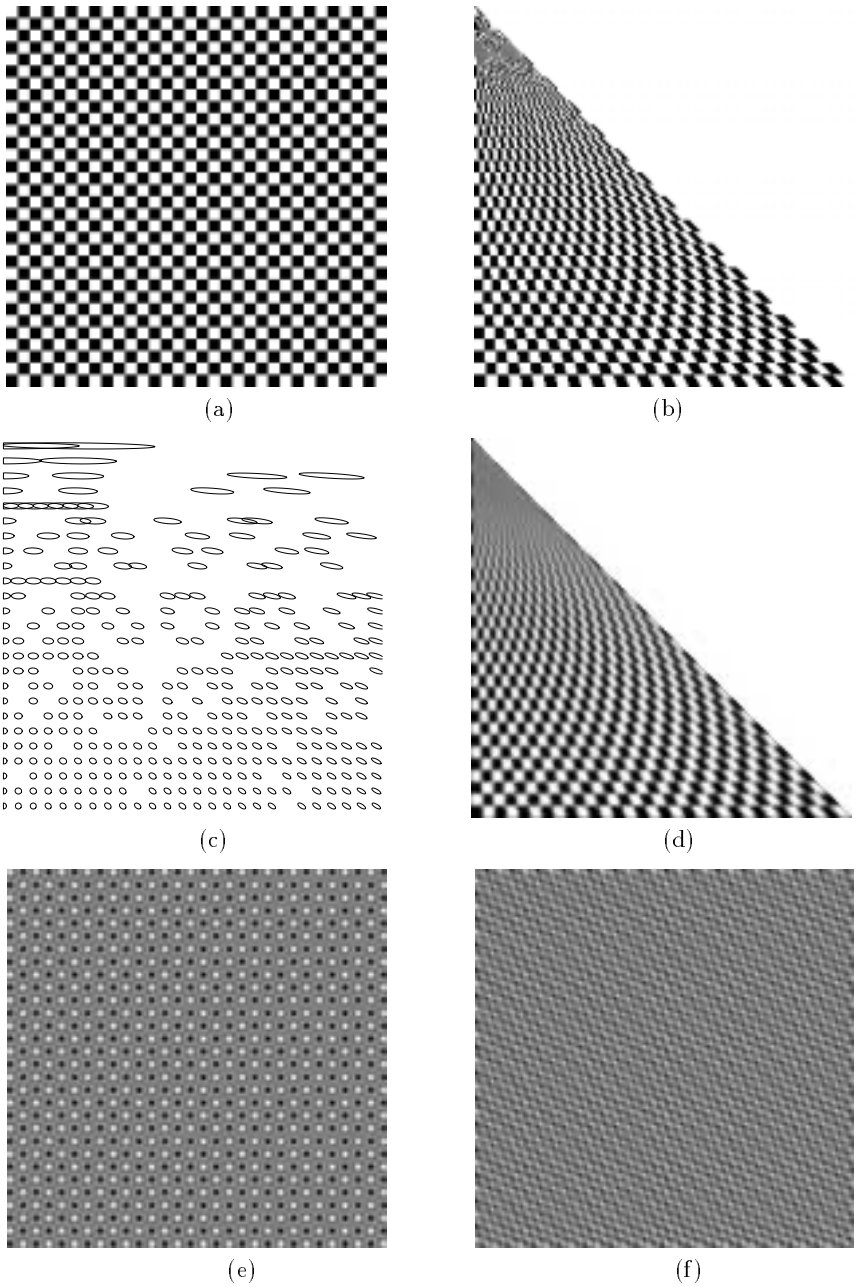


Figure 6: *Using Gaussian filtering to warp an image:* **(a)** *A sample* 256×256 *pixel black and white checkerboard image. Each colored square in the image has* 8×8 *pixels.* **(b)** *Point sampled warp of (a) with warp functions* $x(u,v) = 256u/v$, $y(u,v) = v$. **(c)** *Gaussian kernel shapes required to filter the warp at some texture positions. The ellipses are tri-parametric -*(a,b,θ) *deformations of a circle of unit-pixel radius. Note that the smaller ellipses (in the lower rows) are significantly rotated from an axis-aligned orientation.* **(d)** *Warp of (a) with Gaussian constant-time filtering using our method. Kernels with parameters outside the implemented range were reduced to the implemented parameter ranges.* **(e-f)** *Prefiltered versions of the image (a) with the basis kernels (a.4) and (a.8) of Fig. 3, used to obtain (d). The borders, always problematic when filtering with kernels with large support, have been cut away.*

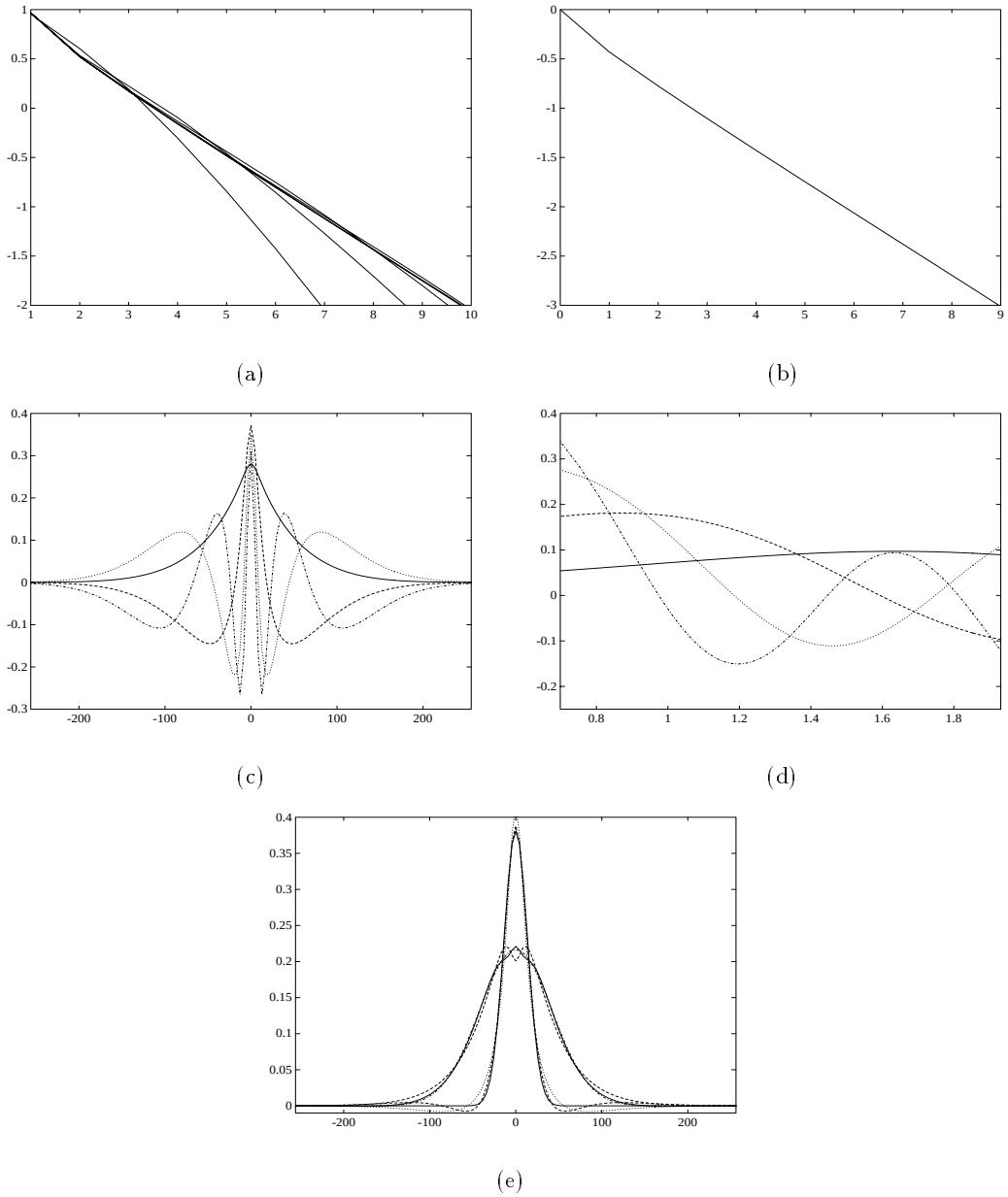
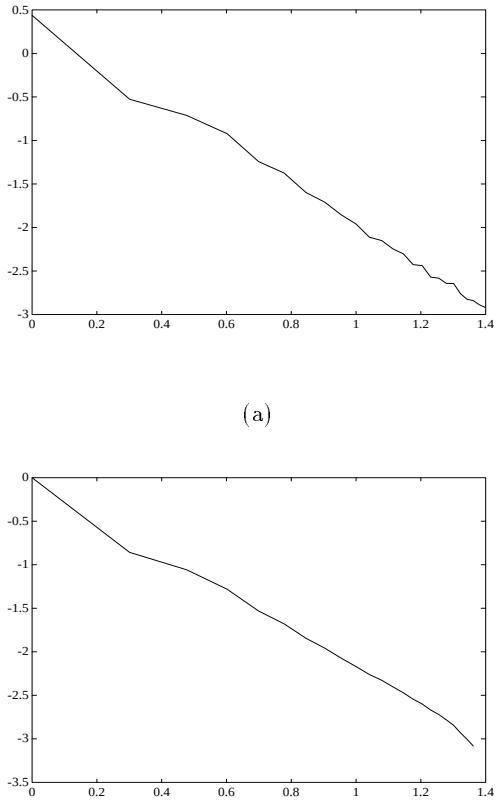


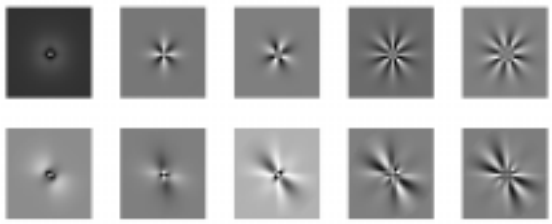
Figure 3: *Approximation of one-dimensional origin-centered Gaussians $G_w(x)$ with variance $w \in [5, 85]$ on the interval $x \in [-256, 256]$ obtained by numerical SVD. The intervals were discretized on a $n \times n$ mesh for the values $n = 20, 40, .., 120$ and SVD applied to the resulting $n \times n$ matrix: $M_{ij} = G_{w_i}(x_j)$.* **(a)** *The logarithms of the first 10 singular values $\{\sigma_i\}_{i=1}^{10}$ for the various n (the higher curves belong to larger n). Exponential decay is evident. Convergence is achieved rapidly, so $n = 80$ was adopted as the working point.* **(b)** *Logarithms of the approximation error e_k obtained when neglecting the (smaller) singular values above a given index k. This decreases exponentially. A 8% approximation error is achieved by retaining only the first three singular values, and a 4% error by retaining the first four.* **(c)** *First four basis functions $\{B_i(x)\}_{i=1}^{4}$.* **(d)** *First four coefficient functions $\{c_i(w)\}_{i=1}^{4}$ on a logarithmic scale.* **(e)** *True and approximated Gaussians G_{14} and G_{42} obtained from the first three and first four basis functions. G_w is obtained as the weighted average of the $B_i(x)$ of (b) reading the weights off the curves in (d) for the appropriate w.*



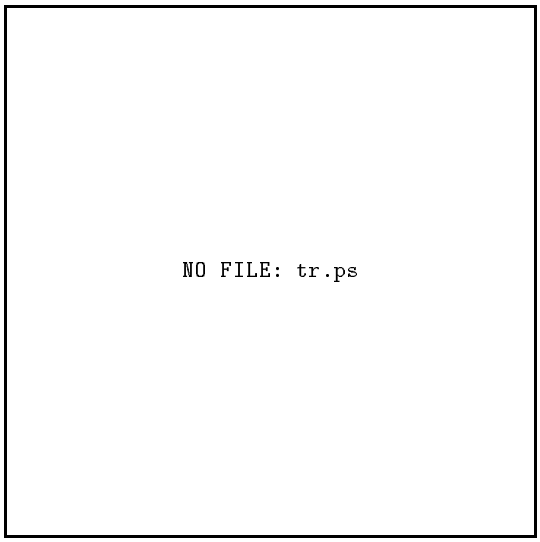
(a)

(b)

Figure 4: *SVD of rotated versions of the 2D axis-aligned kernel obtained from the tensor product of the first basis function of Fig. 1(c) with itself.* **(a)** *First 25 singular values on a log-log axis.* **(b)** *Approximation error e_k as a function of singular value truncation above index k on a log-log axis. The decay is $e_k = O(1/k^2)$. A 4% error is achieved after only five values.*



(a)



(b)

Figure 5: *SVD of Gaussian kernel family $G_{[a,b,\theta]}(x,y)$ in the range $a,b \in [5,85]$, $\theta \in [0,2\pi]$, $x,y \in [-256,256]$:* **(a.1-a.5)** *Basis kernels obtained from decomposition by rotation parameter θ of the tensor product of the second basis function of Fig. 1(c) with itself. Note the radial symmetries.* **(a.6-a.10)** *Basis kernels obtained from decomposition by rotation parameter θ of the tensor product of the first and third basis functions of Fig. 1(c). Note the lack of radial symmetries.* **(b)** *Visual comparison of true Gaussian kernels and their approximations:* **(b.1-b.3)** *True $G_{[30,30,0]}$, $G_{[20,40,0]}$, $G_{[10,40,\pi/4]}$.* **(b.4-b.6)** *Approximations of (b.1-b.3) using 80 (=4x4x5) basis kernels. Approximation error is less than 5%.* **(b.7-b.9)** *Approximations of (b.1-b.3) using 81 (=3x3x9) basis kernels. Approximation error is less than 10%.*

14. W. Press, B. Flannery, S. Teukolsky, and W. Vetterling. *Numerical Recipes in C*. Cambridge University Press, 1988.

15. E.P. Simoncelli, W.T. Freeman, E.H. Adelson, and D.J. Heeger. Shiftable multiscale transforms. *IEEE Transactions on Information Theory*, 38(2):587–607, 1992.

16. L. Williams. Pyramidal parametrics. *Computer Graphics*, 17(3):1–11, 1983.

kernels used for the entire Gaussian family. Fig. 5(b) presents a visual comparison of some Gaussian kernels from the family and their approximations using (c) 4x4x5=80 basis functions and (d) 3x3x9=81 basis functions. The approximation error is less than 5% for the first set, and less than 10% for the second. Compare this to errors of $60\% - 70\%$ achieved by classic constant-time filter approximations (see Fig. 1). Fig. 6 shows (a) an image, (b) its warped counterpart using simple point sampling, and (d) Gaussian filtering with our methods. Fig. 6(c) shows the kernel shapes required to filter some selected image points, computed from the partial derivatives of the warping function.

7. Conclusion

We have achieved constant-time space-variant filtering for arbitrary parametric families of kernels. This has been demonstrated with the useful family of Gaussian kernels, and is applicable to many other families. In fact, these techniques were originally used on derivatives of Gaussians and other kernels for edge-detection purposes in computer vision applications. The decomposition method and kernel basis calculation is not complex, implemented by us as a 50 line MATLAB program. The number of basis kernels required for the approximation depends on the symmetries present in the kernel family, relative to each of the decomposition parameters. For example, kernels which exhibit approximate circular symmetry require less basis kernels when decomposing by a rotation parameter. In any case, there is an explicit tradeoff between the number of basis kernels required and the accuracy of the approximation.

Our method prescribes precomputing the inner products of the original image (texture) with the basis kernels, so if the family of kernels is decomposed by *all* its parameters, only $O(k)$ space is required, where k is the number of resulting basis kernels. If some of the parameters are retained (as the translation parameters were in the example of Fig. 5), the space requirements are multiplied by the number of different possible values in the undecomposed parameter domains. In both cases, the filtering operation requires $O(k)$ time.

When kernels with large supports are required, as frequently happens when the image is significantly contracted (e.g. the horizon area of a perspective texture mapping operation), the proposed method can result in significant savings in filtering time, with excellent quality results.

Acknowledgements: I thank Pietro Perona for helpful comments and friendly advice. This research was partially supported by the Fund for the Promotion of Research at the Technion.

References

1. J. Canny. A computational approach to edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 8:679–698, 1986.
2. F.C. Crow. Summed-area tables for texture-mapping. *Computer Graphics*, 18(3):207–212, 1984.
3. C.L. DeVito. *Functional Analysis and Linear Operator Theory*. Addison-Wesley, 1990.
4. L.A. Ferrari, P.V. Sankar, and J. Sklansky. Efficient two-dimensional filters using B-spline functions. *Computer Vision, Graphics, and Image Processing*, 35:152–169, 1986.
5. A. Fournier and E. Fiume. Constant-time filtering with space-variant kernels. *Computer Graphics*, 22(4):229–238, 1988.
6. W.T. Freeman and E.H. Adelson. The design and use of steerable filters. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(9):891–906, 1991.
7. A. Glassner. Adaptive precision in texture mapping. *Computer Graphics*, 20(4):297–306, 1986.
8. N. Greene and P.S. Heckbert. Creating raster Omnimax images from multiple perspective views using the elliptical weighted average filter. *IEEE Computer Graphics and Applications*, 6(11):21–29, 1986.
9. P.S. Heckbert. Filtering by repeated integration. *Computer Graphics*, 20(4):315–321, 1986.
10. P.S. Heckbert. Survey of texture mapping. *IEEE Computer Graphics and Applications*, 6(11):56–67, 1986.
11. R.A. Horn and C.R. Johnson. *Matrix Analysis*. Cambridge University Press, 1990.
12. P. Perona. Steerable-scalable kernels for edge detection and junction analysis. *Image and Vision Computing*, 10(10):663–672, 1992, Also in Proceedings of European Conference on Computer Vision, 1992.
13. D. Porter and D.S. Stirling. *Integral Equations*. Cambridge University Press, 1990.

plified due to the fact that the unrotated Gaussian $G_{[a,b,0,0,0]}$ is $x-y$ separable:

$$G_{[a,b,0,0,0]}(x,y) \;=\; G_a(x)G_b(y)$$

Using (15), we then have

$$G_{[a,b,0,0,0]}(x,y) \quad (20)$$

$$\approx \left[\sum_i c_i(a)B_i(x)\right]\left[\sum_j c_j(b)B_j(y)\right] \quad (21)$$

$$= \sum_{i,j}\left[c_i(a)c_j(b)\right]\left[B_i(x)B_j(y)\right] \quad (22)$$

$$= \sum_{i,j} c_{ij}(a,b)B_{ij}(x,y) \quad (23)$$

after renaming the basis and coefficient functions in an obvious way. The rotated Gaussian $G_{[a,b,\theta,0,0]}$ is obtained by applying the rotation operator R_θ to the plane:

$$G_{[a,b,\theta,0,0]}(x,y) = G_{[a,b,0,0,0]}(R_{-\theta}(x,y))$$

Since R_θ is linear, substituting in (23),

$$G_{[a,b,\theta,0,0]}(x,y) \;\approx\; \sum_{i,j} c_{ij}(a,b)B_{ij\theta}(x,y)$$

where $B_{ij\theta}$ is B_{ij} rotated by θ, so that the basis kernels for $G_{[a,b,\theta,0,0]}$ are just the basis kernels for $G_{[a,b,0,0,0]}$ rotated by θ. For fixed i and j consider the continuum of basis kernels obtained by taking θ to be any angle in the range $[\theta_{min},\theta_{max}]$. Again using the SVD, it is now possible to construct a small number of basis kernels which approximately span this continuum of kernels. Renaming again the basis and coefficient functions, we finally obtain:

$$G_{[a,b,\theta,0,0]}(x,y) = \sum_{i,j,k} c_{ijk}(a,b,\theta)B_{ijk}(x,y) \quad .$$

Theoretically, it is possible to continue to separate the two translation parameters, but this results in too many basis functions to be practical. We chose to use the family B_{ijk} *translated in the plane* instead. This keeps the number of basis functions managable, but increases the storage requirements, as now all inner products $\sum_{u,v} I(u,v)B_{ijk}(x_0+u)(y_0+v)$ must be stored for each possible pixel center (x_0,y_0). The storage requirements sum to $O(k)$ times the original image size.

6. Experimental Results

Greene and Heckbert [8] advocate the use of Gaussian kernels for approximating kernels arising during filtering in typical image warping scenarios. They provide a simple algorithm for this space-variant scheme for which the filtering time is proportional to the number of texture pixels in the effective kernel support, therefore not constant-time. We apply our methods to perform this filtering in constant time, at the expense of additional memory. Assume we are warping the plane according to the coordinate functions $x(u,v)$ and $y(u,v)$. When determining the value of a pixel (x,y) in the warped image (= image space), it is possible to *point sample* the original image (= texture space) by inverting x and y, and assigning to the image space pixel the value of the texture space pixel $(u(x,y),v(x,y))$. The quality of the result may be improved by computing the (x,y) image space pixel value from texture pixels in the neighborhood of (u,v), especially if $x(u,v)$, $y(u,v)$ are contractive. This is a filtering operation, where a Gaussian kernel is recommended. The three parameters (a,b,θ) of the appropriate origin-centered Gaussian filter kernel to be applied to the texture may be calculated from the partial derivatives u_x,u_y,v_x,v_y at the point (u,v), and translated to (u,v). For details of the calculations, see [8].

Using the methods developed in the previous section, we approximated the 2D origin-centered Gaussian kernels of (16) in the parametric ranges

$$a\in[5,85], \qquad b\in[5,85], \qquad \theta\in[0,2\pi]$$

which is sufficient for many practical applications. For smaller widths it is easier to compute the inner product directly in image space. Seeing that four basis functions give an excellent approximation to the one-dimensional Gaussians (see Fig. 3), so 16 (=4x4) basis kernels suffice to approximate well any axis-aligned two-dimensional Gaussian, as in (23). When rotating the Gaussians, their inherent symmetry allows us to consider only the parameter range $\theta\in[0,\pi/2]$ and use the relations:

$$G_{[a,b,\theta]} = G_{[b,a,\theta+\pi/2]} = G_{[a,b,\theta+\pi]} = G_{[b,a,\theta+3\pi/2]} \quad .$$

We found that rotation is more difficult to separate than scaling, requiring at least five basis kernels to decompose a function satisfactorily by this parameter. Fig. 4 explains this, showing that in a typical scenario including rotated versions of a given (basis) kernel, the decay of the singular values is only polynomial in k. A 4% approximation error is achieved by retaining the five largest singular values. Fig. 5(a) shows some of the basis

sume x is a variable on the continuous domain $[x_{min}, x_{max}]$ and y a variable on the continous domain $[y_{min}, y_{max}]$. Sample both domains at n equally spaced intervals, forming the vectors $(x_1, .., x_n)$ and $(y_1, .., y_n)$. From these form the $n \times n$ matrix $M_{ij}^{(n)} = f(x_i, y_j)$. Using matrix SVD (which may be computed numerically), as in (13), write $M^{(n)} = U^{(n)T} S^{(n)} V^{(n)}$. The rows of $U^{(n)}$ and $V^{(n)}$ are the discrete versions of functions $g_i(x)$ and $h_i(y)$ respectively. Now we have separated $f^{(n)}$ - a discrete version of f - to the sum of n outer products. The separation of $f^{(n)}$ can be extended to the continuous f by increasing n indefinitely, i.e. sampling the intervals more densely. In this case, the limits $\lim_{n\to\infty} S_{ii}^{(n)}$ exist for all i and form the discrete spectrum of f. In practice, n is taken to be sufficiently large for convergence to within a desired tolerance to have been achieved. The resulting discrete functions $g_i(x)$ and $h_i(y)$ may be extended to any $x \in [x_{min}, x_{max}]$ and $y \in [y_{min}, y_{max}]$, respectively, by any reasonable interpolation scheme, resulting in (14) (this may inject an insignificant error into the separation). The error of the approximation obtained by truncating to a finite k, as in (12), is exactly that of the matrix approximation error, defined by (11).

4. Approximating Gaussians

In this section (in preparation for the 2D case) we show how to approximate the one-parameter family of zero-centered Gaussians of varying widths as a linear combination of a small number of basis functions. This method is completely general, applicable in an analogous fashion to any other parametric family of one dimensional functions. Denote

$$G_w(x) = \frac{1}{\sqrt{2\pi}w} \exp\left[-\frac{1}{2}\left(\frac{x}{w}\right)^2\right]$$

We would like to be able to write:

$$G_w(x) \approx \sum_{i=1}^{k} c_i(w) B_i(x) \qquad (15)$$

for a "small" k. If the parameter w is allowed to range over the infinite interval $(0, \infty)$, this is impossible, as $G_w(x)$ is not compact. However, in practical applications we are interested in the compact intervals $x \in [x_{min}, x_{max}]$ and $w \in [w_{min}, w_{max}]$. Consider $G_w(x)$ as a function of *two* variables: w and x, and apply the numerical procedure described in the previous section for increasing values of n, until convergence to an acceptable tolerance is achieved.

Fig. 3 shows the results of this method applied to the approximation of zero-centered Gaussians in the ranges $w \in [5, 85]$, $x \in [-256, 256]$. The intervals were discretized to a $n \times n$ mesh for the values $n = 20, 40, ..., 120$, and SVD performed on the resulting $n \times n$ matrix $M_{ij} = G_{w_i}(x_j)$. It is clearly seen that convergence to the "true" singular values is achieved already at $n = 80$, and that their decay, and resulting approximation error, is expontential in k. A 8% approximation error is achieved for $k = 3$ and a 4% approximation error for $k = 4$. In terms of visual impression, the reconstruction of Gaussians of various widths with three basis functions leaves something to be desired, but four give excellent results.

5. Approximating Two-Dimensional Kernels

Filtering images in many graphics applications requires the use of two-dimensional space-variant kernels. In most applications, the kernels used are deformations of a "canonical" kernel. The space of possible deformations is usually governed by a small number of deformation parameters. For example, a useful parametric family of kernels are the two-dimensional Gaussians, governed by the five parameters $[a, b, \theta, x_0, y_0]$ - describing the operations to be performed on the origin-centered unit-variance Gaussian in order to form the kernel: scaling (a, b), rotation (θ) and translation (x_0, y_0):

$$G_{[a,b,\theta,x_0,y_0]}(x, y) = \frac{1}{2\pi ab} \qquad (16)$$

$$\exp\left(-\frac{1}{2}\left[\frac{(x - x_0)\cos\theta - (y - y_0)\sin\theta}{a}\right]^2\right) \qquad (17)$$

$$\exp\left(-\frac{1}{2}\left[\frac{(x - x_0)\sin\theta + (y - y_0)\cos\theta}{b}\right]^2\right) \qquad (18)$$

In practical applications, we are usually interested only in the kernels whose parameters are taken from compact sets. We would like to approximate the Gaussians well as a linear combination of a small number k of "basis" kernels B_i:

$$G_{[a,b,\theta,x_0,y_0]}(x, y) \approx \sum_{i=1}^{k} c_i(a, b, \theta, x_0, y_0) B_i(x, y) \quad . \qquad (19)$$

Multi-parameter SVD involves the construction of the basis kernels by introducing the parameters one at a time, performing the procedure described in Section 4 at each stage. For the Gaussian kernels, the procedure may be somewhat sim-

ordered in decreasing value:

$$\sigma_1 \geq \sigma_2 \geq ... \geq \sigma_n \quad .$$

Furthermore, assume that there exists a $1 \leq k \leq n$ such that the magnitudes of $\sigma_{k+1}, .., \sigma_n$ are negligible relative to those of $\sigma_1, .., \sigma_k$. In this case, a good approximation for M is obtained by assuming $\sigma_{k+1} = \sigma_{k+2} = .. = \sigma_n = 0$, hence

$$M \approx M_1 = U_1^T S_1 V_1 \qquad (10)$$

where U_1 and V_1 are $k \times n$ matrices and S_1 a $k \times k$ diagonal matrix, obtained from U, V and S respectively. This is another way of saying that M may be approximated well as a sum of the type (7) of only k terms. The faster the decay of the singular values, the more terms may be neglected, yielding a more compact approximation. The rate of decay of the singular values is intimately related to the smoothness of M, namely, a smoother matrix results in more rapid decay (see e.g. [13], Chap. 5 for results of this kind).

Not only is the SVD a tool for decomposing matrices, but it is also optimal in the following sense: For a given k, no other k vector pairs u_i, v_i yield a better approximation to M as a sum of the type (7), measured by the $e()$ function (defined in (5)) than those obtained from the SVD of M by setting $\sigma_i = 0$ for $k < i \leq n$. Using (9), it may be shown ([11] p. 436) that the approximation error is then:

$$e(M_1, M) = \left[\frac{\sum_{i=k+1}^{n} \sigma_i^2}{\sum_{i=1}^{n} \sigma_i^2} \right]^{1/2} \quad , \qquad (11)$$

and for any other $M_2 = U_2^T S_2 V_2$ with the correct matrix characteristics (for U_2, S_2 and V_2)

$$e(M_2, M) \geq e(M_1, M) \quad .$$

3.2. Separating Functions Numerically with the SVD

The SVD technique, described in the previous section, may be used for *separating* functions of two continuous variables. A bivariate function f is *separable* if $f(x, y) = g(x)h(y)$. In general, very few functions are separable. An extension of this simple form of separability is the sum

$$f(x, y) = \sum_{i=1}^{k} \sigma_i g_i(x) h_i(y) \quad . \qquad (12)$$

The *spectral theorem* of functional analysis ([3], Chap. 5) provides the conditions under which this is possible, namely, that f may be expressed as a *discrete* sum, a continuous integral not being neccessary. It requires that f be *compact*, i.e. that $f \in L^2$ and the domains of x and y be compact. In this case, k is either finite or infinite such that $\lim_{k \to \infty} \sigma_k = 0$. Any one of the singular values $\{\sigma_i\}_{i=1}^{k}$ may be computed by solving integral eigenvector-eigenvalue equations. A finite k is achieved (among other cases) if the domains of x and y are discrete, i.e. x and y assume only a finite number of values. Indeed, assume $f(x, y)$ is a function of the discrete variables x and y, each of which could take one of a set of possible values, $\{x_1, .., x_n\}$ and $\{y_1, .., y_m\}$ respectively. Form the $n \times m$ matrix F with entries $F_{ij} = f(x_i, y_j)$. Use the matrix SVD (as in the previous section) to write

$$F = U^T S V \qquad (13)$$

or, equivalently:

$$f(x, y) = \sum_{i=1}^{k} \sigma_i g_i(x) h_i(y) \qquad (14)$$

where g_i and h_i are the i'th rows of U and V respectively, and $k = \min(m, n)$. This is the separated form of f.

For the case of compact f with continous domain, it is possible to approximate the discrete spectrum of the function by suitable discretization of the domain, and use of matrix SVD. As-

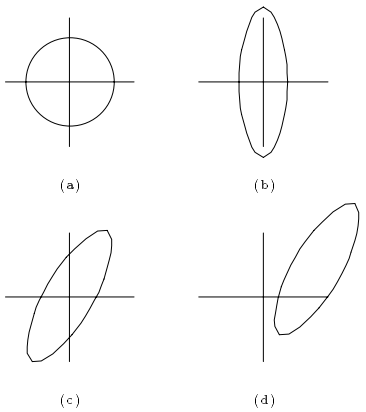
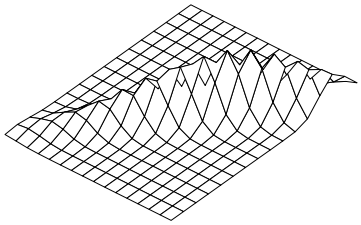
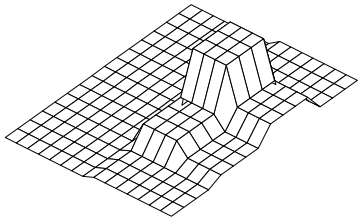


Figure 2: *Constructing an arbitrary Gaussian filter from the canonical one by deformation of an origin-centered circle to an arbitrary ellipse in the plane.* **(a)** *The canonic origin-centered circle.* **(b)** *Scaled version of (a).* **(c)** *Rotated version of (b).* **(d)** *Translated version of (c).*

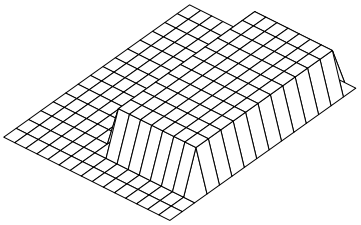
with an elliptic shape The five parameters characterizing any such kernel in the plane are (a, b, θ, x, y), defining the deformation of a origin-centered circle to an arbitrary positioned ellipse in the plane: a - stretch along the x axis, b - stretch along the y axis, θ - rotation angle, x - translation along the x-axis and y - translation along the y-axis (see Fig. 2). In the sequel we show how to find, for any integer k, a compact basis of size k for an arbitrary parametric family of filter kernels. We apply the method to calculate a basis for the origin-centered family of Gaussian kernels (without translation), and show some sample results.



(a)



(b)



(c)

Figure 1: *Approximation of a rotated elongated Gaussian filter on a 16x16 pixel texture map by various classic constant-time schemes:* **(a)** *The Gaussian.* **(b)** *Approximation by trilinear interpolation on a pyramid. Error = 64%.* **(c)** *Approximation by summed-area table. Error = 70%.*

3. Singular Value Decomposition

3.1. Some Matrix Theory

Singular value decomposition is a fundamental tool in operator theory. It is best described in the domain of vector-spaces, where the basic (linear) operator is the matrix. Consider the matrix

$$M = \begin{pmatrix} -1 & 4 & 2 \\ -3 & 12 & 6 \\ 2 & -8 & -4 \end{pmatrix}$$

It is easily verified that $M = (1, 3, -2)^T(-1, 4, 2)$, so the amount of "real information" in M is less than might seem at first glance. Most matrices are not decomposable into the outer product of just one pair of vectors. However, a fundamental theorem of matrix theory ([11], p. 414) guarantees that any $n \times n$ real matrix M may be expressed as the sum of n such outer products:

$$M = \sum_{i=1}^{n} u_i^T \sigma_i v_i \tag{7}$$

where u_i, v_i are real row vectors and σ_i positive scalars. In matrix form:

$$M = U^T S V \tag{8}$$

where U, S and V are $n \times n$ real matrices, S diagonal, and U and V orthonormal. The rows of U are the eigenvectors of MM^T, and the rows of V the eigenvectors of M^TM. The diagonal values of S, denoted $\sigma_1, .., \sigma_n$, are the positive square roots of the eigenvalues of MM^T (and M^TM), called the *singular values* or *spectrum* of M. Moreover, rank(M) = rank(S). The decomposition (8) is called the *singular value decomposition* (SVD) of M. Efficient numerical methods for obtaining the SVD of a matrix are well known ([14], Chap. 2). They involve procedures such as Householder reduction to bidiagonal form and diagonalization by the QR procedure with shifts, and are implemented in most popular numerical packages (e.g. IMSL, MATLAB, LINPACK, etc.). The SVD is defined also for rectangular $n \times m$ matrices. In this case, the dimensions of U are $n \times n$, of S $n \times m$, and of V $m \times m$. The number of singular values is $\min(m, n)$.

There is an interesting connection between the l_2 norm of a matrix (defined analogously to (4)) and its singular values:

$$||M||_2 = ||S||_2 = \left[\sum_{i=1}^{n} \sigma_i^2\right]^{1/2} \tag{9}$$

Assume, without loss of generality, that the σ_i are

$$= \sum_{i=1}^{k} c_i(p_1, .., p_m) < I, B_i >$$

In this case, if the k inner products $< I, B_i >$ are precomputed and stored, all that is required to compute $< I, K >$ for any $K \in \mathcal{K}$ is k additions, multiplications and table lookups, irespective of the size and shape of supp(K). Usually, a large (possibly infinite) number of basis kernels are required to represent all $K \in \mathcal{K}$ exactly. However, if we are willing to make do with a good *approximation*, in many cases a small number of basis kernels suffices. The above analysis reduces the problem of constant-time filtering to the problem of approximating well a parametric family of kernels by a constant number of basis kernels. This method was first proposed by Fournier and Fiume [5], who used polynomial spline surfaces as basis kernels, to span the universal family of *all* kernels. Loosely speaking, in their formulation, the B_i are spline patches, and the $c_i(p_1, .., p_m)$ are effectively the control points.

The quality of an approximation of a parametric family of kernels $\mathcal{K}$ depends on the simultaneous approximation of all $K \in \mathcal{K}$. To quantify this we need to define a *norm*. The simplest norm is the l_2 (sum of squares) norm, denoted by $|| \cdot ||_2$. If f is a function on a domain, with (multi-dimensional) variable $\bar{x}$, its l_2 norm is

$$||f||_2 = \left[\int_{\mathbb{R}^n} f(\bar{x})^2 d\bar{x} \right]^{1/2} \quad . \tag{4}$$

We denote by L^2 the set of functions with finite L_2 norm. Using this norm, it is possible to define the *metric* (distance) between two functions $f_1, f_2 \in L^2$ as $||f_1 - f_2||_2$. If f' is an approximation of f, the *approximation error* may then be defined as

$$e(f', f) = \frac{||f - f'||}{||f||} \quad . \tag{5}$$

To demonstrate the use of this quantity, Fig. 1 shows the approximation error of a filter kernel by the various "classic" constant-time methods. Not surprisingly, they turn out to be rather bad. Let $\mathcal{B} = \{B_i\}_{i=1}^k$ be a set of k basis kernels, and denote by $\text{Approx}_{\mathcal{B}}(\text{K})$ the approximation of the kernel K as a linear combination of these basis kernels. The quality of the approximation of an entire kernel family $\mathcal{K}$ using $\mathcal{B}$ is measured by averaging the error over all $\mathcal{K}$:

$$q(\mathcal{K}, \mathcal{B}) = ||e(\text{Approx}_{\mathcal{B}}(\text{K}), \text{K})|| \quad . \tag{6}$$

Fournier and Fiume use $e()$ of (5) to measure the quality of single kernel approximation. For any given K, their approximation method achieves only a local minimum of this error measure. The advantage of the method to be presented in this paper over theirs is that for any given kernel *family* $\mathcal{K}$ and integer k, we are able to construct a basis of size k which approximates *all* kernels $K \in \mathcal{K}$, and *globally* minimizes the quality metric $q()$ of (6) over all possible bases $\mathcal{B}$. If the kernels used in a given application are confined to a certain family, this is the best that can be hoped for.

It is interesting to note that some non-trivial parametric kernel families may be expressed *exactly* as a linear combination of a constant number of basis kernels. For example, the family of arbitrary rotations of the Gaussian *first derivative* kernel

$$G'(x, y) = \frac{\partial}{\partial x} \exp(-\frac{1}{2}(x^2 + y^2))$$

(governed by the single parameter θ), may be expressed as a linear combination of just *two* basis kernels. Moreover, these kernels are $G'(x, y)$ itself, and a version of it rotated by the angle $\pi/2$, with coefficient functions $\cos\theta$ and $\sin\theta$, i.e. denoting by G'_θ the version of G' rotated by angle θ,

$$G'_\theta(x, y) = \cos\theta G'_0(x, y) + \sin\theta G'_{\pi/2}(x, y) \quad .$$

This observation is the basis of the Canny edge detector [1].

Approximation problems similar to ours have recently been the focus of some attention in the image processing literature. Freeman and Adelson [6] first considered the approximation of the family of *rotated* versions of a given kernel. They call kernels susceptable to this analysis *steerable*, and provide an analytic method to compute the basis kernels. Their method, however, is not optimal in the sense used here (it does not guarantee the minimal basis). Simoncelli et al. [15] later generalized the methods of Freeman and Adelson to a family of kernels with two parameters (orientation and scale), and pointed out the connection between these methods and *wavelet* theory. Independently, Perona [12] achieved the same using singular value decomposition techniques on two-parameter kernel families. His method, extended here to larger parameter families and applied to the problem of constant-time filtering, has the advantage of being optimal.

To illustrate, consider the family of two-dimensional filter kernels which are Gaussians

bined effect of the 3D mapping and the 3D $\rightarrow$ 2D projection is similar to the image warping scenario. Different areas of the texture map are distorted in different manners, requiring different kernel shapes for filtering.

Denote supp(K) = {(u, v) : K(u, v) $\neq$ 0}. A naive computation of the inner product (1) would be to exhaustively sum over the pixels in supp(K). The complexity of this operation is $O(|\text{supp(K)}|)$. Depending on the application, supp(K) may be arbitrarily large and have complex shape, resulting in a time-consuming filtering operation. This is amplified in an animation scenario, where the same image is warped again and again (differently) for each animation frame. It would be very convenient if the inner product in Eq. (1) could be calculated in *constant time* $O(1)$, i.e. independent of the size of supp(K). Algorithms aimed at achieving this have been the subject of much research over the past ten years (see review in [10]). In many applications, preprocessing time is not critical, so can be considered "free". To date, a constant-time algorithm computing (1) exactly for arbitrary kernels with reasonable space requirements is not known. All known efficient filtering algorithms do not compute (1) exactly. Instead, they *approximate* the true value of (1), sacrificing accuracy in favor of efficiency. In general, there is a tradeoff between the preprocessing time, filtering time, space requirements and approximation quality. A popular method for approximate constant-time filtering of a kernel whose support has complex shape is to approximate it by another kernel whose support has a simpler geometric shape, for which exact constant-time filtering is possible. Williams [16] builds a pyramid of the image at various resolution levels, allowing filtering with combinations of certain axis-aligned *square* kernels (box filters) in constant time. This requires space larger than the image size by a factor of 4/3. Crow [2] introduced the summed-area table, enabling constant-time filtering with axis-aligned *rectangular* kernels. The space requirements increase by a factor of 2-4. Ferrari et. al [4] and, independently Heckbert [9], later generalized this by repeated integration of the image to allow filtering with axis-aligned B-spline kernels of order n in $O(n^2)$ time, at the expense of increasing the storage by a factor of $O(n)$. All these approximations are very efficient, but also very crude, as the approximating filter is quite different from the desired one (see Fig. 1). Glassner [7] shows how to modify the summed-area table method to better approximate shapes not oriented with the image axes, but this does not run in constant time. Fournier and Fiume [5] propose a different approach. They approximate the filter kernel by a spline surface which is a linear combination of a small number of "basis" kernels. The entire image is then *prefiltered* with each of the basis kernels (a space-invariant filtering operation) and stored. During space-variant filtering, one value from each of the prefiltered images is used in the computation to form the inner product. Their method has the advantage of working for arbitrary kernels, but is sub-optimal (i.e. does not achieve the best approximation possible for a given number of basis kernels) precisely because of this. We use an approach similar to theirs, restricting ourselves to kernel families small enough to be able to achieve optimal results for, yet rich enough for practical applications. Our main mathematical tool is the *singular value decomposition*. It enables us to tightly control the accuracy of the approximation against the filtering time and space requirements. In a sense (to be made precise later), our tradeoff is the best possible for the kernel family.

2. Approximate Filtering Using Basis Kernels

Assume the filter kernel K may be expressed as a linear combination of k "basis" kernels:

$$K(u,v) = \sum_{i=1}^{k} c_i B_i(u,v) \quad ,$$

then the inner product of I and K is:

$$< I, K > \; = \sum_{u,v} I(u,v)K(u,v) \qquad (2)$$

$$= \sum_{i=1}^{k} c_i < I, B_i > \qquad (3)$$

a linear combination of the inner products of I and the basis kernels B_i. Now assume we are interested in kernels from an (infinite) family $\mathcal{K}$, where each $K \in \mathcal{K}$ is characterized by the values of the parameters $[p_1, p_2, .., p_m]$. Assume also that *all* kernels in $\mathcal{K}$ may be expressed (spanned) as linear combinations of the *same* basis functions $\{B_i\}_{i=1}^{k}$. We allow the coefficients c_i to be functions of the parameter vector $[p_1, .., p_m]$, and call these the *coefficient* functions. Analogously to (3), we may write

$$\forall K \in \mathcal{K} \qquad < I, K[p_1, .., p_m] >$$

Volume xxx, (2000) number yyy pp. 000–000

Constant-Time Filtering by Singular Value Decomposition †

Craig Gotsman

Department of Computer Science, Technion - Israel Institute of Technology, Haifa 32000, Israel
E-mail: gotsman@cs.technion.ac.il

Abstract

We present a technique implementing space-variant filtering of an image, with kernels belonging to a given family, in time independent of the size and shape of the filter kernel support. The essence of our method is efficient approximation of these kernels, belonging to an infinite family governed by a small number of parameters, as a linear combination of a small number k of "basis" kernels. The accuracy of this approximation increases with k, and requires $O(k)$ storage space. Any kernel in the family may be applied to the image in $O(k)$ time using precomputed results of the application of the basis kernels. Performing linear combinations of these values with appropriate coefficients yields the desired result. A tradeoff between algorithm efficiency and approximation quality is obtained by adjusting k.

The basis kernels are computed using singular value decomposition, distinguishing this from previous techniques designed to achieve a similar effect. We illustrate by applying our methods to the family of elliptic Gaussian kernels, a popular choice for filtering warped images.

Keywords: Filtering, Warping, Texture mappping.

1. Introduction

Filtering is a basic ingredient for image rendering in computer graphics, usually applied to overcome problems of *aliasing*, due to the discrete nature of image pixels. In typical image manipulation, such as texture mapping and warping, correct filtering can make all the difference between a high-quality pleasing-to-the-eye image, and a low-quality image.

The operation of a kernel K on an image I, producing one output pixel, is the inner product:

$$< I, K >= \sum_{u,v} I(u,v)K(u,v) \qquad (1)$$

† A preliminary version of this paper was presented at the Fourth Eurographics Workshop on Rendering, June 1993, Paris, France.

The familiar convolution operation is just the operation of translated copies of the same kernel on the image. Convolution is a *space-invariant* filtering operation, and may be computed efficiently (for all possible translations) using Fourier transform methods. The more difficult case, which frequently arises in computer graphics, is *space-variant* filtering, where a *different* kernel is applied at each image pixel position. This need arises during *image warping*, a pure two dimensional operation, where a given (usually rectangular) image is mapped with varying degrees of distortion onto the plane. As the local characteristics of the mapping may be different at different points of the plane, various filter kernels may be required to perform the filtering operation at different image positions. The more general case is *texture mapping*. An image (texture) is mapped onto a three dimensional object, which is then projected onto a viewing plane. The com-