Overcomplete Steerable Pyramid Filters and Rotation Invariance

H. Greenspan, S. Belongie
R. Goodman and P. Perona
Department of Electrical Engineering
California Institute of Technology
Pasadena CA 91125
(hayit@micro.caltech.edu)

S. Rakshit and C. H. Anderson
Department of Anatomy and Neurobiology
Washington University School of Medicine
St. Louis MO 63110

Abstract

A given (overcomplete) discrete oriented pyramid may be converted into a steerable pyramid by interpolation. We present a technique for deriving the optimal interpolation functions (otherwise called steering coefficients). The proposed scheme is demonstrated on a computationally efficient oriented pyramid, which is a variation on the Burt and Adelson pyramid. We apply the generated steerable pyramid to orientation-invariant texture analysis to demonstrate its excellent rotational isotropy. High classification rates and precise rotation identification are demonstrated.

1 Introduction

Oriented filters play a key role in early visual processes and in image processing. Earlier work by Freeman and Adelson [1] and Perona [2] has shown how the use of steerable filters allows one to use a small set of such filters and still treat all orientations in a uniform way. Freeman and Adelson address the problem of synthesizing exactly steerable filters. Perona addresses the problem of calculating the best steerable approximation to a given impulse response. We approach here a third related problem. It is sometimes desirable to use a particular set of oriented filters, due to certain desired filter characteristics, computational complexity, existing hardware implementations or other constraints. The question arises: what is the best way to interpolate a given set of filters? In this paper we present a technique for deriving the optimal set of interpolation functions (steering coefficients) for a given overcomplete discrete representation, thus generating a steerable representation. We illustrate our technique using as a sample case an oriented Laplacian pyramid, which allows for a computationally efficient Gabor-like filtering scheme.

The oriented Laplacian pyramid filtering scheme is a variation on the Burt and Adelson pyramid [3,4]. It was presented in the context of a texture-recognition system in [5]. Some of its interesting characteristics are its computational efficiency and compactness, which lead to minimal hardware requirements. We describe it in section 2.

We next show how the oriented pyramid, which has $8/3$ redundancy (this is a more compact representation than has previously been used in the literature [1,3]), can be transformed into a steerable one. In sections 3 and 4 we present the procedure to calculate the interpolation functions which give us a steerable representation.

We conclude this paper by addressing the issue of rotation invariant recognition via the extracted steerable representation. We present a scheme for generating rotationally-invariant feature-vectors for an input image, together with extracting the actual rotation information. High classification rates and precise rotation estimation results are presented for both synthetic and natural textured images, which demonstrate the usefulness and precision of the steerable pyramid in the difficult real-world task of rotation invariant texture recognition.

2 The Oriented Laplacian Pyramid

There is both biological and computational evidence supporting the use of a bank of orientation-selective bandpass filters, such as the Gabor filters, for the initial feature extraction phase of many image-processing tasks. These tasks include edge-detection, motion-flow analysis and texture recognition [5,6,7,8]. Orientation and frequency responses are extracted from local areas of the input image and the statistics of the coefficients characterizing the local area form a representative feature vector. In the application domains listed above, the extracted feature vectors are used as an intermediate step towards orientation analysis, or other higher-level analysis. The constraints are therefore computational efficiency and memory requirements (especially important for real-world applications), as opposed to achieving a complete self-inverting representation which is important for coding and reconstruction purposes. It is this distinction which motivates us into using the oriented

Laplacian pyramid described below, which is both computationally efficient and compact.

The pyramid filtering scheme

In a pyramid representation the original image is decomposed into sets of lowpass and bandpass components via Gaussian and Laplacian pyramids, respectively [3]. The Gaussian pyramid consists of lowpass filtered (LPF) versions of the input image, with each stage of the pyramid computed by lowpass filtering and subsampling of the previous stage. The Laplacian pyramid consists of bandpass filtered (BPF) versions of the input image, with each stage of the pyramid constructed by the subtraction of two corresponding adjacent levels of the Gaussian pyramid. We use the Filter-Subtract-Decimate (FSD) Laplacian pyramid [4], which is a variation on the Burt Laplacian pyramid [3]. In the following we refer to the input image as G_0, the LPF versions are labeled G_1 thru G_N with decreasing resolutions and the corresponding BPF versions are labeled L_0 thru L_N respectively.

$$G^0_{n+1} = W * G_n \; ; L_n = G_n - G^0_{n+1}$$
$$G_{n+1} = \text{Subsampled } G^0_{n+1} \qquad (1)$$

The LPF, W, is Gaussian in shape, normalized to have its coefficients sum to 1. The values used in this work for W, which is a 5-sample separable filter, are (1/16, 1/4, 3/8, 1/4, 1/16).

In order to extract the orientationally tuned band-pass filtering responses, the oriented pyramid is formed next. The oriented pyramid is the result of modulating each level of the Laplacian pyramid with a set of oriented sine waves, followed by another LPF operation using a separable filter, and corresponding subsampling, as defined in (2):

$$O_{n\alpha} = LPF[e^{(i\vec{k_\alpha}\cdot\vec{r})} L_n[x,y]] \qquad (2)$$

where $O_{n\alpha}$ is the oriented image at scale n and orientation α, $\vec{r} = x\vec{\imath} + y\vec{\jmath}$ (x and y are the spatial coordinates of the Laplacian image), $\vec{k}_\alpha = (\pi/2)[\cos\theta_\alpha\vec{\imath} + \sin\theta_\alpha\vec{\jmath}]$ and $\theta_\alpha = (\pi/N)(\alpha - 1), \; (\alpha = 1..N)$.

In this work we use 4 oriented components ($N = 4$). From (2), each level (n) of the pyramid is thus modulated by the following complex sinusoids:

$$m_1(x,y) = e^{i(\pi/2)x} \; ; m_2(x,y) = e^{i(\pi\sqrt{2}/4)(x+y)}$$
$$m_3(x,y) = e^{i(\pi/2)y} \; ; m_4(x,y) = e^{i(\pi\sqrt{2}/4)(y-x)} \qquad (3)$$

These four modulators differ only in their orientations, which are 0°, 45°, 90° or 135° for m_1 through m_4, respectively. The origin of x and y is taken to be the center of the image being modulated. Note that the modulating

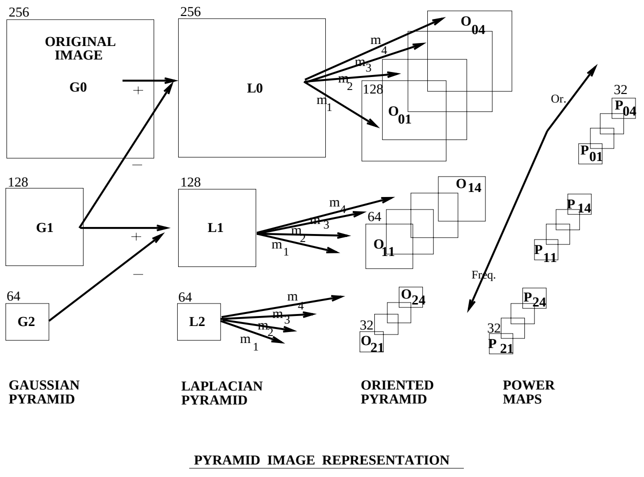


Figure 1: Block diagram of the oriented pyramid generation. The power maps represent the local statistics of the oriented pyramid's coefficients (section 5).

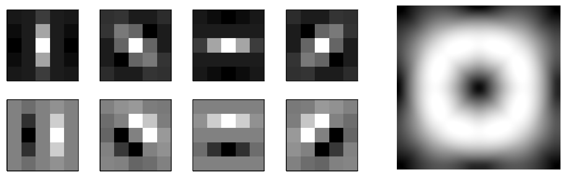


Figure 2: Left: A set of oriented pyramid filters, $O_{n\alpha}$. Real and imaginary components are presented, top and bottom, respectively, for $n = 0$ and $\alpha = 1..4$. Right: Power spectra characteristics for the chosen filter set (+ conjugate counterparts).

frequency remains constant for each level of the pyramid. After modulation, the Laplacian images are lowpass filtered and subsampled. At this point, the Laplacian images have effectively been filtered by a set of log-Gabor filters:

$$\psi_k(x,y) = \frac{1}{2\pi} e^{-(x^2+y^2)/2} \times m_k(x,y) \; ; k = 1..4 \quad (4)$$

Fig. 1 shows a block-diagram of the orientation-pyramid generation. A set of oriented-pyramid filters are displayed in Fig. 2.

Pyramid characteristics

We briefly present some characteristics of the pyramid. The interested reader is referred to [1,3,4,10] for more elaborate details.

- In the FSD pyramid (1), subtraction occurs before the decimation step. This ensures that aliasing does not get incorporated into the mid-band regions of the bandpass images, L_n, and gives better bandpass characteristics overall.
- The FSD pyramid allows for a simple pipeline architecture.

- The filtering operation in (2) is not the standard one found in the literature. Usually, the original image is filtered with a set of oriented sinewave modulated Gaussian filters. In (2) the oriented filters are applied to the bandpass image, L_n. This ensures good low-frequency (dc) rejection. In addition, a reversal in the ordering of the filtering operations is performed (the image is first modulated by a sinewave and then LPFed, rather than modulating the LPF prior to convolving with the image). This reversal gives us separable filters, and therefore it allows for a computationally efficient filtering scheme.

We next investigate the redundancy of the generated pyramid. The redundancy in the nonoriented Laplacian pyramid representation is 4/3. In the pyramid scheme defined above, we use four complex oriented filters to create *eight* oriented bandpass components from each nonoriented Laplacian level. The eight include the real and imaginary response maps from each complex oriented filter modulation. Since this involves lowpass filtering after the modulation, it is possible to subsample these oriented bands by a factor of 2 in each dimension. From each band of size $M \times M$, we thus hold 8 bands of size $M^2/4$. The total number of pixels at each level therefore increases from M^2 to $(M^2 \times 8)/4 = M^2 \times 2$, leading to an increase of redundancy by a factor of two. Overall, the redundancy of our oriented pyramid is $4/3 \times 2 = 8/3$. This pyramid is more compact than other oriented pyramids described in the literature which usually exhibit $16/3$ redundancy [1-3].

In the following sections we investigate into the extracted pyramid kernels. We show that we span the orientation space, we extract the interpolation coefficients that allow steerability in orientation, and finally, we utilize these properties for rotation invariant recognition.

3 Spanning the Orientation Space

In this section we show that the oriented pyramid as defined in the preceding section, with the selected oriented kernels of 45° bandwidth, spans the orientation space.

Following the work of Perona [2], we make use of the singular-value decomposition (SVD) to investigate the independence of the set of oriented pyramid kernels (as in equation 2). This procedure consists of the following steps:

- Generate 360 oriented pyramid kernels (at a single scale) via equation 2 with $N = 360$.
- Concatenate each of these 360 kernel matrices into column-vectors, and combine these column vectors to form a large matrix, **A**
- Perform the SVD by finding the matrices **U**, **V**, and diagonal matrix $\mathbf{\Sigma}$ such that

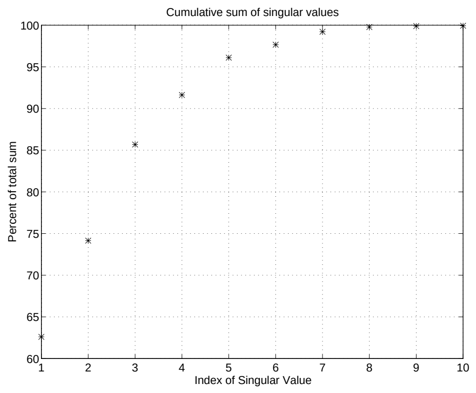


Figure 3: SVD decomposition for the oriented pyramid kernels. The first seven singular values contain approx. 99.5% of the sum of all the singular values.

$$\mathbf{A} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T \tag{5}$$

The diagonal matrix $\mathbf{\Sigma}$ contains the square roots of the positive eigenvalues of $\mathbf{A}^T\mathbf{A}$. The number of nonzero eigenvalues in $\mathbf{\Sigma}$ is equal to the number of linearly independent column vectors in **A**. Upon inspecting the results of the SVD, the first seven singular values, $\sigma_1..\sigma_7$, in $\mathbf{\Sigma}$ contain approximately 99.5% of the sum of all the singular values $(\sum \sigma_i)$. This is shown in Fig. 3. The above result indicates that a set of eight filters, i.e., an orientation bandwidth of 45°, is sufficient to span the 360° of orientation space with more than 99% accuracy. The four filters, O_{n1} through O_{n4}, and their conjugate counterparts, which we hereon term O_{n5} through O_{n8}, satisfy this requirement. The chosen set of 4 filters are shown in Fig. 2 left. The filters' combined power spectra (above 4 together with their conjugate counterparts) covers uniformly the 360° orientation space, as shown in Fig. 2 right.

4 Interpolating in Orientation Space

Given the set of oriented pyramid filters, O_{n1} through O_{n8}, we next define the interpolation functions (or steering coefficients) which allow us to use the finite set of eight filters (per scale) to synthesize oriented filters across the entire orientation space. Note that in this section we assume the input image to the pyramid to be a delta function, $G_0(x, y) = \delta(x, y)$. We wish to use a finite set of oriented filters to calculate the output of filters at any orientation in a continuum. Let $\beta_{n,k}(\theta), \quad k = 1..8$, represent the interpolation coefficients in orientation space. We wish to calculate the filter output for any given angle θ, which we define as $\hat{F}_\theta(x)$, via a linear interpolation scheme as follows:

$$\hat{F}_{n,\theta}(x) = \sum_{k=1}^{8} \beta_{n,k}(\theta) O_{n,k}(x) \tag{6}$$

For clarity purposes we hereon avoid using the scale notation, with the understanding that the following derivation is performed at each scale, n, independently.

Our goal is to minimize the error between the filter output, $F_\theta(x)$, and the interpolated output, $\hat{F}_\theta(x)$, (in space for a particular orientation θ):

$$min_{\beta_{n,k}} \; ||F_\theta(x) - \hat{F}_\theta(x)||^2_{\Re^2} \tag{7}$$

We have

$$||F_\theta(x) - \hat{F}_\theta(x)||^2 = ||F_\theta(x)||^2 + ||\hat{F}_\theta(x)||^2 - 2\langle F_\theta(x), \hat{F}_\theta(x)\rangle \tag{8}$$

where

$$\begin{aligned} ||\hat{F}_\theta(x)||^2 &= \sum_{h,k} \langle \beta_k(\theta) O_k(x), \beta_h(\theta) O_h(x)\rangle \\ &= \sum_{h,k} \beta_h \beta_k \langle O_h, O_k\rangle_{\Re^2} \end{aligned} \tag{9}$$

and

$$\begin{aligned} \langle F_\theta(x), \hat{F}_\theta(x)\rangle &= \sum_h \beta_h \langle O_h(x), F_\theta(x)\rangle \\ &= \sum_h \beta_h \langle F_\theta, O_h\rangle_{\Re^2} \\ &= \sum_h \beta_h \cdot \Gamma_h(\theta) \end{aligned} \tag{10}$$

with $\Gamma_h(\theta) = \langle F_\theta, O_h\rangle_{\Re^2}$.

Using (8-10), we need to minimize the following expression with respect to β:

$$\underbrace{\sum_{h,k} \beta_h \cdot \beta_k \langle O_h, O_k\rangle_{\Re^2}}_{a} - \underbrace{2\sum_h \beta_h \cdot \Gamma_h(\theta)}_{b} \tag{11}$$

The derivative with respect to β_z of the left term (a) gives: $2\sum_k \beta_k \langle O_z, O_k\rangle$. The derivative of the right term (b) gives: $-2\Gamma_z(\theta)$.

Equating the sum of the above two terms to zero leads to the following equation for the β's:

$$\sum_k \langle O_z, O_k\rangle_{\Re^2} \beta_k(\theta) = \Gamma_z(\theta) \qquad z = 1..N, k = 1..N \tag{12}$$

In matrix form we have:

$$\mathbf{O}\beta = \mathbf{\Gamma}, \tag{13}$$

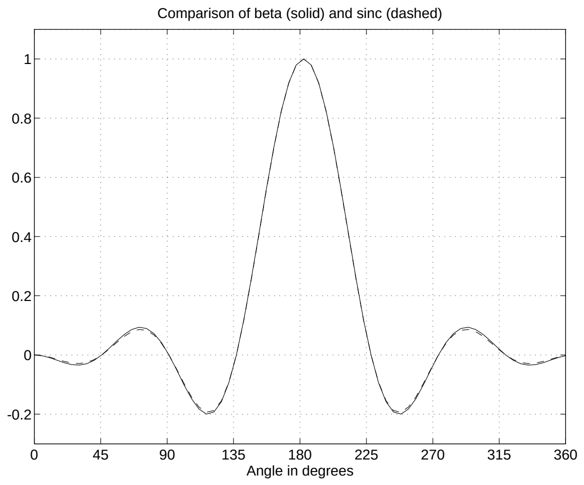


Figure 4: One characteristic interpolation function (solid), as compared with the Sinc function (dashed).

with $\mathbf{O} = \langle O_h, O_k\rangle_{\Re^2}$, $\beta =$ a column of the β_k's, $k = 1..N$, and $\mathbf{\Gamma} =$ a column of the Γ_z's, $z = 1..N$.

In the orthonormal case, $\mathbf{O}$ is a diagonal (identity) matrix since $\langle O_h, O_k\rangle = \delta_{hk}$.

In that case from (12) we get:

$$\beta_h(\theta) = \Gamma_h(\theta) = \langle F_\theta, O_h\rangle_{\Re^2} \tag{14}$$

For the nonorthonormal case, the solution requires more computation – one method would be Gauss elimination method, the other would be to decompose $\mathbf{O}$ by SVD to $\mathbf{U\Sigma V}^T$ and use this to calculate $\mathbf{O}^{-1}$. Here, $\mathbf{UU}^T = \mathbf{I}$ and $\mathbf{VV}^T = \mathbf{I}$. The inverse matrix, $\mathbf{O}^{-1}$, can be found as:

$$\mathbf{O}^{-1} = \mathbf{V}[diag(1/(\lambda_j)]\mathbf{U}^T \tag{15}$$

The solution for β can now be extracted as:

$$\beta = \mathbf{V}[diag(1/(\lambda_j)]\mathbf{U}^T \cdot \mathbf{\Gamma}, \tag{16}$$

where in the case of a zero eigenvalue, $\lambda_j = 0$, the corresponding $1/\lambda_j$ in $\mathbf{\Sigma}^{-1}$ gets replaced by a zero. The above scenario takes care of all possible $\mathbf{O}$ matrices, even if the matrix is not full rank. Overall, if $\mathbf{\Gamma}$ is in the range of $\mathbf{O}$ then the extracted β functions are exact. If $\mathbf{\Gamma}$ is *not* in the range then the β functions are the closest we can find in least-squares sense; i.e., minimizing $|\mathbf{O} \cdot \beta - \mathbf{\Gamma}|$.

Using the eight 45° bandwidth oriented filters, O_1 through O_8, we extract the eight steering coefficients (β_k for $k = 1..8$), as outlined above. A plot of $\beta_5(\theta)$ over the range $0° - 360°$ is shown in Fig. 4. It is very similar to a sinc function. The curves for each $\beta_k(\theta) k = 1..8$ are cyclic shifts of one another at 45° increments.

With the interpolation functions in hand, we can now go back and calculate the oriented filters, $\hat{F}_\theta$, from the finite

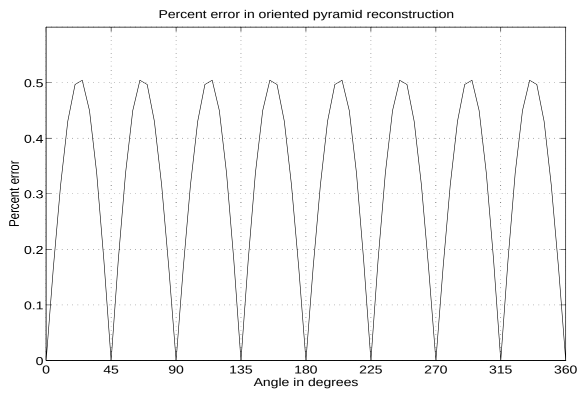


Figure 5: Percent error in the reconstruction of oriented filters across the continuous orientation space, $\theta = 0 - 360$ degrees, from the finite filter set, $O_k, k = 1..8$.

set of oriented filters, O_k, as:

$$\hat{F}_\theta = \sum_k \beta_k(\theta) O_k \tag{17}$$

across the continuous orientation space, $\theta = 0^\circ - 360^\circ$. Fig. 5 shows the percent error, $E(\theta)$, in the reconstruction of the oriented filters, for $\theta = 0^\circ - 360^\circ$ with steps of 5°. Here the interpolation error E is defined as:

$$E(\theta) = \frac{\|F_\theta - \hat{F}_\theta\|}{\|F_\theta\|} \times 100. \tag{18}$$

Note that the peak error is less than 1%. This is in agreement with the SVD bound found in section 2. We have thus completed the proof of the pyramid steerability.

An alternative scheme to the interpolation functions derivation of above which does not involve matrix inversions, makes use of the Gram-Schmidt orthogonalization process. This scheme is outlined in [10].

5 Application of the Steerable Pyramid Kernels for Rotation-Invariant Texture Recognition

We conclude this paper by demonstrating the application of the steerable pyramid kernels to rotation-invariant texture recognition. Here we are interested in learning a set of textured inputs, following which we would like to recognize new test inputs as belonging to one of the prelearned classes, even if the new input is rotated relative to the original input. Furthermore, we wish to state the orientation of the test input relative to the original one. An example of 10 textures, 8 taken from the Brodatz texture database [9], and 2 (cardboard, denim) acquired by us, is presented in Fig. 6.

Feature vectors are extracted from the oriented pyramid via the following procedure: *Power maps* are first extracted from the oriented pyramid. The power of each filtered map can be defined as:

$$P_{n\alpha} = |O_{n\alpha}|, n = 0, 1, 2 \;\; \alpha = 1, 2, 3, 4. \tag{19}$$

The power maps form a pyramid of the local statistics of the oriented pyramid's coefficients, which characterize the image local-area response to the different orientations and frequencies. Levels 0 and 1 of the power-pyramid are low-passed and subsampled to be the size of the smallest level of the pyramid (see Fig. 1). Each pixel in the resultant power maps thus represents an 8×8 window in the original image. 15 dimensional feature-vectors are formed from the extracted power maps. These vectors consist of the 4 oriented components per scale together with a non-oriented component extracted from the Laplacian pyramid. For additional details on the feature extraction stage see [5].

For a given input texture we define a characteristic curve (per scale) across orientation space as the texture's response curve to any oriented filter in the 360° space. The four oriented components (per scale) and their conjugate counterparts, sample the texture's characteristic curve at eight points. These samples cycle along the continuous curve as the texture is rotated.

We have so far shown the steerability of the oriented filters (17). The exact interpolation equation for the filter *powers* is complex and will not be derived here. Given the fact that the energy is lowpass in orientation we make the approximation that the filter output powers can be interpolated with the (sinc-like) β functions (this is confirmed by empirical observations):

$$\hat{P}_\theta \approx \sum_k \beta_k(\theta) P_k. \tag{20}$$

Here, $P_k, k = 1..4$, are the 4 oriented power components, $P_{n\alpha}, \alpha = 1..4$, and $P_k, k = 5..8$ is a duplicate set representing the power components of the conjugate counterparts. $\hat{P}_\theta$ represents the estimated power map for the texture rotated at θ.

We test the estimation accuracy on a few texture examples. Fig. 7 presents the estimation error, $E(\theta)$, across orientation space (steps of 5°), for a set of 5 textures. Here:

$$E(\theta) = \frac{\|P_\theta - \hat{P}_\theta\|}{\|P_\theta\|} \times 100. \tag{21}$$

with P_θ representing the actual power map extracted from the input texture rotated to θ, and $\hat{P}_\theta$ representing the estimated response based on the original, nonrotated power maps. The error is less than 3%. These results demonstrate that the finite set of oriented filters which we chose

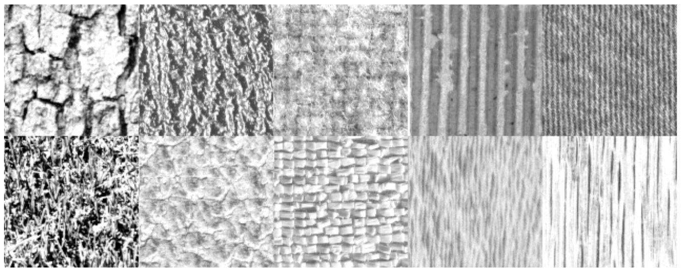


Figure 6: 10 texture database. Top row (left to right): bark, calf, cloth, cardboard, denim. Bottom row (left to right): grass, pig, raffia, water, wood.

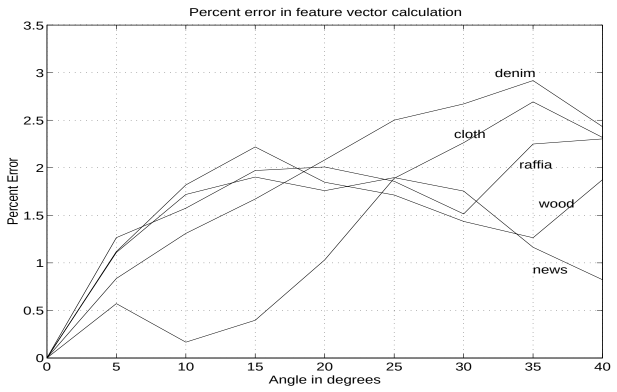


Figure 7: Percent error in the calculation of characteristic curves, 5 texture case.

for our representation gives us a steerable pyramid and indicates the validity of the interpolation functions in a real application. Furthermore, the results confirm the validity of (20).

Shifting to a DFT representation

We note that four samples allow us to reconstruct the characteristic curve for each texture in orientation space. We can therefore shift to any other four-dimensional representation. For recognition purposes it is beneficial to use a shift-invariant representation, such as the Discrete Fourier Transform (DFT). Companion feature-vectors are formed, $\hat{f}$, which contain the 4-point DFT of the oriented components for each level, as in (22), in addition to the three unaltered non-oriented components from the original feature-vector. In our case, f_q represents the four components P_{nq}.

$$\hat{f}_{k+1} = \sum_{q=0}^{3} f_{q+1} e^{-i\pi qk/2} \text{ for } k = 0, 1, 2, 3 \qquad (22)$$

The DFT can be used to create companion feature-vectors that are invariant in magnitude and reveal through their phase the rotation of the input texture.

To begin investigating the prospect of an invariant feature waveform, the feature-vectors for an ideal sinusoidal grating texture at orientations from 0° to 45° with steps of 5° were set aside, for a total of 10 feature-vectors. Then a companion set of 10 feature-vectors was formed. Fig. 8 top shows magnitudes of the 10 DFT's for the ideal sinusoidal grating and denim test textures. Fig. 8 bottom shows the phase of the DFT's for the sinusoidal grating. The magnitudes overlap onto a single characteristic curve. With regard to the phase, either $\hat{f}_2$ or $\hat{f}_4$ can be inspected to determine the amount of rotation on the input.

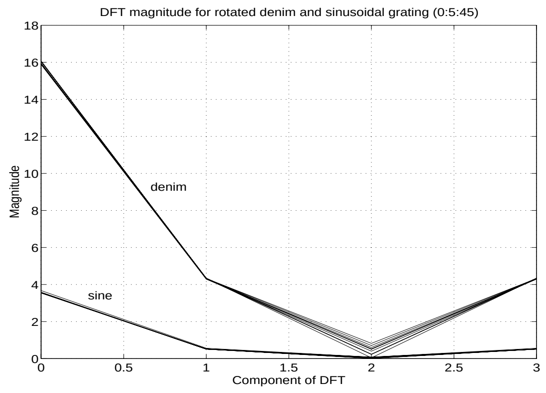
Rotation invariant texture recognition results

Finally, we present results of applying the above analysis to a 10-texture recognition task (see Fig. 6). The test consists of presenting different 128×128 images from the input set, with each image arbitrarily rotated at one of 5 angles: (0, 10, 20, 30, 40) degrees. In the recognition process feature-vectors are extracted and each component is averaged over the entire image, to produce one representative feature-vector per input. The extracted feature-vector, f, is next used to generate the companion feature-vector, $\hat{f}$, via the DFT transformation of the previous section.

For each of the 10 textures we investigate the *magnitude* deviation of the representative feature-vector, $\hat{f}$, as the input texture is rotated. We compare the standard deviation within each class, c_i (in the 15 dimensional space), to the average (and minimum) distance between the mean of class c_i and the means of all other texture classes c_j, $j \neq i$; i.e. the average (/min) interclass distance. This is shown in the following table:

texture	innerclass std.	interclass avg.	interclass min.
bark	0.99	10.89	5.92
calf	2.62	12.56	6.37
cloth	3.84	10.29	0.74
cardboard	2.76	12.67	6.45
denim	1.27	14.84	6.37
grass	1.33	18.17	7.86
pig	0.87	10.32	0.74
raffia	0.97	9.61	5.92
water	0.65	15.54	8.11
wood	0.96	10.59	6.08

In the above results we observe more than a factor of 10 difference between the innerclass and average interclass distances, for most textures. Looking at the minimum interclass distance for each texture, we again see that it is much larger than the innerclass standard deviation, except for the cloth and pig texture pair, for which the representative feature-vector means are very close. This difficulty is inherent in the similarity of the textures, as can be seen in Fig. 6. The small innerclass standard-deviation strongly



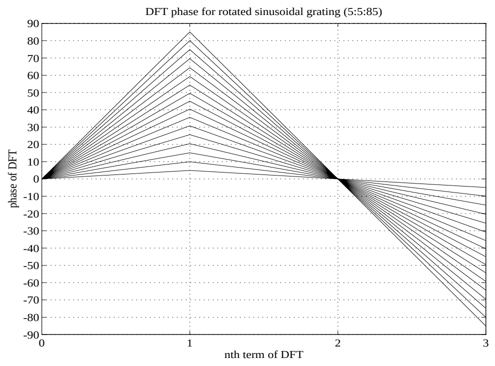


Figure 8: Top: DFT magnitudes for 10 rotated sinusoidal and denim test textures. Bottom: DFT phase for 10 rotated ideal sinusoidal-grating textures.

indicates the consistency of the DFT magnitude representation; i.e., the invariance of the response with the rotation of the input textures. To make this claim stronger, we use the well known K-nearest-neighbor classification scheme on the set of 10 textures above. In the training stage, we use 16 128×128 examples per texture class, with no rotation. The test set consists of a new set of textures, which are rotated arbitrarily in one of the 5 angles: $(0, 10, 20, 30, 40)$ degrees. In this 10 texture recognition case we get 100% correct classification.

Once the identity is found we utilize the *phase* information from the DFT representation, to estimate the orientation of the test input, relative to the original texture from the training set. Here we are interested in the error, in degrees, between the true rotation angle and the estimated rotation angle. The average rotation-angle estimation error for the 10 textures is $0.84°$.

Both the perfect class identification and the high-resolution orientation estimation, as presented above, are very encouraging results in the difficult domain of rotation invariant natural texture identification.

6 Summary and Conclusions

In this paper we have presented an optimal technique for deriving the set of interpolation functions (or steering coefficients) which enable us to convert a given overcomplete oriented filter set into a steerable representation. As a sample case we have chosen to work on a computationally efficient oriented Laplacian pyramid. We described the characteristics of the $8/3$ redundant pyramid and have shown that the pyramid is steerable by defining a set of eight $45°$ bandwidth oriented kernels and deriving the corresponding steering coefficients. Properties of the kernels and interpolation functions have been investigated. Finally, we demonstrated highly encouraging results in applying the pyramid for rotation-invariant recognition.

A similar framework to the one presented here can be applied to scale invariance. Future work involves extending the work to scale and rotation invariant texture recognition on large databases [11].

Acknowledgements

This work was supported in part by Pacific Bell, and in part by ARPA and ONR under grant no. N00014-92-J-1860. H. Greenspan was supported in part by an Intel fellowship. P. Perona is supported by NSF Research Initiation grant IRI 9211651. S. Rakshit and C. H. Anderson are supported in part by ONR Grant N00014-89-J-1192.

References

[1] W. T. Freeman and E. H. Adelson, “The Design and Use of Steerable Filters,” *IEEE Trans. on Pattern Analysis and Machine Intelligence,* Vol. 13, No. 9, 891-906, Sept. 1991.

[2] P. Perona, “Deformable Kernels for Early Vision,” *IEEE Conference on Computer Vision and Pattern Recognition,* pages 222-227, June 1991.

[3] P. J. Burt and E. A. Adelson, “The Laplacian Pyramid as a Compact Image Code,” *IEEE Transactions on Communications,* Vol. 31, 532-540, 1983.

[4] C. H. Anderson, “A Filter-Subtract-Decimate Hierarchical Pyramid Signal Analyzing and Synthesizing Technique,” *United States Patent* 4,718,104, 1987.

[5] H. Greenspan, R. Goodman, R. Chellappa and C. Anderson, “Learning Texture Discrimination Rules in a Multiresolution System,” to appear in the special issue on “Learning in Computer Vision” of the *IEEE Transactions on Pattern Analysis and Machine Intelligence,* July 1994.

[6] H. E. Knutsson and G. H. Granlund, “Texture analysis using two-dimensional quadrature filters,” In *IEEE Workshop Comp. Arch. Patt. Anal. Im. Dat. Base Man.*, Pasadena, CA, 1983.

[7] M. R. Turner, “texture discrimination by gabor functions,” *Biol. Cybern,* 55:71-82, 1986.

[8] A. C. Bovik, M. Clark and W. S. Geisler, “Multichannel Texture Analysis Using Localized Spatial Filters,” *IEEE Transactions on Pattern Analysis and Machine Intelligence,* Vol. 12, 55-73, 1990.

[9] P. Brodatz, *Textures,* New York:Dover, 1966.

[10] H. Greenspan, *Ph.D. Thesis* in preparation.

[11] H. Greenspan, S. Belongie, P. Perona and R. Goodman, “Rotation Invariant Texture Recognition Using a Steerable Pyramid,” submitted to the *12th International Conference on Pattern Recognition*, Oct. 1994.