# Chapter 3

# Lecture 3: Filters, Quadrature, and Pyramids

Weds, Feb. 9, 2010 MIT EECS course 6.869, Bill Freeman and Antonio Torralba
Slide numbers refer to the file "03ImagePyramids2011"

**slides 105, 106 of lecture 2, slide 3 of lecture 3** a good start is to analyze an image with a sinusoidal analysis region. so let's multiply a fourier basis function by a spatially localizing gaussian window. The result is called a Gabor function. It's a complex valued function, but we can look at the real or imaginary parts to examine cosine or sine phase ripples.

**slide 4** Note that these Gabor filters are very similar to the shape of cortical receptive fields found in the mammalian visual system. This provides a hint that we're on the right track with these. What can we do with them? In isolation, they are useful for analyzing line or edge phase structures in images. but they have many other benefits when we combine them together in quadrature pairs.

**slide 5** Here we've taken sine and cosine phase Gabor filters, at two different spatial scales, and applied them vertically to the same image. In those 4 bandpass image outputs, you can see differences that reveal what the Gabor filters are doing. First of all, all 4 outputs emphasize the vertical structures and remove horizontal details, as you'd expect. The lower-frequency Gabors of the top row pick out larger scale structures (windows, people and their limbs), than the higher-frequency Gabors of the bottom row (which pick out edges and narrow lines). If you look carefully, you can spot where the even-phase filter favors even-phase structures, while the odd-phase Gabor naturally favors edges.

**slides 7-14**
quadrature pairs measure local oriented energy. These can be used to identify contours, independently of the phase of the contour.

**slides 7**: This slide shows the space-domain story of how quadrature pair filters work. Sin(x) squared plus cos(x) squared is one, and that holds even if those sinusoids are windowed by a Gaussian, as the Gabor filters are. The filters convolved with an impulse, shown here, are just the filters themselves, so when we square and add the outputs, the filters, we get one everywhere, windowed by the Gaussian. It tells us how much power is in this sinusoidal frequency band, at the region of the image defined by our windowing Gaussian.

**slides 9, 10**: is the fact that the energy response to the line signal is wider than the energy response to the edge signal significant? I don't think so and I should remake these images to be sure it's not simply some artifact of how it was processed.

**slide 11**: let's go through how quadrature pairs work. Just to review gabor filters, we start with a sine or cosine wave. That gives two delta functions at complex conjugate complex exponentials. They have the same sign, for the cosine phase wave, and opposite signs, for the sine phase. Then we spatially localize those sinusoids by multiplying by a Gaussian envelope. This spatial localization broadens out the frequency response in the complementary domain, making those delta functions become little patches in the frequency domain.

**slide 12**: Now we square each filter response (multiply it by itself), in the spatial domain. (see the spatial picture for that on the right hand side). You know the drill: in the frequency domain, that means we convolve the fourier transforms with themselves, after transposing one of them. Now we get 3 lobes of responses: one at the origin, where everything overlaps, and two more, and (positive and negative) twice the frequency of the original lobes.

You can see that frequency doubling in the space domain images–the negative lobes of the gabor filters become new positive cycles of the squared filters. (The freq domain sketch is not drawn to scale and doesn't show the freq doubling as it should–those outer ellipses should be a bit further away.)

Then, here's the cool part of quadrature phase: when we add the squared sine and cosine phases together, those frequency doubled bands cancel out, leaving only this modulated-down baseband that tells you, at a low spatial frequency, how much energy there is in this region of the image in the spectral region that these gabor filters are sensitive to. Nice! (There are two sign inversions to get figure b: we transpose one copy of the H2 filter to perform the convolution, then remember that the odd-phase transform components are imaginary, and give us another factor of -1 with their product).

(you can see that you can get the same effect of a quadrature pair of filters if you square and then blur one phase of the filters. That low-pass filters the spectrum of (a) or (b), leaving just the baseband (c).)

**slide 13, 14**: These Gabor filters (and indeed, quadrature pair filters in general) are useful for many things. We can measure energy, as shown above, and we can also measure local phase, which is useful in identifying contour type (line or edge). With measuring phase, one application they've been applied to is quantifying the random textures of the human iris, an algorithm developed by John Daugman, at Cambridge University. The goal is to find a texture descriptor that is invariant to the various conditions under which one might acquire an image of an eye. The iris code measures the relative phase of a Gabor filter pair and quantifies that measurement into one of 4 bins (slide 119). Of course, the filters must be aligned with the features of the eye, and concentrically oriented around the eye. The result is a high-dimensional code for an individual's iris. This code is able to ascertain identity with very high certainty (and immune to any issues with identical twins, because their irises develop under unique random processes).

**slide 15** often the precise shape of the filter can be tuned differently than a gabor filter for one application or another. So let's look at oriented filters in general, and how to work with them.

**slide 16** one question that arises with oriented filters is how to move them in orientation. For a filter of a given shape, how many filters do you need?

What you'd like would be an analogy in orientation for the nyquist sampling theorem in space: given a certain number of discrete samples in orientation (or space), can you interpolate between the samples you have and synthesize what you would have found from having a filter (or a spatial sample) at some arbitrary, intermediate orientation? As with the spatial interpolation problem, it turns out you can, depending on restrictions on the form of the filter (or on the frequency content of the spatial signal).

**slide 17** The derivative filter is the simplest example. As we know from our differential calculus, we can synthesize a directional derivative in any direction as a linear combination of derivatives in the horizontal and vertical directions. By linearity, that applies to the derivative applied to any filter or image, as well. On the top row, we see the derivative of a gaussian, horizontal and vertical,, and one oriented at 30 degrees formed as a linear combination of those two. Again by linearity, the output of the 30 filter applied to any image just that same linear combination of the outputs of the appropriate basis filters applied to those images, as well.

**slide 18** Here we're showing the functional form of the Gaussian derivatives, and you can see those basis filters applied to a slightly more interesting image.

**slide 19** This leads to an architecture for processing images with oriented filters: you pass the input image through a set of "basis filters", then modulate the outputs of those filters with a set of "gain maps" (which can be different at each pixel). Those gain maps adjust the linear combinations of the basis filters to let you steer the input filter to the desired orientations at each position.

**slide 20** How many basis filters does it take to steer any given filter? You could imagine that will depend on sharply oriented the filter is. A circurly symmetric filter takes just one basis function to synthesize all other orientation responses, and a very narrow filter will take quite a few. This is quantified by steering theorems.

In particular, if we write the filter in polar coordinates (using complex exponentials for notational convenience), and write down an equation for the unknown steering functions of theta needed to synthesize the output filter from the basis filters, you can show the result stated in **slide 20**.

Let's check it for a simple example. Our derivative of a gaussian filter is x times a gaussian. This gives a cos(theta) angular distribution when written in polar coordinates. This requires two complex exponentials to write (to create the cos theta from complex exponentials) and thus requires two basis functions to steer.

**slide 21** sometimes its more convenient to think of the filters as polynomials times radially symmetric window functions. Then you can state the result listed in **slide 21**.

**slide 22, 23** for computational convenience, it's more convenient to have the basis filters all be x-y separable functions. In many cases, it's straightforward to find such basis functions (and where it's not, there are simple numerical methods to find the best fitting x-y separable basis set. See for example, Perona PAMI 1993).

**slides 24-28** What we'd really like is a steerable, separable quadrature pair of filters. We can design such filters. The G2 filter is a 2nd order, even polynomial in x-and-y times a gaussian. So its Hilbert transform will have the same spectral content, but the opposite phase. So we fitted a 3rd order odd polynomial to the Hilbert transform of the G2 filter, to make a steerable H2 filter (requiring 4 basis functions to steer, not just 3 as for G2). We can also make x-y separable versions of this filter.

Putting it all together, can we compute oriented energy as a function of angle, for all angles, just from the basis filter responses. This oriented energy as a function of angle is an analytic function of the basis filter responses, and results in a Fourier series in theta. We can look at the lowest order terms in that fourier series to find a measure of the dominant angle of oriented energy in the filters' frequency response band. This expression is given in the appendix of the PAMI 1991 paper referenced in the slides.

**slide 29**. We can also make contour detectors that fire independently of the phase of a contour, or are sensitive to it as you choose. Consider this figure. when we look at it, we easily parse it as a circle and a square, the circle formed by an edge, the square formed by a line. But if you take a conventional edge detector and apply it to that figure, the line-based figure puzzlingly because described as two edges. Of course, that makes sense, given what the edge detector is doing, but it doesn't make sense perceptually.

But with steerable quadrature pair filters, we can look for regions of local maximum energy oriented perpendicularly to the contour orientation. Those regions are marked here.

And we can also, if we choose, pull out contour regions of one particular phase or another, again just by looking at the quadrature pair filter responses when oriented on and along the contour. The 0 and 90 degree phase objects are shown here, pulling out the circle and square figures.

**slide 30**. From the basis filter responses we can form polar plots of the oriented energy as a function of angle.

**Slide 31** note some strange goings on at intersections using the G2H2 filters. You might think this was a result of simply not enough angular resolution from those filters, and indeed the G4 H4 filter pair doesn't suffer from that problem. But actually the G2H2 filters do have enough angular resolution, and the issue is a more subtle one.

**Slide 32, 33** when there are two oriented structures within the passband of the quadrature pair filters, the sum of the energies of the individual structures is not the same as the energy of the sum of the structures. Because we're squaring to find the energies, the combination of multiple structures isn't linear. As the figure shows, when there are two oriented structures within the passband, when the filter responses are squared, the convolution in the fourier domain picks up extra cross-terms from the one oriented structure interacting with the other, in addition to the desired term from simply squaring all the frequency responses individually within the passband. These cross terms show up as spurious spatial frequencies in the energy term, and we can get rid of them by spatially low-pass filtering the squared oriented energy responses. Using the blurred squared basis filter responses, we get much cleaner oriented energy as a function of angle plots, even with the G2H2 filters in the junctions (figure d of slide 31).

**slide 34, 35** Steerable filters let us filter *along* the dominant local orientation at each pixel, to enhance oriented structures and remove noise. These images have an additional enhancement, unrelated to steerable filters, which is very useful and often underutilized in low-level computer vision processing: contrast normalization. Every pixel value was normalized by the square root of a spatially blurred estimate of the local energy. There are many ways to do this, reflecting different choices in the output processing. Perhaps the simplest is to divide by the (square root of the) pooled oriented energy from all different filter orientations, averaged over some local neighborhood. The DC term of the oriented energy as a function of angle in slide 26 (table XI) can be used for this. You'll want to add some small positive constant to the pooled energy before you divide the steered filter responses by it, to avoid division by zero (or a small number) in very low contrast regions. That constant effectively sets the expected noise level of the bandpass filtered images.

**slide 36** We can also make steerable filters in 3 dimensions, allowing us to denoise medical volumetric data, or to analyze spatio-temporal volumes to measure image motion, as we'll discuss in the next section.

**slide 38** Here's a mock-up of a real scene. We have a stationary background, and two cars driving in opposite directions. A visual system may want to infer many different things with this video sequence, but a very useful low-level inference is to report back what is moving in the scene, and how is it moving?

There are many ways to do that, computationally, and our point here is to show that it is possible to measure such motion even with the simple processing machinery that we've developed so far.

**slide 39, 40** Let's consider the signal itself. Here is a 2-d slice of the volumetric video data of 2-d pixel intensities observed over time. In our x-t slice we notice what we learn in elementary geometry classes: the slope of the constant-intensity lines, change in time per change in x, indicate the speed of the observed pixels. (This indentification assumes constant pixel intensities over time). So identifying the *speed* of objects in the image is equivalent to finding the *orientation* of structures in the spatio-temporal volume. (In the full 3-d video volume, the 3-d orientation of constant pixel values indicates the *velocity* of motion).

**slide 41** But we just studied how to measure orientation with linear filters, so let's use the same approach and just re-label our axes. (We'll describe all this for 1-d spatial motions over time, but the ideas generalize to 2-d spatial motions over time, although with extra embellishments, such as the aperture effect, that we won't discuss now).

First of all, since we could learn so much about spatial filters by considering them in the Fourier domain, let's do the same for our spatio-temporal filters. Pop quiz: what is the locus of the non-zero components of the Fourier transform of a space time video signal?

First of all, what are the axes of that Fourier transform? $\omega_x$ and $\omega_t$, the spatial and temporal frequencies.

There's a real easy way to compute where the non-zero Fourier transform components of a signal like this lie. Pause.

We have the freedom to rotate the coordinate axes of our Fourier transform in whatever direction is convenient. Let's align our Fourier transform axes with the constant intensity lines of our signal. Now take the Fourier transform, separably along each coordinate axis.

First, along the constant intensity direction, the Fourier transform of each constant value is a delta function at the origin, of some height. Now let's take the Fourier transform along the other axis. Every coefficient of the Fourier transform we just took is zero, except for those along a line through the origin, perpendicular to the orientation of the constant pixel intensities. So the only non-zero Fourier components of the full, 2-d transform are along that line. (And for the volumetric signal, the non-zero Fourier components lie in the plane perpendicular to the pencils of constant intensity in the image volume. In degenerate conditions when there are no pencils of constant image intensity defined, the geometric ambiguity in defining the normal plane reflects the ambiguity in specifying the motion defined by the image intensity volume).

So we can use quadrature pairs of oriented filters in space-time to find motion speed and direction in the video signal. We just need to find the space-time orientation of strongest response, as we learned how to do above.

By the way, what do these space-time filters look like? What would we see as we observed the output of such a filter convolved with an impulse?

**slide 42** The fact that we find such space-time filters in neurophysiological studies of mammalian visual systems suggests that we may use such filter outputs in our own visual systems. Let's look at a visual demonstration which addresses that question.

**slide 43** Consider a square wave. From Fourier analysis, we know we can synthesize it from a sum of odd harmonics, as shown in the equation. Each frequency comes in with an amplitude of $\frac{1}{\omega}$, where $\omega = 2\pi f$ is its spatial frequency.

**slide 44** Consider the following signal over time: a square wave, translating $\frac{1}{4}$ cycle each frame of a movie. The signal in space-time would look something like this.

What are the different Fourier components of the square wave doing in this movie? The fundamental sine wave has the same frequency as the square wave itself, so it, too, is jumping by $\frac{1}{4}$ of a cycle (or $90°$) each frame.

The next harmonic, $\frac{1}{3}\sin(3\omega t)$ jumps by $270°$ each frame, or, equivalently, by $-90°$. It is aliased and, viewed in isolation, would appear to be moving in the other direction.

**slide 45** What if we make a funky, "fluted square wave" signal, the same as the original square wave, but with the fundamental sinusoid, $\sin(\omega t)$, subtracted out? The signal would look like this, in space-time.

The question is, how would we perceive it moving? The fluted square wave signal is really moving to the left, as the original sine wave was. But the dominant sinusoidal component (which is $\frac{1}{3}\sin(3\omega t)$, after we remove the fundamental from the square wave), viewed in isolation, is moving to the right.

So will we percieve the signal as an integral whole, and perceive it to be moving to the left? Or will we perceive it as if through a federation of oriented filters, each one telling us, for the spatial frequency it is tuned to, what is the speed and direction with the most motion energy? Will the visual system choose signal integrity, where it has to assimilate all the motion energy filter responses into a coherent story describing them all? Or will it choose computational expedience and just listen to the spatio-temporal energy response that shouts the loudest?

**slides 46, 47** Here is the answer. The visual system seems to take the easy (fast?) way out in this case, and signals to us which motion energy filter is responding the strongest: the fluted square wave stimulus appears to be moving to the right.

The quadrature pair motion energy filters, while very simple, are powerful enough to perform processing that mimics what a system as sophisticated as the human visual system does in some circumstances.

**slides 49 - 51** The result above is consistent with independent processing in a battery of different spatial frequency channels. There is other psychovisual evidence for such processing in the human visual system.

In static images, we have different sensitivity to different spatial frequencies (normalized for the distance away from the viewer, and thus typically given in frequency units of cycles per degree at the observer's eye). The spatial frequency chirp in slide 49 decays in amplitude uniformly from the bottom to the top of the image. By tracing out where you stop being able to see the sinusoidal variation, you can map out your own sensitivity to sinusoidal modulations as a function of spatial frequency.

For most people, their spatial frequency of maximum sensitivity is about 6 cycles per degree of visual angle.

So let's have fun with this. We can "hide" image content in spatial frequencies where you can't see them well, especially if they are masked or dominated by other image content presented at spatial frequencies to which we are sensitive. We can switch between what is hidden and what is visible by moving closer to or further from the image. This sort of display has been proposed by Oliva, Schyns and Torralba and dubbed "hybrid images" by them. One of the homework problems is to make such images.

**slides 58 - 60** Image information occurs over many different spatial scales. Image pyramids–multi-resolution respresentations for images–are a useful data structure for analyzing and manipulating images over a range of spatial scales. Here we'll discuss four different ones, in a progression of complexity.

The first is a Gaussian pyramid, which creates versions of the input image at multiple resolutions.

This is useful for analysis across different spatial scales, but doesn't separate the image into different frequency bands. The Laplacian pyramid provides that extra level of analysis, breaking the image into different isotropic spatial frequency bands. The Wavelet or QMF (quadrature mirror filter) pyramid provides some splitting of the spatial frequency bands according to orientation (although in a somewhat limited way). The Steerable pyramid provides a clean separation of the image into different scales and orientations. There are various other differences between these pyramids, which we'll describe below.

**slide 61**
We'd like to make a recursive algorithm for creating a multi-resolution version of an image. A gaussian filter is a natural one to use to blur out an image, since multiple applications of a gaussian filter is equivalent to application of a single, wider gaussian filter.

**slides 62 - 68** Here's an elegant, efficient algorithm for making a resolution reduced version of an input image. It involves two steps: convolving the image with a low-pass filter (for example, [1 4 6 4 1] / 16, separably in each dimension), and then subsampling the result.

Applied recursively, this algorithm generates a sequence of images, subsequent ones being smaller, lower resolution versions of the earlier ones in the processing.

**slide 67** shows a matrix which shows the contributions of each pixel of a (1-d) image in the Gaussian pyramid image two levels above it in the pyramid. Note the large spatial extent of the low-pass filtering, but accomplished much more efficiently than a naive filtering with the equivalent low-pass filter all in one step.

**slide 69** The Gaussian pyramid is useful for coarse-to-fine processing of images–using analysis at a coarse resolution to initialize processing at a finer level of detail.

**slide 70** Shows the equivalent matrix for a simple, 3-level Gaussian pyramid (the pyramid is actually constructed more efficiently than by performing this matrix multiplication). This matrix generates 3 versions of the image: one a full resolution, a second at half the (linear) resolution, and a 3rd at 1/4 the resolution of the original.

**slides 72-80** The Laplacian pyramid is simple: it represents, at each level, what is present in a Gaussian pyramid image of one level, but not present at the level below it. We calculate that by expanding the lower-resolution Gaussian pyramid image to the same pixel resolution as the neighboring higher-resolution Gaussian pyramid image, then subtracting the two. This calculation is made in a recursive, telescoping fashion. By storing a recursion-ending Gaussian low-resolution image, the original image can be reconstructed from its Laplacian pyramid representation (**slide 77**).

**slide 80** This is the equivalent matrix which creates a 1-d Laplacian pyramid from an input column vector (again, the actual computation is more efficient than is multiplying by this matrix).

**slide 81-84** The Laplacian pyramid is used in many image processing or analysis applications. Here we show one fun application: image blending (you'll do this in a homework problem). Making a sharp transition from one image to another gives an artifactually sharp image boundary (see the straight edge of the apple/orange in **slide 82** ). However, we can create gradual transitions within a Laplacian pyramid to make a gradual transition between the two images, as shown in the apple/orange in the bottom right of slides **slide 82-84**.

**slide 86-89** The Laplacian pyramid is an overcomplete representation (more coefficients than pixels). The wavelet (or sometimes called a quadrature mirror filter (QMF) pyramid, in the signal processing community) transform will be complete (instead of overcomplete), and adds some orientation tuning. The Haar transform is the simplest version, yet still has many of the properties of other wavelet transforms, so let's derive that first.

Suppose we have the submatrix, U. The first coefficient of the transform is just the average of the two input pixels, and the 2nd coefficient it the difference. Clearly, that's an invertible change of basis, and the inverse is shown here as $U^{-1}$.

**slide 88** One thing to note here is the aliasing of each subband, and its cancellation when the two subbands are combined. In both the high and the low bands, we're taking every other sample of a signal convolved with some two-tap filter. Those two-tap (high or low-pass) filters don't sufficiently pre-filter the signal to avoid aliasing in the resulting subsampled signal. So how do we obtain perfect reconstruction from those filter banks? Each band relies on the other one to supply the information needed to removing the aliasing that happens from one band alone. (We can easily see how the other band is used for reconstruction in the space-domain view of things. The Fourier domain explanation involves the aliasing components from each band that cancel each other.)

We can replicate this submatrix in different subspaces of a larger matrix, and re-order the rows to make it obvious that we're creating a low-pass version of the signal, and a high-pass one. **slide 89** shows another construction of the Haar transform matrix, and its inverse.

**slide 90** We apply the 1-d Haar transform separably in 2-d, which leads to: a low-pass band, a vertical high-pass band, a horizontal high-pass band, and a "diagonal" band, high-pass in both horizontal and vertical. Unfortunately, that last band isn't a rotation of either of the other two band-pass bands. The wavelet transform is great for compression, since it's "complete", with the same number of coefficients as pixels. For the Haar filter, reconstruction from the transform domain is exact, but for other filters it is only approximate.

**slide 98** The Haar filters are just the simplest example of an orthonormal wavelet, or QMF, filter set. Larger filters have only approximate image reconstruction, but have nicer frequency localization properties and are generally used instead of the Haar filters.

**slide 101**

The price we pay for the completeness property is that each subband is aliased. We want each subband, by itself, to tell us something abou the image components in a particular frequency band. But these aliased subbands don't tell us such a clean story. This figure is a simple demonstration of that. The top row shows an example signal, translated by one pixel from the left column to the right. the three rows below in each column show the wavelet/qmf filter subband representation . (see discussion at slide 88). At one position of the signal, our representation tells us that all the signal energy is confined to just one sub-band. But then we move the signal one pixel forward and the signal is represented as a splattering of coefficient values across different subbands. This limits the usefulness of complete orthonormal wavelets for various image analysis applications (for example, you couldn't compute stereo offsets between two images within such a representation).

**slide 102** Ideally, we'd like to have an image transformation that was shiftable–where we could perform interpolations in position, scale, and orientation using linear combinations of a set of basis coefficients. A representation that goes part way there is the steerable pyramid.

**slide 104-106**

We analyze in orientation using a steerable filter bank. We form a decomposition in scale by introducing a low-pass filter (designed to work with the selected bandpass filters), and recursively breaking the low-pass filtered component into angular and low-pass frequency components. Pyramid subsampling steps are preceeded by sufficient low-pass filtering to remove aliasing.

To ensure that the image can be reconstructed from the steerable filter transform coefficients, the filters must be designed so that their sums of squared magnitudes "tile" in the frequency domain. We reconstruct by applying each filter a second time to the steerable filter representation, and we want the final system frequency response to be flat, for perfect reconstruction.

**slide 107** There's a mis-match between rotation invariance and the rectangular pixel sampling lattice. The frequency domain is square, but a rotationally invariant image representation will have frequency subbands arranged as concentric circles. As with the other pyramid image representations, we have a residual, unoriented low-pass band, but the steerable pyramid also has a high-pass residual subband, as well.

**slide 108** Overall, the steerable pyramid representation is quite overcomplete. You can see that in this visualization of an image and its transform. If there are N steerable filters in the representation, then we have N images the size of the original at the highest frequency representation (plus 1 for the high-pass residual image), and extra coefficients are needed to represent the steerable subbands at the coarser levels of the pyramid, plus the final low-pass residual image.

So we may not want to use such a representation for image compression applications (although the over-complete Laplacian pyramid has been used for compression). But it is useful for various image and texture analysis applications, since the subbands represent what they are advertised to represent–the signal components in a particular frequency band.

**slide 114-120** Here is a pictorial summary of the different pyramid representations we've been discussing during this lecture. To recap briefly: The Fourier transform reveals spatial frequency content of the image wonderfully, but suffers from having no spatial localization. A Gaussian pyramid provides a multi-scale representation of the image, useful for applying a fixed-scale algorithm to an image over a range of spatial scales. But it doesn't break the image into finer components than simply a range of low-pass filtered versions. The representation is over-complete that is, there are more pixels in the Gaussian pyramid representation of an image than there are in the image itself.

The Laplacian pyramid reveals what is captured at one spatial scale of a Gaussian pyramid, and not seen at the lower-resolution level above it. Like the Gaussian pyramid, it is overcomplete. It is useful for various image manipulation tasks, allowing you to treat different spatial frequency bands separately.

A wavelet/QMF pyramid brings in some limited orientation analysis, and, different than the other pyramid representations, is complete, rather than overcomplete. This helps it for image compression applications, but hurts it for others, since each subband depends on information in other subbands to let it reconstruct the original image without artifacts. So if you alter one band without altering the corresponding other ones, you can easily introduce artifacts.

A steerable pyramid representation can be very overcomplete, depending on the number of orientations represented, but has negligible aliasing artifacts and so can be useful for various image anlaysis applications.