X-Y separable pyramid steerable scalable kernels *

Douglas Shy

California Institute of Technology
M.C. 116-81
Pasadena, CA 91125

email: shy@systems.caltech.edu

Pietro Perona

California Institute of Technology
M.C. 116-81
Pasadena, CA 91125
and Università di Padova, Italy
email: perona@systems.caltech.edu

Abstract

A new method for generating x-y separable steerable scalable approximations of filter kernels is proposed which is based on a generalization of the Singular Value Decomposition (SVD) to 3 dimensions. This "pseudo-SVD" improves upon a previous scheme due to Perona in that it reduces convolution time and storage requirements. An adaptation of the pseudo-SVD is proposed to generate steerable and scalable kernels which are suitable for use with a Laplacian pyramid. The properties of this method are illustrated experimentally in generating steerable and scalable approximations to an early vision edge-detection kernel.

keywords*: steerable, scalable filters, multi-resolution multi-orientation filtering, early vision, multi-way arrays, SVD, pyramid*

1 Introduction

Multi-orientation multi-resolution filtering of images followed by elementary local nonlinearities has recently become a popular computational paradigm for early vision. While earlier work was based on coarse discretizations of the filters along the space, scale and orientation dimensions (for a review see [11]), it has recently been realized that filter outputs need not be discrete since they may be represented in a continuum by interpolating the outputs of appropriately chosen discrete sets of filters. This was first proposed independently by Lenz [8] and Freeman and Adelson [4] in the case of filter rotation, or "steering" a filter; by Perona [9] and by Simoncelli et al. [14] in the case of scale and rotation, and by Perona [10] in the case of general compact deformations in any finite dimension.

*Research funded by NSF Grant IRI 9306155 on "Geometry driven diffusions", NSF grant IRI 9211651, ONR grant NOOO14-93-1-0990 and the California Institute of Technology.

There are two major improvements that may be made on existing schemes for generating steerable, scalable filter decompositions: (1) Compact term-by-term x-y separability (to reduce convolution times) and (2) Pyramid implementation: both issues are addressed in this paper. We present a method for generating compact steerable, filter kernel approximations, in which the basis kernels are x-y separable. This method is also extended to generating steerable and scalable approximations.

Pyramid implementations [1] allow one to save computations by performing lowpass convolutions on coarsely subsampled versions of the image. We propose a method for generating compact x-y separable, steerable, scalable kernel decompositions designed for a Laplacian pyramid.

All of these decompositions are designed from the point of view of minimizing the number of 1D convolutions that need to be performed.

1.1 Background material

A steerable/scalable kernel is one which can be generated at a continuum of orientations/scales by taking linear combinations of, or interpolating between, a finite set of basis kernels. In other words, if $F_{\theta,\sigma}(x,y)$ denotes our filter kernel F rotated by an angle θ, and scaled by a factor σ, then F is steerable and scalable if we can express $F_{\theta,\sigma}$ as the finite sum

$$F_{\theta,\sigma}(x,y) = \sum_{r=1}^{R} h_r(\theta,\sigma) a_r(x,y) \qquad (1)$$

where $h_r(\theta,\sigma)$ may or may not be separable in θ,σ

For most kernels, the number of terms in the sum above is infinite, and thus they are not *exactly* steerable. In all cases, the terms in the sums can be arranged so that truncating the sum after some finite number of terms provides us with the best R-term *approximation* to our original kernel in terms of the functions $a_r(x,y)$. We denote the R-

term steerable/scalable approximation to our kernel F as $F^{[R]}(x, y, \theta, \sigma)$ and refer to R as the *rank*.

A complete steerable, scalable filtering scheme includes three separate stages: decomposition, convolution, and reconstruction. First, a kernel is made steerable and scalable by decomposition into a set of basis kernels. Second, an input image is convolved with this set of basis kernels, producing a set of basis images. Third, the basis images are combined with appropriate factors to produce the convolution at any orientation or scale. The structure of such a scheme suggests the main criteria for judging the optimality of a decomposition: namely, processing time (for convolution and reconstruction), storage costs (for storing the basis images), and the distribution of error in the reconstructed kernel as a function of the deformation parameters, orientation and scale. In this paper, we use processing time as the main measure of optimality.

In order to minimize convolution time, decompositions should be x-y separable [15]. In previous works, this x-y separability was either limited to a small class of kernels [3], or produced *after* the steerability or scalability was generated [11]. The latter method caused the decompositions to contain more terms than was necessary to approximate the filter. We would like to explicitly constrain the decomposition to be x-y separable from the very start.

Perona [10, 11, 12] was the first to propose the use of the singular value decomposition (SVD) to generate simultaneously steerable, scalable, and x-y separable kernel approximations. Indeed, the SVD provides the optimal, non x-y separable, fixed-rank approximation of a kernel, for any combination of deformations. It does this by treating the variables (x, y) as one variable, the deformation parameters as one parameter, thus producing a decomposition which is separable in *two* variables. One would like to find an analog to the SVD which would produce a decomposition separable in three or more variables and would retain the optimality of the SVD. Such a decomposition would approximate our kernel, $F_\theta(x, y)$ in the following way:

$$F_\theta(x, y) \simeq F^{[R]}(x, y, \theta) = \sum_{i=1}^{R} f_i(x) g_i(y) h_i(\theta) \qquad (2)$$

so that this R-term decomposition was the optimal R-term approximation to the original kernel array, in the sense of minimizing the cost function:

$$C(f_i, g_i, h_i) = \|F_\theta(x, y) - \sum_{i=1}^{R} f_i(x) g_i(y) h_i(\theta)\| \qquad (3)$$

for fixed R. Here, we have not explicitly written the "singular values" of such a decomposition, since they may be absorbed into the functions themselves. The properties of this sort of decomposition, which we refer to as the 3D pseudo-SVD, as well as an iterative least squares algorithm for producing it, are covered in section 2.1.

To summarize: we introduce a method for generating steerable, x-y separable kernel approximations: the pseudo-SVD, and we compare it to a previous scheme proposed by Perona. We also apply the 3D pseudo-SVD to generating scalable decompositions, as well as decompositions suitable for use with a Laplacian pyramid.

2 X-Y separable, steerable kernel approximations

For a discussion of the use of the SVD for generating steerable kernel approximations, see Perona [10, 11].

2.1 The 3D pseudo-SVD

The SVD may be used to generate optimal fixed-rank approximations of a matrix, kernel, or linear operator. The crucial property of the SVD is that the first R terms of the SVD of the 2D function/kernel $F(x, y)$ (the set of R triples $\{\lambda_r, f_r(x), g_r(y)\}$) is the "optimal" R-term approximation to $F(x, y)$, in the sense of being the minimizer of the cost function

$$C(\lambda_r, f_r, g_r) = \|F(x, y) - \sum_{r=1}^{R} \lambda_r f_r(x) g_r(y)\| \qquad (4)$$

for a fixed R. The reader should be aware that in the following sections the λ_r are absorbed into the corresponding functions and not (explicitly) written in the sum. Of course, this implies that there are a continuum of equivalent "optimal" solutions which differ only in the norm of $f_r(x)$ and $g_r(y)$.

In generating steerable filter approximations, we have the task of decomposing a 2D filter with 1 parameter: orientation. If we consider this parameter to be a variable, then our problem of simultaneously generating an x-y separable, steerable approximation, can be stated as follows: find a decomposition of the form

$$F^{[R]}(x, y, \theta) = \sum_{r=1}^{R} f_r(x) g_r(y) h_r(\theta) \qquad (5)$$

such that the R-term sum is the optimal R-term approximation to $F(x, y, \theta)$ in the same sense as the 2D SVD: in that it minimizes the corresponding cost function:

$$C(f_r, g_r, h_r) = \|F(x, y, \theta) - \sum_{r=1}^{R} f_r(x) g_r(y) h_r(\theta)\| \qquad (6)$$

Rather than approaching the problem as a series of two, 2D SVD calculations, one would like to solve the problem in toto with a 3D decomposition method which possesses

all of the desirable properties of the two dimensional SVD. Much work has been done in this area [2, 6, 7], yet the current research indicates that there is no known method of decomposition in 3D which will retain *all* of the desirable properties of the 2D SVD. However, certain methods of decomposition have been proposed for 3D functions which attempt to retain one feature of the 2D SVD: that the R-term sum is the optimal R-term approximation of a 3D function, as defined by equation (6). In this section, we will illustrate a method – an iterative least squares algorithm – for producing a decomposition which we refer to as the 3D pseudo-SVD.

There are a few crucial differences between 3D functions/arrays and 2D arrays (matrices). First, there are straightforward ways of computing the rank of a matrix, but there is no known algorithm for computing the rank of a 3D array. Second, the decomposition of most 3D arrays is not "nested" like the SVD of a matrix: i.e. the rank R approximation of a 3D array generally does not contain the rank $R-1$ approximation. This last observation suggests that any algorithm designed to decompose a 3D array should start by assuming a certain rank, and then adjust simultaneously all the different terms to arrive at the decomposition.

2.1.1 The iterative least squares algorithm

Carroll & Chang [2] and Harshman [6] independently introduced a method for decomposing multi-way data arrays based on an iterative least squares approach to minimizing the cost function shown above in equation (6). Carroll & Chang called their method CANDECOMP; Harshman, PARAFAC. Their motivation was to find some underlying structure in a 3D array whose dimensions represented different variables used in gathering the data. In our situation, there is no inherently meaningful structure which we hope to uncover through this type of analysis. Rather, we merely wish to find a decomposition which will minimize convolution time and have certain other desirable features.

The algorithm is outlined below. It is an iterative algorithm, which essentially holds 2 of the 3 sets of functions fixed and optimizes the third. With 2 of the sets held constant, the optimization becomes a straightforward least squares problem. For ease of explanation, we write equation (6) in discrete notation, with superscripts numbering the functions, and subscripts denoting variables:

$$C(f_i^r, g_j^r, h_k^r) = ||F_{ijk} - \sum_{r=1}^{R} f_i^r g_j^r h_k^r|| \qquad (7)$$

Keep in mind the following facts: R, the rank of the decomposition, is the number of terms in the sum given in equation (7) above. The vectors f^r, g^r, h^r have lengths n_x, n_y, n_θ. Before starting the algorithm, one must initialize f, g, h.

1. Define the 3D array F_{ijk} as the original 2D kernel at pixel (i, j) and orientation k.
2. Define the 3D array $A_{jkr} = g_j^r h_k^r$.
3. Create matrices F_{il} A_{lr} by combining the indices (j, k) into one index $l \in (1, \ldots, n_y n_\theta)$.
4. If we consider f_i^r to be a matrix f_{ri} then our cost function, equation (7) becomes:

$$C(f_{ri}) = ||F_{il} - \sum_{r=1}^{R} A_{lr} f_{ri}|| \qquad (8)$$

   which, in matrix notation, becomes:

$$C(f) = ||F^T - Af|| \qquad (9)$$

5. Set $f = \text{pinv}(A)F^T$, where pinv(A) is the pseudo-inverse of A. Then the rows of f are the updated versions of our 1D x kernels.
6. Cycle $f \rightarrow g \rightarrow h \rightarrow f$ and repeat the above steps.

While there is no guarantee that this iterative least squares process will find the *global* minimum of the cost function, it *will* reduce the cost function on every iteration, since the least squares solution is the minimizer of the given cost function.

2.1.2 Experimental results

The algorithm was implemented in `Matlab`, and followed the basic outline given in Section 2.1.1. In generating high rank approximations, the algorithm was initialized with the previous rank solution plus a random triple. As with all minimization routines, the algorithm only guarantees a local minimum, and so to insure a good solution, completely random initial conditions were occasionally used. In practice, the algorithm would almost always converge to within 10% of the final accuracy.

A quick calculation shows the computational complexity of the algorithm, which is mainly due to calculating a pseudo-inverse, which requires the calculation of a reduced SVD. According to [5], the number of flops necessary for the reduced SVD of an $m \times n$ matrix, with $m \geq n$ is $7mn^2 + \frac{11}{3}n^3$. Thus, for our filter array, an $n_x \times n_y \times n_\theta$ array, the total number of flops for one iteration of a rank R approximation is:

$$7R^2(n_x n_y + n_x n_\theta + n_y n_\theta) + 11R^3 \simeq 7R^2(n_x n_y + n_x n_\theta + n_y n_\theta)$$

For our particular array, with $n_x = n_y = 17, n_\theta = 36$ (due to the symmetry of our filter under 180° rotations, we were able to cut the # of angle divisions by half, from 72 to 36) , and for an approximation of rank 16, we have $\simeq$ 2.7 Mflops. Starting from the previous rank solution, the algorithm converged in $\sim$ 50 iterations.

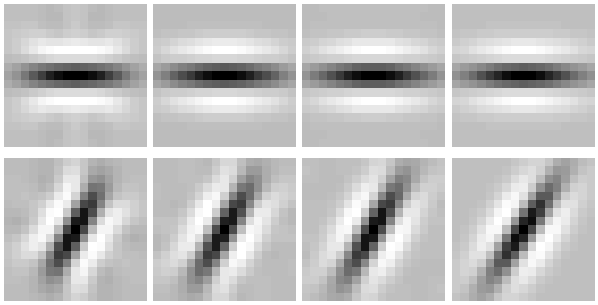

Figure 1: The 3D pseudo-SVD reconstructed filter approximations $F_{ijk}^{[R]}$ at 20%, 10%, 5%, 0% error. These correspond to $R = 7, 10, 13$, and the original filter. The top row shows the filter at 0°; the bottom row, at 60°.

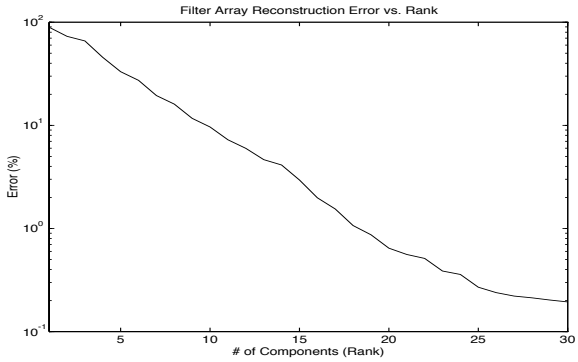

Figure 2: Filter array reconstruction error (in %) for $F_{ijk}^{[R]}$ vs. the rank of the approximation, R, for the first 30 components of the 3D pseudo-SVD

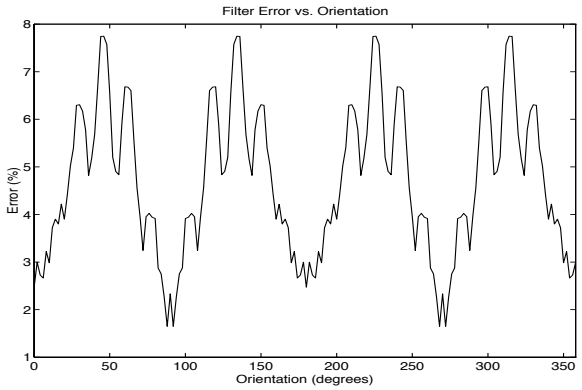

Figure 3: The % error in the reconstructed filter approximation $F_{ijk}^{[13]}$ (a 5% approximation) as a function of the angle of reconstruction.

↓Property Scheme→	Perona–92	3D pseudo-SVD
# x kernels	34	13
# y kernels	34	13
# θ functions	11	13
# 1D convolutions	68	26
# stored images	11	13
off-line kernel decomposition time	$\simeq$ 1 min	$\simeq$ 1 hour
Convolution time	34 sec.	13 sec.
mean error vs. θ	$4.9 \pm .5$	4.8 ± 1.6

Table 1: A comparison of the properties of two decomposition methods. Computation times are referred to a steerable 17x17 kernel, and a 512x512 image on a SUN-SPARC 10.

2.2 Comparison of X-Y-separable, steerable decompositions

In table 1, we summarize the properties of two decomposition methods: the original scheme due to Perona, and the 3D pseudo-SVD. The decompositions were performed on an orientation selective gaussian kernel: a real, 17×17 pixel sampling of a kernel that has been used for brightness boundary detection and texture analysis: the second derivative of a gaussian along the y axis, and a normal gaussian along the x axis. The standard deviation in the x direction was 3 times that of the y direction, which was 1.7 pixels. The set of all angles was discretized in 72 samples. The comparison here is made between kernel approximations of $\simeq 5\%$ accuracy.

The mean error vs. orientation listed in table 1 refers to the mean and distribution of the reconstruction error of the kernel at different orientations, for 5% approximations. Although the 3D pseudo-SVD has a relatively large variation in error vs. orientation, the error is bound below 8%.

3 The 4D pseudo-SVD for steerability and scalability

In section 2.1.1, an iterative least squares algorithm was used to produce the pseudo-SVD of a filter kernel: essentially, a steerable sum of x-y separable kernels. The algorithm, as well as the pseudo-SVD, is by no means limited to 3D problems (2 variables (x, y) and 1 parameter (θ) $\Longrightarrow$ 3 dimensions). The algorithm generalizes easily to an N dimensional array, which in most cases corresponds to 2 variables (x, y) and N–2 parameters. Thus, the pseudo-SVD will provide not only "steerable" decompositions in which the basis filters are x, y separable, but also generally "deformable" decompositions, e.g. "scalable", "stretchable", "shearable", etc., provided that the N–2 deformations involved are continuous and can be parameterized.

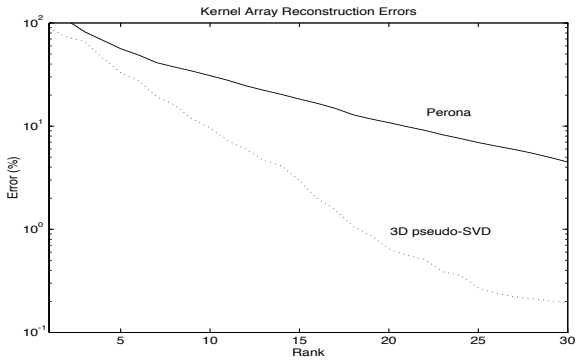

Figure 4: A comparison of the filter reconstruction error vs. # of components for Perona's decomposition (solid line) and the 3D pseudo-SVD (dotted line)

Specifically, one could produce a 4D pseudo-SVD of a kernel which was steerable, scalable and x-y separable using the natural 4D extension of the method outlined above. However, this would unnecessarily constrain the interpolation functions $h(\theta, \sigma)$ to be separable. We can expect to approximate our 2D kernel with fewer terms if we allow the interpolation functions to be non-separable functions.

In practice, this means that our 4D filter array F_{ijkl} becomes F_{ijk} where the index k now parametrizes both orientation and scale, with $k \in (1, 2, \ldots, n_\theta n_\sigma)$. This reduces the problem to finding the 3D pseudo-SVD which was described in detail in section 2.1.1 except that here, the functions h_k^r in equation (7) are now functions of both orientation and scale.

For our experiments we used the edge detection kernel described in section 2.2. The range of scales was chosen to be two octaves, $\sigma \in [.25, 1]$, discretized logarithmically into 8 divisions. In figures 5, 6 and table 2 we show the results of our decomposition for the 3D pSVD, using non-separable interpolation functions.

3.1 Comparison of the 3D $(x, y, (\theta, \sigma))$ pseudo-SVD to a previous scheme

Perona [11] was the first to propose a complete steerable, scalable and x-y separable decomposition of a kernel using the SVD apparatus. The procedure that was followed in his paper was essentially: (1) generate a steerable kernel approximation, (2) generate a scalable approximation for every basis kernel from step (1), and (3) generate an x-y separable approximation for every basis kernel from steps (1) & (2).

As previously noted, this type of "nested" SVD is not optimal. The fact that the x-y separability is produced as the last decomposition creates a much larger number of 1D kernels than does the pseudo-SVD. To provide a reference

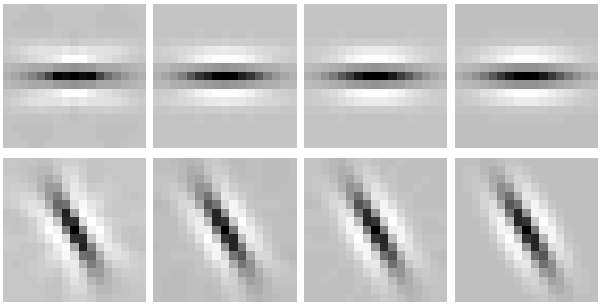

Figure 5: Reconstructed & Original kernels. Columns have constant reconstruction errors 20, 10, 5, 0 % (These correspond to Rank = 18, 28, 38, original kernel). Rows have $\sigma = [.75], \theta = [0^o, 120^o]$.

↓Scheme Property→	# of 1D x kernels	# of stored images
Perona (92)	290	58
Perona (92) + ordered svs	136	44
3D $(x, y, (\theta, \sigma))$ pSVD	39	39

Table 2: A comparison of the compactness of an x-y separable steerable scalable decomposition, for three schemes: a "nested" SVD scheme due to Perona (with no reordering of the singular values), the same scheme with reordering, and the 3D $(x, y, (\theta, \sigma))$ pseudo-SVD. For all schemes, the average filter error was 5%.

point for understanding the compact nature of the pseudo-SVD, table 2 compares the 3D $(x, y, (\theta, \sigma))$ pseudo-SVD to this previous scheme by Perona, for a 5% filter approximation. Also included for comparison is a modified version of the Perona scheme, with a reordering of the product singular values. The pseudo-SVD decreases convolution costs by more than a factor of 7 over the original scheme, and 3 over the same scheme with ordered singular values.

4 Steerable, scalable filters for use with a Laplacian pyramid

The use of a pyramid image representation scheme has become a standard for many areas of image processing. A particular choice of pyramid, the Laplacian pyramid, decomposes an image into essentially band-pass components, and stores it as a set of subsampled images. One can take advantage of the pyramid's structure to reduce convolution time by designing a filtering scheme which performs convolutions at all levels of the pyramid. Thus, some of the convolutions will be done on subsampled images, with smaller size kernels, reducing computation time. In this section we outline such a filtering scheme, and show some preliminary results. We assume that the reader is already

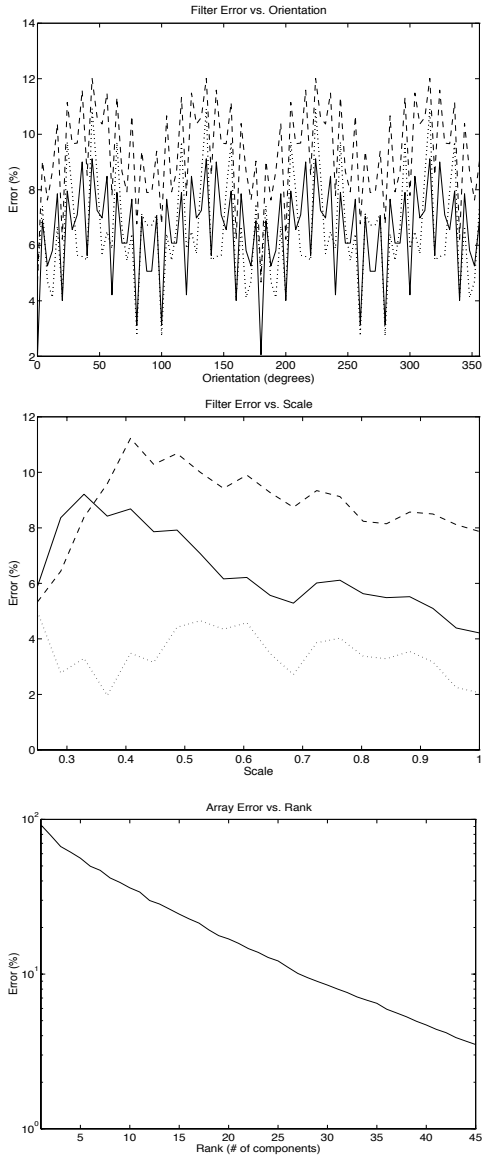

Figure 6: The top figure shows the error in the reconstructed kernel vs. orientation for a 5% array approximation ($F^{[39]}$) at three different scales: solid line ($\sigma = 1$), dashed line ($\sigma = .5$), dotted line ($\sigma = .25$). The middle figure shows the error vs. the scale for three different orientations: solid line ($\theta = 0°$), dashed line ($\theta = 37.5°$), dotted line ($\theta = 60°$) .The bottom figure shows the array reconstruction error (in %) of $F^{[R]}$ vs. the # of components for the first 45 components ($R = 1 \ldots 45$) of the 3D pseudo SVD x-y separable steerable scalable decomposition of the kernel in Fig. 1.

familiar with the pyramid structure: a basic reference is [1]. The following analysis of the Laplacian pyramid generation follows [13].

Let $W(\omega_x)W(\omega_y)$ be the DFT of the 2D x-y separable kernel used to generate our Laplacian pyramid. We will assume that $W(\omega)$ is a sufficiently good approximation to the ideal half-bandwidth, low-pass filter that aliasing terms in the following analysis can be ignored. Denote the procedure of filtering with W then downsampling as *downscaling*, and upsampling then filtering with W as *upscaling*. Our filtering process consists of the following steps: (1) Generate the Laplacian pyramid image levels L_i, $i \in (0, 1 \ldots N-1)$ through the usual process of downscaling, upscaling, and taking differences; (2) Convolve each L_i with a set of kernels $f_{ik}(x)$, $g_{ik}(y)$, $k \in (1, 2 \ldots R_i)$ where R_i is the number of kernels at pyramid level i. If our full-size kernel has a size $2^n + 1$, then f_{ik} has a length $2^{n-i} + 1$. (3) Upscale the convolved levels until they are full size, and combine with coefficients $h_{ik}(\theta, \sigma)$ to obtain the desired orientation and scale of the convolution. For an input image $I(\omega_x, \omega_y)$, we obtain an output which has the form $K^{[R_i]}(\omega_x, \omega_y, \theta, \sigma)I(\omega_x, \omega_y)$ where:

$$K^{[R_i]}(\omega_x, \omega_y, \theta, \sigma) = \sum_{i=0}^{N-1} \sum_{k=1}^{R_i} \left[P_i(\omega_x)P_i(\omega_y) - P_{i+1}(\omega_x)P_{i+1}(\omega_y)\right] \cdot F_{ik}(2^i\omega_x)G_{ik}(2^i\omega_y)H_{ik}(\theta, \sigma) \quad (10)$$

The filter P_i is the generator of the nth Gaussian pyramid level (upscaled to full size), and is defined as:

$$P_i(\omega) = \begin{cases} 1 & i = 0 \\ W^2(\omega) \cdots W^2(2^{i-1}\omega), & i > 0 \end{cases} \quad (11)$$

Rather than writing our cost function in terms of the frequency space functions $F_{ik}(2^i\omega_x)$ and $G_{ik}(2^i\omega_y)$, we can transform equation (10) into position space and write it in terms of the functions $f_{ik}(x), g_{ik}(y), h_{ik}(\theta, \sigma)$ where $f_{ik}(x) = \mathcal{F}^{-1}[F_{ik}(2^i\omega_x)], g_{ik}(y) = \mathcal{F}^{-1}[G_{ik}(2^i\omega_y)]$, and $h_{ik}(\theta, \sigma) = H_{ik}(\theta, \sigma)$. Strictly speaking, $f_{ik}(x)$ is equal to $\mathcal{F}^{-1}[F_{ik}(2^i\omega_x)]$ subsampled by a factor of 2^i. If our initial kernel has a pixel size of $2^n + 1$ on a side, then we would like $f_{ik}(x)$ to be of length $2^{n-i} + 1$. Since we have kernels of different sizes at different levels of the pyramid, in order to use the pseudo-SVD apparatus we will have to solve for each level's kernels separately. In this fashion we isolate one level of the pyramid, treat the kernels at every other level as constants, and then solve a least squares problem for the kernels at that level.

What follows is a brief outline of the algorithm. Basically, the user inputs N = the number of pyramid levels, R_i = the number of kernels at level i, $w(x)$ = the pyramid generating kernel, and the usual kernel parameters $n_x, n_y, n_\theta, n_\sigma$. The algorithm has the iterative least squares structure previously explained, with an added twist: it iterates not only over x, y, and the parameters θ, σ, but also

over the pyramid levels. The parameter functions $h_{ik}(\theta, \sigma)$ are calculated just as in the normal pseudo-SVD: by simultaneously minimizing over all the ranks. The description below applies to solving for the functions $f_{lk}(x)$ and $g_{lk}(y)$: it is explicitly written for calculating $f_{lk}(x)$ at the lth pyramid level. For simplicity, we assume $n_x = n_y$. Denote our original kernel array by $K(x, y, \theta, \sigma)$.

1. Construct the toeplitz matrices T_i which correspond to convolution with $p_i(x)$.

2. Calculate $f^0_{ik} = T_i f_{ik}$ and $f^1_{ik} = T_{i+1} f_{ik}$ for all i. Since the f_{ik} have sizes $2^{n-i} + 1$, it is necessary to select only every 2^ith row of T_i. Repeat for g^0_{ik} and g^1_{ik}.

3. Let $K_l(x, y, \theta, \sigma) = K(x, y, \theta, \sigma) -$

$$\sum_{\substack{i=0 \\ i \neq l}}^{N-1} \sum_{k=1}^{R_i} \left[f^0_{ik}(x) g^0_{ik}(y) - f^1_{ik}(x) g^1_{ik}(y) \right] h_{ik}(\theta, \sigma)$$

   $K_l(x, y, \theta, \sigma)$ is the residual kernel array to be approximated by f_{lk}, g_{lk}.

4. Minimize $C(f_{lk}) =$

$$||K_l(x, y, \theta, \sigma) - \sum_{k=1}^{R_i} (T_l f_{lk} g^0_{lk} h_{lk} - T_{l+1} f_{lk} g^1_{lk} h_{lk})||$$

   Clearly, each of the elements in the sum is linearly dependent upon the elements of f_{lk}. Thus we can write this difference as $||\boldsymbol{K}_l - \boldsymbol{A} \cdot \boldsymbol{f}_l||$ where the R_l kernels in f_{lk} have been stacked into a long vector $\boldsymbol{f}_l$. $\boldsymbol{K}_l$ has been similarly reshaped.

5. Take $\boldsymbol{f}_l = \text{pinv}(\boldsymbol{A}) \cdot \boldsymbol{K}_l$.

4.1 Experimental Results

The one drawback to the algorithm given above is that it is computationally expensive, due to the large pseudo-inverse. In order to limit processing time, the kernel was made steerable and scalable over a small range of scales and orientations: $\sigma \in [.5, 1]$ with 4 discretizations and $\theta \in [0°, 30°]$ with 4 discretizations. Also, we used a variant of the previously described edge-detection kernel which was less elongated in order to make the steerable decomposition more compact. The kernel was defined on a 17×17 pixel grid. We chose a simple pyramid generating kernel: a 5 tap kernel used by Burt [1], $\left[\frac{1}{16} \quad \frac{1}{4} \quad \frac{3}{8} \quad \frac{1}{4} \quad \frac{1}{16} \right]$. The decomposition was generated for a two level pyramid of a 233×233 image, with $R_0 = 8$, $R_1 = 8$, i.e. 8 kernels at each pyramid level. The results are shown in figures 7 and 8.

Figure 7: Top, the original image, Paolina. Below, the two levels of the pyramid.

4.2 Analysis

The problems in generating a pyramid based decomposition can be divided into two categories: first, deducing a cost function which correctly represents the process of performing the convolutions on the pyramid images; and second, minimizing this cost function so as to generate a steerable and scalable decomposition. We have introduced a method, the pseudo-SVD, that attempts a solution to the second problem: its effectiveness is supported by the relatively small error in the right hand graph of figure 9. For comparison, a normal pseudo-SVD decomposition would require 13 kernels to achieve the same array accuracy. Since the level 1 convolutions require $\frac{1}{8}$ the time of those at level 0 we have decreased our convolution time by a factor of $\frac{9}{13}$. The first problem, however, is more complex. By comparing the two graphs in figure 9, one can see that although the kernel is well approximated, the convolution accuracy is less than satisfactory. This implies that our analysis of the filtering scheme was somewhat flawed. Of course, we made the simplifying assumption that the pyramid generating kernel was ideally low-pass, in order to manipulate the cost function into a less complicated form. However, this cannot be the case for finite size pyramid kernels, and ideally our cost function should include the effects of alias-

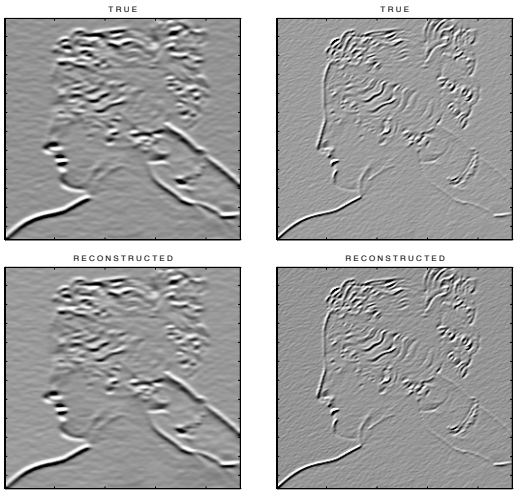

Figure 8: Top row – the actual convolutions at $(\theta, \sigma) = (0°, 1)$ and $(\theta, \sigma) = (30°, .5)$. Bottom row – the reconstructions from the pyramid convolutions.

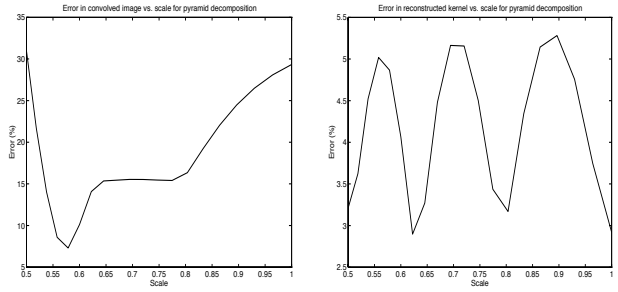

Figure 9: The error in the convolved image (left) and the reconstructed kernel (right) as a function of σ. $(\theta = 15°)$

ing, and should constrain the basis kernels to eliminate this aliasing.

5 Conclusions

In this paper we have presented a method, the pseudo-SVD, for generating compact x-y separable, generally "deformable" filter approximations. Specifically, we applied this method to generating steerable scalable x-y separable approximations of a basic edge-detection kernel used for early vision, and the decomposition was compared to a previous scheme by Perona. This method improves upon this previous scheme in producing a more compact x-y separable decomposition, thus reducing convolution time by a factor of 2 to 7. A modification of the pseudo-SVD was described which generates kernel decompositions suitable for use with a Laplacian pyramid.

Acknowledgements

The authors wish to acknowledge the following people: Mike Burl for his suggestions and work regarding the pyramid scheme; Thomas Leung for debugging the matlab code and helping with implementation; Jan de Leeuw and Steve Breiner for pointing out previous work done on 3D tensor decomposition.

References

[1] P. Burt and E. Adelson. The laplacian pyramid as a compact image code. *IEEE Trans. Commun.*, COM-31:532–540, 1983.

[2] J. D. Carroll and J.-J. Chang. Analysis of individual differences in multidimensional scaling via an n-way generalization of "eckart-young" decomposition. *Psychometrika*, 35(3):283–319, 1970.

[3] W. Freeman and E. Adelson. The design and use of steerable filters. *IEEE Trans. Pattern Anal. Mach. Intell.*, 13:891–906, 1991.

[4] W. Freeman and E. Adelson. Steerable filters. In *Topical Meeting on Image Understanding and Machine Vision*, page vol. 14. Optical Society of America, technical digest series, Cape Cod, June 1989.

[5] G. H. Golub and C. F. V. Loan. *Matrix Computations*. The Johns Hopkins University Press, Baltimore, Maryland, 1983.

[6] R. A. Harshman. Foundations of the parafac procedure: Models and conditions for an "explanatory" multi-modal factor analysis. *UCLA Working Papers in Phonetics*, 16:1–84, 1970.

[7] J. B. Kruskal. Rank, decomposition, and uniqueness for 3-way and n-way arrays. In R. Coppi and S. Bolasco, editors, *Research Methods for Multimode Data Analysis*, pages 7–18. North-Holland, Amsterdam, 1989.

[8] R. Lenz. *Group theoretical methods in image processing*, volume 413 of *Lecture Notes in Computer Science*. Springer Verlag, 1990.

[9] P. Perona. Finite representation of deformable functions. Technical Report 90-034, International Computer Science Institute, 1947 Center st., Berkeley CA 94704, 1990.

[10] P. Perona. Deformable kernels for early vision. *Proc. IEEE Comput. Soc. Conf. Comput. Vision and Pattern Recogn.*, pages 222–227, Maui, June 1991.

[11] P. Perona. Steerable-scalable kernels for edge detection and junction analysis. In *Proc. 2nd Europ. Conf. Comput. Vision, G. Sandini (Ed.), LNCS-Series Vol. 588, Springer-Verlag*, pages 3–18, 1992. Reproduced in Image and Vision Computing, vol 10, pag. 663-672, 1992.

[12] P. Perona. Deformable kernels for early vision. *IEEE Trans. Pattern Anal. Mach. Intell.*, 1994. In press. Previous version appeared as MIT-LIDS TR 2039, 1991.

[13] S. Ranganath. Image filtering using multiresolution representations. *IEEE Trans. Pattern Anal. Mach. Intell.*, 13(5):426–440, 1991.

[14] E. Simoncelli, W. Freeman, E. Adelson, and D. Heeger. Shiftable multi-scale transforms. Technical Report 161, MIT-Media Lab, 1991. Appeared in IEEE Trans. Inform. Theory, vol.38 pgg. 587-607.

[15] S. Treitel and J. Shanks. The design of multistage separable planar filters. *IEEE trans. Geosci. Electron.*, GE-9(1):10–27, 1971.