Implementation of Velocity-Tuned Filters and Image Encoding

David J. Fleet

FBI-HH-M-159/88

Juli 1988

Fachbereich Informatik
Universität Hamburg
Bodenstedtstrasse 16
D-2000 Hamburg 50

Abstract

Issues concerning the design, implementation, and utility of 3-d velocity-tuned filters are discussed. The filters are linear, shift-invariant, tuned to narrow ranges of scale, orientation and normal speed, and they use only local support in space-time. This report concentrates on the design and implementation of an entire family of such filters that collectively yield a complete and efficient image representation for the initial stages of visual processing. The implementation nows serves as a starting point for research in early motion analysis and understanding.

Zusammenfassung

Wir diskutieren Design und Implementation von 3-dimensionalen geschwindigkeitsselektiven Filtern. Die Filter sind linear und ortsinvariant. Sie sind jeweils eingestellt auf einen bestimmten Auflösungsbereich, eine bestimmte Orientierung und eine bestimmte Normalgeschwindigkeit. Wir beschreiben Design und Implementation einer gesamten Filterfamilie, welche eine vollständige und effiziente Scenenrepräsentation auf einer frühen Verarbeitungsebene liefert. Die Implementation wird angesehen als ein Startpunkt für weitergehende Untersuchungen im Bereich Bewegungsanalyse und Bildverstehen.

Implementation of Velocity-Tuned Filters and Image Encoding

David J. Fleet[1]
Fachbereich Informatik
University of Hamburg
Bodenstedtstr. 16
D-2000 Hamburg 50

1 Introduction

A definition of image velocity in the frequency domain, and the use of linear shift-invariant filters tuned to specific ranges of speed and orientation (often accompanied by energy extraction) have appeared in several recent publications [Adelson and Bergen, 1985, 1986; Fleet and Jepson, 1984, 1985, 1988; Heeger, 1987; Watson and Ahumada, 1985; also see van Santen and Sperling, 1985]. Although the initial results appear favourable the approach is somewhat novel to both the machine and biological vision communities and therefore several questions remain unanswered. For instance, how should such filters be designed, and implemented? More importantly, how should their output be interpreted? The goal here is not to answer these questions definitively, but to provide some perspective, and facilitate further research.

This technical note is aimed in two directions, the goal of which is the construction of a family of spatiotemporal filters that yield a rich representation of time-varying image intensity. Section 2 provides a short discussion of the relevant theoretical issues surrounding the design of such a representation, and briefly touches on the utility of this approach with respect to subsequent stages of processing. (Much of Section 2 follows from [Fleet and Jepson, 1988].) Section 3 then concentrates on the implementation of one exemplary family of velocity-tuned filters. This covers the non-trivial aspects of the implementation issues, as well as a variety of tools which might allow extensions of the current framework. In general I view the system as the beginnings of a test-bed for research on such filters and subsequent stages of interpretation.

This report provides only a brief review of the underlying filter theory. For a good intuitive introduction to the one-dimensional case see [Adelson and Bergen, 1985]. A more detailed mathematical account, including the two-dimensional case, can be found in [Watson and Ahumada, 1985] and [Fleet and Jepson, 1984]. [Heeger, 1987] is also worth reading. Below, I

[1]Department of Computer Science, University of Toronto, 10 King's College Road, Toronto, Ont., Canada, M5S1A4

include portions of the appropriate mathematical details that are specific to the implementation, or that are not documented well in other references. The implementation was done on a Symbolics Lisp machine using **IMAGE-CALC**[2] and Zetalisp.[3] I assume the reader has a basic knowledge of Lisp, and of IMAGE-CALC.

2 Theoretical Issues

2.1 Image Velocity

Following Horn (1986), let the *motion field* be the perspective projection of object (or world) velocity onto an image plane, perpendicular to the optical axis of an ideal pin-hole camera. The *image velocity field*, sometimes referred to as the optic flow field, may be defined qualitatively as the apparent velocity of intensity structure in the image sequence.[4] One research objective in early vision is to develop a definition of image velocity that allows inference about its associated motion field, and yields an efficient and robust measurement technique from the image. Aspects of the motion field that are then inferred from the measured image flow can then, in principle, be used to facilitate the determination of object surface structure and egomotion. Unfortunately, but not surprisingly, such a definition of image velocity has been elusive.

Recently, several researchers have suggested that the frequency domain yields a useful definition of image velocity. This is premised on the observation that linear structures in the image correspond to linear subspaces in the frequency domain, which, in the cases of orientation and velocity, are lines and planes containing the origin. *Orientation* in space corresponds to a line in the frequency domain in that all the power (non-zero Fourier components) associated with an oriented pattern of image intensity must lie on a line through the origin. The orientation of the line varies continuously with the orientation of the structure in the image.

In practice we are only concerned with intensity structures which are oriented in small neighbourhoods (windows) of the image. When viewed through such local windows, however, the non-zero power associated with the oriented structure will not lie strictly along a line, but will be distributed throughout the neighbourhood about such a line. More precisely, viewing a straight-edged intensity profile through a local image region near $\vec{x}_0$ amounts to a multiplicative window operation,

$$I_w(\vec{x};\vec{x}_0) = W(\vec{x};\vec{x}_0)I(\vec{x}). \tag{1}$$

This operation amounts to the convolution of the Fourier transform of the image with the Fourier transform of the window. The window acts as a low-pass filter in the frequency domain, blurring the amplitude spectrum of the image. This is useful in discussing interactions between localization and directional tuning as seen below.

[2]IMAGE-CALC is a trade-mark of SRI International.

[3]Zetalisp is a trade-mark of Symbolics.

[4]Of course, pixel intensities do not actually move. Rather we perceive motion in their patterns of change over time.

In space-time the situation is very similar. Consider a 2-d intensity pattern $I_0(\vec{x})$ translating in the image plane with velocity $\vec{v}$, that is,

$$I(\vec{x},t) \;=\; I_0(\vec{x} - \vec{v}t)\;, \tag{2}$$

where $\vec{x} = (x,y)$ and $\vec{v} = (v_x, v_y)$. Its Fourier transform is given by

$$\hat{I}(\vec{k},\omega) \;=\; \hat{I}_0(\vec{k})\;\delta(\omega \;+\; \vec{v}\cdot\vec{k})\;, \tag{3}$$

where $\hat{I}_0(\vec{k})$ is the Fourier transform of $I_0(\vec{x})$, and $\vec{k} \equiv (k_x, k_y)$ and ω denote spatial and temporal frequencies. Analogous to the case with orientation information, this means that all the non-zero power associated with a translating 2-d pattern will lie on a plane containing the origin in the frequency domain since $\delta(\omega \;+\; \vec{v}\cdot\vec{k})$ is non-zero only when $\omega = -\vec{v}\cdot\vec{k}$. The speed, $|\vec{v}|$, determines the angle between the two planes $\omega = -\vec{v}\cdot\vec{k}$ and $\omega = 0$. The direction of $\vec{v}$ determines the orientation of the velocity plane about the ω-axis. Finally, there is a one-to-one correspondence between finite 2-d image velocities, $\vec{v}$, and planes intersecting the origin in frequency space which do not contain the entire ω-axis.

The utility of this definition is, however, somewhat limited in that 2-d translation (2) will only be a good approximation to image motion locally. Because of perspective projection in the image-forming process and non-rigidity, 3-d object motion will often produce significant deviations from 2-d image translation. Therefore any measurements based on image translation should be restricted to narrow spatiotemporal windows. Furthermore, within such local windows spatial intensity changes will often appear straight-edged; that is, we only expect to find small edge fragments. In this case we can only *expect* to measure velocities normal to local orientation information. This is commonly referred to as the aperture problem (e.g., see [Horn, 1986]).

Let $\vec{v}$ denote a 2-d image velocity, and let $\vec{n}$ denote the unit normal to some local oriented spatial intensity structure, then the normal velocity $\vec{v}_n$ is given by

$$\vec{v}_n \;=\; (\vec{n}\cdot\vec{v})\;\vec{n}\;. \tag{4}$$

Moreover, all non-zero Fourier components $(\vec{k},\omega)$ of an oriented pattern with normal velocity $\vec{v}_n$ satisfy

$$(\vec{k},\omega) \;=\; c\;(\vec{v}_n,\; -|\vec{v}_n|^2)\;, \qquad c \;\in\; \mathcal{R}\;. \tag{5}$$

This line is contained in the velocity plane associated with the 2-d image velocity $\vec{v}$, and it intersects the origin. It's slope (relative to the plane $\omega = 0$) corresponds to speed, and the direction of motion determines it's orientation about the ω-axis. Finally, there is a one-to-one mapping between lines intersecting the origin in frequency space and 2-d normal velocities, $\vec{v}_n$.

The aperture problem may be simply interpreted in terms of the infinite number of velocity planes that contain a single normal velocity line [Fleet and Jepson, 1984,1985a; Watson and Ahumada, 1985]. In fact, as noted above, for windows large enough to capture two distinct normal velocities and thereby a unique 2-d image velocity, 2-d image translation may no longer be a useful basis for our measurement primitives.

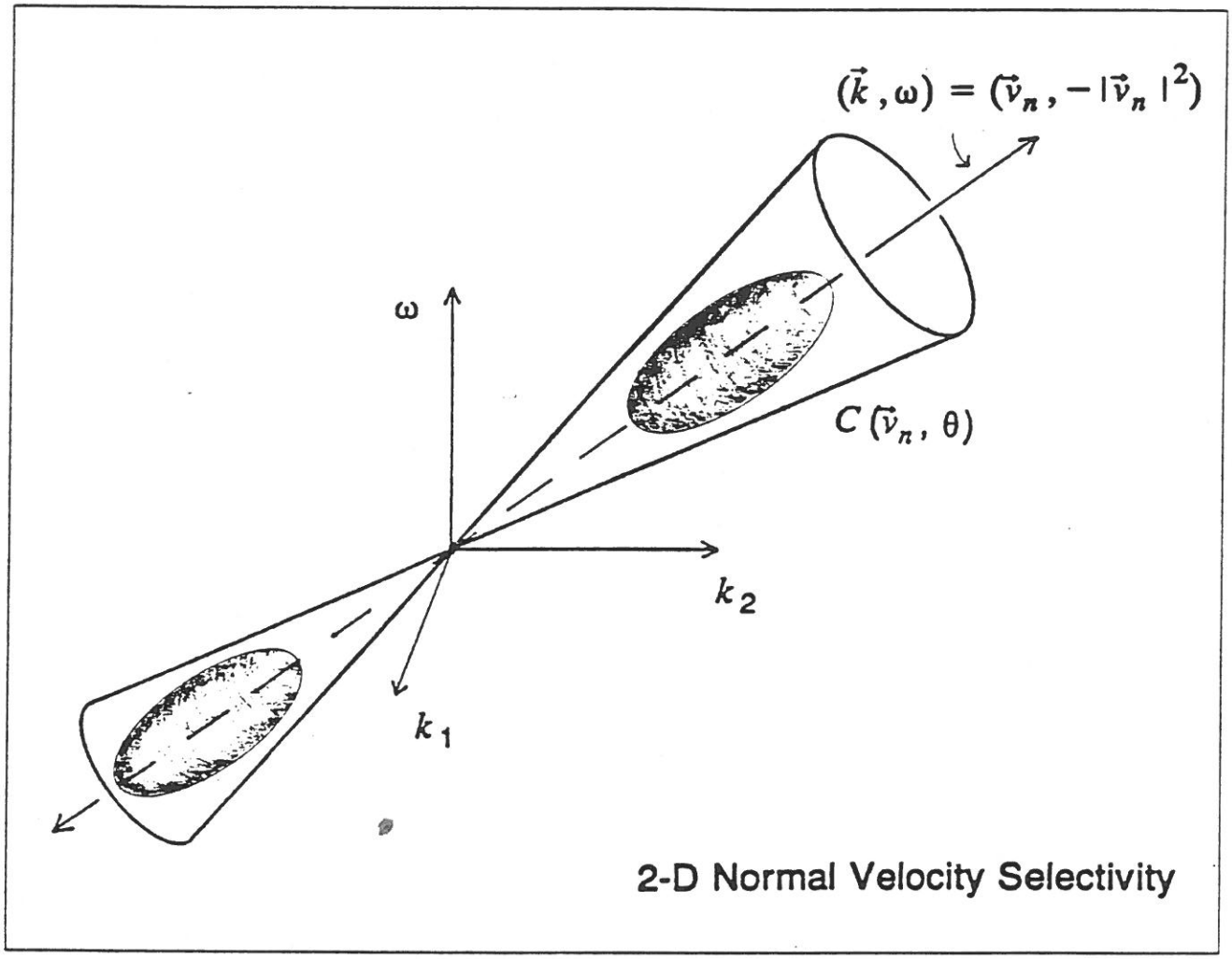

Figure 1: The shaded region represents the tuning of a velocity-tuned filter in the frequency domain. As discussed in the test, the amplitude spectrum for such a filter should fall mainly within a cone, the opening angle of which determines directional tuning.

2.2 Basic Design Constraints

One principal virtue of these definitions of orientation and image velocity information, other than their intuitive appeal, is the ease with which simple methods of extraction or measurement can be derived based on linear shift-invariant filters of the form

$$R(\vec{x}_0) = \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} K(\vec{x} - \vec{x}_0)\, I(\vec{x})\, d\vec{x} , \tag{6}$$

where $I(\vec{x})$ is the input image, and $K(\vec{x})$ is the convolution kernel. But how should such filters be designed, and implemented?

At present the basic design constraints come mainly from intuition. (In the future one might hope for more well-founded criteria.) They are:

- Localization in Space-Time. The response $R(\vec{x}_0)$ should depend mainly on the local structure of the image $I(\vec{x})$. To ensure this we assume that the magnitude of the impulse response function $|K(\vec{x})|$ is small for large values of $|\vec{x}|$. In particular, let $K(\vec{x}) = K_0(\vec{x})\, W(\vec{x})$ for some bounded function $K_0(\vec{x})$, where $W(\vec{x})$ is a windowing function. Then, (6) can be re-expressed as $K_0(\vec{x})$ applied to the windowed input, i.e.,

$$R(\vec{x}_0) = \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} K_0(\vec{x} - \vec{x}_0)\, I_w(\vec{x};\ \vec{x}_0)\, d\vec{x} \tag{7}$$

  where $I_w(\vec{x};\ \vec{x}_0)$ is defined above in (1).

- Orientation/Velocity Specificity. The filter should be tunable to a narrow range of orientation or normal velocity, with its amplitude spectrum concentrated about a line through the origin in frequency space. For example, for velocity specificity, the amplitude spectrum should fall mainly within a cone, $C(\vec{v}_n,\ \theta)$, about a line with a relatively small opening angle θ (see Figure 1).

There are restrictions on the use and satisfaction of these two conditions. First, localization is meaningful only in relation to the scale at which one-dimensional orientation and normal velocity are reasonable approximations to local image structure; in effect, for general utility, the second condition requires the first. Consequently, the measurements should be scale specific, with the scale related to the window size. Second, the well-known uncertainty relation imposes finite theoretical limits below which the two criteria cannot be satisfied simultaneously [Slepian, 1981; Bracewell, 1978, p.160]; these are discussed further below.

Several filter designs have been suggested in the literature (e.g., [Fleet and Jepson, 1984, 1988; Adelson and Bergen, 1985, 1986; Watson and Ahumada, 1985; Heeger, 1987]). A common, analytically convenient form is a Gaussian windowed sinusoid, called a Gabor function [Gabor, 1946], i.e.,

$$Gabor(\vec{x}, t;\ \vec{k}_0, \omega_0, C_{xt}) = e^{-i(\vec{x}\cdot\vec{k}_0 + t\omega_0)}\, G(\vec{x}, t; C_{xt}) , \tag{8}$$

where

$$e^{\pm i(\vec{x}\cdot\vec{k}_0 + t\omega_0)} = \cos(\vec{x}\cdot\vec{k}_0 + t\omega_0) \pm i\sin(\vec{x}\cdot\vec{k}_0 + t\omega_0) \tag{9}$$

is a complex exponential, and G denotes a 3-d Gaussian distribution with covariance matrix C_{xt}, i.e.,

$$G(\vec{x},t;\ C_{xt}) = \frac{|C_{xt}|^{-1/2}}{(2\pi)^{3/2}} e^{-\frac{1}{2}(\vec{x},t)' C_{xt}^{-1}(\vec{x},t)} , \tag{10}$$

where A' denotes the matrix transpose of A. The Fourier transform of a Gabor function is simply a Gaussian envelope in frequency space centred at $(\vec{k}_0, \omega_0)$, i.e.,

$$\hat{Gabor}(\vec{k},\omega;\ \vec{k}_0,\omega_0,C_{xt}) \ = \ \hat{G}(\vec{k}-\vec{k}_0,\omega-\omega_0;\ C_{xt}) , \tag{11}$$

where $\hat{G}$ denotes the Fourier transform of a Gaussian, which is also a Gaussian but with covariance matrix C_{xt}^{-1}. The Gaussian in (8) determines the profile of the amplitude spectrum, and the sinusoidal modulation (the complex exponential in (8)) determines its placement in the frequency domain. If the Gaussian has a diagonal covariance matrix then the resulting amplitude spectrum is symmetric about lines that run parallel to the spatial or temporal frequency axes. If $C_{xt} = \sigma \mathrm{I}$ then it is isotropic.

Although, most of the velocity-tuned filters suggested to date are constructed using separable first stages followed by linear combinations resulting in inseparable real kernels, their shapes still basically correspond to the case in which C_{xt} is diagonal. Fleet and Jepson (1985a, 1988) described filters with non-unit aspect ratios with radial tuning to velocity. In other words, the amplitude spectra are elongated along lines through the origin that correspond to the preferred normal velocities. For example, in 2-d the elongated Gabor function can be written as

$$e^{i\vec{x}\cdot\vec{k}_0} \ \frac{\alpha}{2\pi\sigma^2} \ e^{-\frac{1}{2}\sigma^{-2}\vec{x}' R_\theta' D_\alpha R_\theta \vec{x}} , \tag{12}$$

where $D_\alpha \equiv \mathrm{diag}(1,\alpha^2)$, and R_θ is a simple rotation matrix, i.e.,

$$D_\alpha \equiv \begin{pmatrix} 1 & 0 \\ 0 & \alpha^2 \end{pmatrix} , \qquad R_\theta \equiv \begin{pmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{pmatrix} .$$

Here, the orientation about which the filter is tuned is determined by the sinusoidal modulation; it is $\arctan(-k_x/k_y)$. Also, α is the aspect ratio, σ^2 is the variance (or measure of extent) of the spatial window along the preferred orientation, and σ^2/α^2 is the variance in the perpendicular direction. Accordingly, the rotation matrix serves to align the elongation of this envelope with the orientation of the modulating sinusoid. Therefore $\theta = \arctan(-k_x/k_y)$.

Such elongation has been common in the orientation case (e.g., see [Sakitt and Barlow, 1982; Daugman, 1985]), but not in space-time. As a consequence, velocity tuning is typically quite poor.

2.3 Family Design Issues

Our goal is not simply the design of single filters as might be infered from the discussion of design constraints above. Rather, the goal of image measurement is the construction of a rich description of image intensity structure. This section provides a review of the major issues

involved in designing an entire representation. (Because more research is still needed this section is somewhat terse and incomplete. Also, some details are left until Section 3.)

The first basic constraint, as indicated already, is that the primitives of the representation should be *generic* in that they capture properties that are often useful for subsequent processing and are robustly available for measurement. Typically these aims are met by restricting primitive measurements to those aspects of intensity structure that reflect, as much as possible, salient and independent properties of the scene. In particular, potentially independent scene properties should not be unnecessarily confused (confounded) in single measurements. It is in this sense that localization and directional tuning are basic constraints. Through localization an attempt is made to ensure that neighbouring yet disjoint scene properties are not merged accidently into a single measurement thus confounding independent pieces of information. Through directional tuning an attempt is made to isolate manifestations of useful scene properties, and to ensure that potentially independent orientations are not confounded or ignored. Within any local window there is little reason to suspect any statistical dependence over a wide range of orientations. Furthermore, it is reasonable to assume *a priori* that no specific local window should be treated in some special way; i.e., shift-invariance in (6) is also reasonable.

Further constraints on image measurement include image-independence, completeness, and efficiency. As explained in Fleet and Jepson (1985b), it is preferable to avoid the need for restrictive *a priori* assumptions as well as previous or concurrent interpretation during the measurement process. The measurement primitives will of course be based on a model of expected image structure, but the process should not fail or degrade considerably if the model is inappropriate. Completeness ensures that all the relevant information is accessible, despite the relevance of the model in certain situations.

Efficiency is also a major issue [Sakitt and Barlow, 1981; Binford, 1983; Langer 1988]. Given the voluminous amount of potential information contained in the initial pixel-based image description it is natural to attempt to capture as much structure as possible using only a small number of primitives. In essence, this amounts to searching for a good model of expected structure. If redundancy is at some point desirable, it should at least be controlled during the design process.

Information theory is an appropriate perspective from which to view this problem [Shannon, 1949; Gallager, 1968; Gabor, 1946; Brillouin, 1962]. We may view an $n \times n$ image as a vector in an n^2 dimensional vector-space; it has n^2 degrees of freedom. One can regard the pixel-based spatial encoding as one in which the basis functions used to span image-space are ideally localized in space-time yet their amplitude spectra span frequency space. Conversely, in a Fourier decomposition the basis functions are ideally localized in the frequency domain yet they span all of space-time as sinusoidal waveforms.

Along these lines the set of linear shift-invariant filters are simply a new basis for the image vector-space, and the convolutions represent a change of basis, that is, a linear transformation. Each convolution result at a specific location (7) is the inner product (dot product) of the filter kernel (support or impulse response function) centred at that location with the image. Thus, the new basis functions are the support functions placed at certain locations in space-time, and the coefficients of the resulting representation are the convolution outputs at those points. Note that every filter in the eventual representation cannot yield an output at each pixel

location since with κ different filters the total number of coefficients would be κn^2 instead of n^2. Instead the rate at which each filter output is sampled should be inversely proportional to the extent of its amplitude spectrum.

For completeness the new basis set should contain at least n^2 basis functions, and for efficiency it should contain no more than that.[5] In addition, for efficiency, the new basis functions should measure statistically independent types of image structures so that the resulting coefficients (filter outputs) will contain no mutual redundancy. However, it is virtually impossible to ensure the independence of different filter outputs and therefore we relax our demands and ask only for uncorrelated coefficients. One result of such a constraint is that the basis functions should be mutually orthogonal. In terms of the filters discussed above which are localized in space-time and the frequency domain, we take this to imply that they be non-overlapping both in space-time and in the frequency domain.

Unfortunately, as noted in Section 2.2 there is a theoretical uncertainty relation for signals that places a lower bound on the simultaneous localization of a signal and its amplitude spectra. A middle ground, with good simultaneous localization in both domains is obtained using Gabor kernels (8).[6] In effect this uncertainty relation tells us that there is a limited amount of information that can be measured within windows of finite extent. This also makes sense in terms of degrees of freedom; e.g., a 4×4 window only contains 16 degrees of freedom, and as a consequence we can only expect to resolve 16 independent measurements.

This decrease in the information that may be measured as window size decreases also follows from (1) and (7). Consider $I_w(\vec{x};\ \vec{x}_0)$ with $\vec{x}_0 = \vec{0}$ for convenience, and let the window be an isotropic Gaussian with covariance matrix $C_{xt} = \sigma_w \mathrm{I}$. Then, the Fourier transform of the windowed input (1) is

$$\hat{I}_w(\vec{k};\ 0) \ = \ \hat{G}(\vec{k};\ \sigma_w) \ * \ \hat{I}(\vec{k}) \ . \tag{13}$$

For relatively smooth windows, the result is a low-pass smoothing of the image's Fourier transform. As the transform becomes more and more blurred the frequency spectrum may be sampled less densely and information (or resolution) is lost.

Fleet and Jepson (1988) use this to illustrate the interaction between localization, scale and directional tuning. They show that for a fixed window size the potential number of independent orientation measurements for a small range of frequencies (for $|\vec{k}|$ near some f_0) within the window increases nearly linearly with frequency. A simple way to impose the independence of the channels is to arrange their amplitude spectra so that they do not overlap significantly (see Figure 2). As a measure of the radius of the transform, take one standard deviation of $\hat{G}$, that is $R \ \equiv \ 1/\sigma_w$.[7] The opening angle of the orientation cone, which just contains one

[5] Note however that with the goal of efficient coding the significant savings come from discarding those basis functions with negligible expected power. The threshold of course should be a function of the tolerable acceptable degradation and expected noise. This may also be desirable for computer vision although this is beyond the scope of this paper.

[6] Using variance as a measure of extent in space and frequency space a Gaussian is optimal [Slepian, 1982].

[7] Given a fixed number of degrees of freedom and the uncertainty relation it is easy to show that in order to maintain the same coverage (or overlap) of windows in space and in the frequency domain one standard deviation can be used as a measure of extent. If the corresponding convolution results are sampled more densely (thereby taking spatial extent to be less than one standard deviation) then the number of different types of filters used to tile frequency space must decrease yielding less overlap, and holes in frequency spectra.

standard deviation of $\hat{G}$ centred at $(f_0, 0)$, can be approximated by

$$\theta \approx 2 \arcsin(R/f_0) \ . \tag{14}$$

It follows that the maximum number of orientation channels, centred at frequency f_0 and separated by at least one standard deviation in frequency space, is $N = \lfloor \pi/\theta \rfloor$, where $\lfloor x \rfloor$ denotes the integer part of x.[8] For example, consider a frequency f_0 such that the associated period, $2\pi/f_0$, is equal to the spatial diameter of the window, $2\sigma_w$ (i.e. approx. 1 octave). This gives $N = 4$. For higher frequencies it is possible to resolve more orientations, and in fact the number of possibly independent orientation channels grows nearly linearly with the frequency. A similar argument shows that the number of independent normal velocity channels that can be obtained through a given spatiotemporal window, grows quadratically with the frequency $f_0 \equiv |(\vec{k}, \omega)|$.

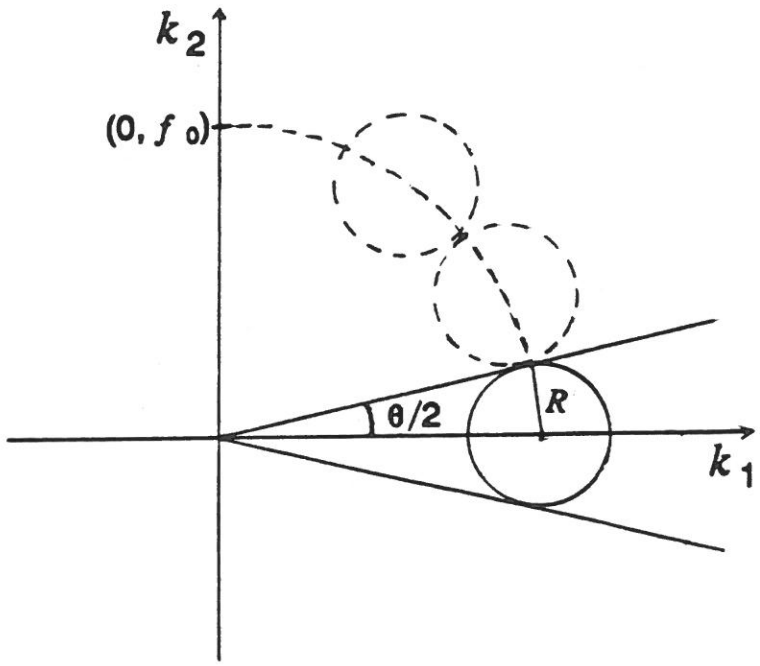

Figure 2: To arrange a collection of directionally-tuned filters that measure independent information, we consider non-overlapping amplitude spectra. Given bandwidth and aspect ratio (here 1.0), this gives a constraint on directional resolution and the appropriate number of filters.

This analysis can easily be extended to the elongated kernels with non-unit aspect ratios in (12). By equating the transform radius of the isotropic case (R in (14)) to the radius

[8] Note that we need only cover orientations ranging over 180 degrees.

along the line of maximal orientation tuning which is the major axis of elongation, we find that the radius in the orthogonal direction which controls orientation tuning is simply, R/α where $\alpha \geq 1$ is the aspect ratio. In comparison to the isotropic Gabors above with similar octave bandwidths (a function of R and f_0) it is easy to show that the number of independent resolvable orientations is about a factor of α larger, i.e. $N \approx \lfloor \alpha\pi/\theta \rfloor$.

The above analysis shows that there are limits to the resolution with which information can be measured in small windows. In part, this appears as a trade-off between window size and the possible number of resolvable orientations. We now turn to consider ways in which entire families of such filters may be designed, thereby covering the entire frequency spectrum with directionally selective units. A useful perspective on this issue follows from the approach of Gabor (1946) in viewing the set of new basis functions as a tiling (or paving) of space-time and the frequency domain simultaneously (also see [Burt and Adelson, 1983; Geisler and Hamilton, 1986]). Towards this end we consider the collection of amplitude spectra along with the way in which their respective support functions and used throughout space. For convenience I will continue to consider Gabor functions as exemplary units. In doing so however note that this, in itself, provides *very few* constraints; the nature of the filters' tuning to different scales (i.e., distance from the origin in frequency space), their directionality, and their aspect ratios have yet to be considered in terms of the entire family.

One approach toward an entire encoding is to emphasize localization in the frequency domain with a uniform coverage of isotropic amplitude spectra. In terms of Gabors this is a uniform tiling of the frequency domain with Gaussian envelopes of constant variance. This may provide excellent tuning to scale as well as directional information, especially for relatively high frequencies (cf. Fig. 1). Furthermore, since all amplitude spectra have the same envelope, so do the corresponding kernels, and therefore all of their respective convolution outputs can be sampled at the same reduced rate (this rate is equivalent to that possible with a Gaussian kernel with similar extent as explained in Sections 3.2 and 3.3). Although similar in Gaussian envelope, the different kernels vary according to the orientation and scale to which they are tuned. For high frequencies the kernel profiles will modulate quite rapidly. In effect, this scheme is equivalent to first decomposing the image into a set of non-overlapping Gaussian windowed image regions as in (1), and then taking the Fourier transform of each windowed region. In other words, if each filter output is sampled on the same sampling lattice as the window centres, then the filter outputs at each sample point will be identical to the Fourier transform coefficients of the corresponding windowed region.

One advantage of this particular tiling is the simplicity and efficiency of an implementation that follows from the separability of the individual kernels as well as the entire collection into horizontal and vertical components. A single horizontal 1-d stage of processing may serve as a precursor to the entire set of kernels sharing the same horizontal tuning. On the other hand, this representation violates our first basic constraint above because high frequency structure is measured (integrated) over relatively large windows. This is most evident in the rapid modulation of the kernel profile for kernels tuned to high frequencies. Above it was proposed that scale should be matched to window size; i.e., image structure should be measured within the smallest reasonable window sizes as a function of scale.

To accommodate this constraint, we consider a second scheme in which the windows decrease in size as the frequencies being measured increase. In the frequency domain this corre-

sponds to an increase in transform extent with frequency. This suggests an octave bandwidth invariance; in other words, that all amplitude spectra have the same extent when measured on a logarithmic scale. Such an approach has been adopted in other studies for a variety of reasons (e.g., see [Sakitt and Barlow, 1981; Watson, 1983; Crowley and Parker, 1984; Koenderink, 1984; Langer, 1988; Field, 1987]). Also note the trade-off between sampling in space-time and in the frequency domain with respect to this approach. As the window sizes decrease, the convolution output must be sampled at a higher rate. This can also be attributed to the larger amplitude spectra, as noted above. Certainly with fewer filters we must sample more frequently to maintain the correct number of coefficients. In terms of the orientation-specific channels, constant octave bandwidth means a fixed relationship between amplitude spectrum extent and scale. According to the arguments above this means that the number of resolvable orientation channels is also fixed at all scales. Therefore, in this scheme, orientation and scale specificity do not increase with scale, but space-time localization does.

As a third alternative consider a representation whose basic filters have non-unit aspect ratios (12). This situation can be viewed as a combination of the two schemes above, one of which emphasizes localization in the frequency domain, and the second which emphasizes localization in space-time. In comparison to the second scheme the extents of the amplitude spectra increase similarly with higher frequencies but only radially along the orientations or normal velocities to which the filter is tuned. This increase in extent with frequency allows better localization in space-time in the direction normal to the orientation (and speed) to which the filter is tuned. Conversely, we now have better localization in the frequency domain in angular terms giving finer orientation tuning like the first scheme above. As a result of this increased directional tuning, however, spatial localization along the prefered orientation is much poorer. Essentially we have traded scale specificity for spatial localization normal to local "edge" segments, and localization along such edges has been sacrificed for finer orientation tuning.

Due to the inseparability of these units an implementation is not as readily straight-forward as with isotropic kernel envelopes. However, a hierarchical approach is feasible in which a first level of processing consists of a set of band-pass filters yielding a scale-specific decomposition. This may be done using simple separable filters in 2-d as well as 3-d [Burt and Adelson, 1983; Fleet and Jepson, 1985a, 1988]. The second stage then decomposes each scale-specific band into directionally tuned channels using simple 1-d inseparable convolutions [Fleet and Jepson, 1988].

Thus it is clear that several alternative representational approaches exist. A choice among them depends on several factors which are not clear at present. One principal factor involves the probability distribution over the image ensemble of interest, in that filter properties should be well matched to average correlation lengths and 3rd order statistics [Langer, 1988]. In particular, the kernel envelopes should be matched to the spatiotemporal extent over which constant image velocity (or orientation) is a good approximation to expected image structure (on average). From this we may derive the appropriate number of orientation and speed specific channels, and their tunings. Also of interest is the stationarity of the ensemble with respect to scale and directional information. If reasonably stationary, a homogeneous tiling scheme is appropriate, where for example, we might have the same number of orientation channels at all scales, or scale-specific bands all with the same bandwidth.

A second principal factor affecting these design decisions comes from the ways in which the representation is to be used, as our goal is certainly to facilitate subsequent stages of processing. At present, however, we have few constraints other than some of the intuitions offered here. We now turn to consider the possible nature of subsequent processing in more detail.

2.4 Interpretation Issues

As discussed above, we view the goal of early measurement to be a rich encoding of image structure in terms of generic properties from which more complex structures are more easily detected and analysed. Such measurement processes should be image-independent and require no previous or concurrent interpretation. In these terms it is useful to contrast spatiotemporal filtering with other approaches to the extraction of image motion [Fleet and Jepson, 1985b].

For example, token-matching techniques (e.g., [Barnard and Thompson, 1980; Dreschler and Nagel, 1982]) consist of the two distinct stages: i) the extraction of tokens in isolated frames; and ii) the temporal matching (or correspondence) of similar tokens in successive frames. The first stage alone involves a significant amount of spatial scene interpretation in order to identify tokens, while removing noise and other irrelevant features. In particular, token-based techniques, although not extremely well defined, are typically associated with a matching process applied to a sparse set of monocular features. One principle issue is whether or not the tokens are (in practice) sufficiently rich to capture the image structure so that some important motion does not go undetected.

Current gradient-based approaches in which spatial and temporal gradients are used to solve for translational image motion (e.g., [Horn and Schunck, 1981; Nagel, 1983]) are also problematic. Velocity estimates are often restricted to specific spatial contours since unique spatial and temporal gradients are required locally (usually only those with large amplitude, high degrees of curvature, or consistency over several spatial scales). Moreover, the particular form of the velocity constraint equation used depends on various assumptions about the local flow. The standard assumptions are either *i)* unique local 2-d translation , or *ii)* conservation of image features [Schunck, 1985].

The use of spatiotemporal filters to extract motion information does not depend on the recognition of particular types of spatial structure, such as contours, peaks, or more elaborate features, in isolated image frames. This is extremely important, especially in noisy domains where spatial features are difficult to obtain reliably. As well, the use of these filters is not premised necessarily on a restricted class of motion. Rather, they make explicit the distribution of local image motion in a form that will facilitate early levels of interpretation such as preliminary segmentation and the determination of a wide class of 3-d motions. Adelson and Bergen (1986) showed that energy extracted from Gabor filters can be used to solve for translational image motion in a way equivalent to gradient-based approaches. This is not surprising since, the Fourier transform of the basic motion constraint equation is

$$\mathcal{F}[I_x v_x + I_y v_y + I_t] \;=\; v_x(ik_x)\hat{I} + v_y(ik_y)\hat{I} + (i\omega)\hat{I} \;. \tag{15}$$

Then, setting the motion constraint equation to zero, and rearranging the transform (which then also equals zero), we obtain the equation for a plane in frequency space as in (3), i.e.

$\vec{k} \cdot \vec{v} + \omega = 0$. Heeger (1987) solved for translational image motion using a least squares fit of local spatiotemporal energy to a plane through the origin in frequency space.

These approaches therefore showed that with the use of such filter outputs we can solve for translational motion as is done in gradient approaches. They also show further promise. For example, based on the conditioning and residual error in a least squares approach, one could determine whether or not i) the local information is largely one dimensional in which case there is not sufficient information to determine 2-D velocity robustly, or ii) there is too much information suggesting that the local motion is too complex to be accounted for simply by 2-d image plane translation. It is of interest to study other types of motion. For example, local spatiotemporal energy has been used to determine properties of turbulent motion [Heeger and Pentland, 1986]. It would also be of interest to consider *i)* semi-transparent surfaces since more than one velocity can be detected at a single location, *ii)* the use of phase information which is discarded in energy-based schemes, and *iii)* an analysis of occlusion.

3 Implementation

This section differs markedly from Section 2 in that attention is shifted primarily to the implementation of families of spatial and spatiotemporal filters. The first sections (3.1 to 3.4) describe functions for the application of Gaussian and Gabor filters, as well as entire families of isotropic Gabor filters. The topics of interest covered include: the generation of the convolution masks with the appropriate bandwidths, frequencies, etc.; the reduced sampling rates and the expected aliasing error caused by sub-sampling; the separability of isotropic Gabors for efficient computation; the use of phase symmetries in families of Gabors yielding a more efficient implementation; and the automatic derivation of channel properties within the families based on bandwidth.

Section 3 is organized as a reference manual with the following issues discussed as they become relevant. Functions for the generation of convolution masks are discussed in Section 3.1. Section 3.2 describes functions that perform Gaussian smoothing in 2-d and 3-d. Section 3.3 deals with 2-d and 3-d Gabor filters as well as several auxiliary functions related to Gabors. Finally, Section 3.4 describes functions that apply entire collections of Gabor filters to an image or image-sequence in order to represent one scale-specific band of the frequency domain. These functions are efficiently implemented and can be applied to provide a complete image encoding.

Details specific to the implementation, but not to filter properties or families, are contained in the Appendix. There I describe the basic convolution software that is used extensively by the functions disucssed in Section 3, and a variety of general utility functions. Furthermore, as described briefly, many of the significant functions can be accessed through a menu when in the top level of the IMAGE-CALC listener.

3.1 Generation of Kernels

This section describes the generation of Gaussian and Gabor convolution kernels (or masks). The masks can be generated for 1-d, 2-d and 3-d convolutions, and are derived in a general fashion.

make-1d-gauss *st-dev &key (trunc-pt 2.0) mask-radius*

This function returns an array containing the coefficients of a discrete approximation to a Gaussian kernel, i.e.,

$$G(x;\sigma) = \frac{1}{\sqrt{2\pi}\sigma} e^{-x^2/2\sigma^2} \quad . \tag{16}$$

The coefficients are scaled so that they sum to one. Therefore, their amplitude spectra have a maximum value of 1.0.

The effective kernel width is, by default, specified in terms of the Gaussian's standard deviation σ, given in pixels. The mask coefficients are then obtained by sampling the appropriate Gaussian at regular intervals, although integrating the area under the Gaussian between sampling intervals might be more appropriate.

The width of the sampling function w with respect to the Gaussian (i.e., how much of the Gaussian envelope gets sampled) is given in terms of σ by the key-parameter *trunc-pt*, which denotes truncation point. This determines the radius of the mask, $\rho = \lceil \sigma w \rceil$. The mask size (the number of coefficients) is then simply $2\rho + 1$. The default value for the truncation point w is 2.0 (in standard deviations, σ). This represents more than 95% of the area under the Gaussian.

In addition, note that the *mask-radius* (in pixels) may be entered directly with a key-parameter thus over-riding the use of the *truncation-point* altogether.

make-1d-gabor-parts *wavelength &key stdev mask-radius frequency (trunc-pt 2.0) (bandwidth 1.0) (extent-measure 1.0)*

This function generates sine and cosine Gabor masks with peak sensitivity at the *wavelength* λ specified in pixels, i.e.,

$$\sin(k_0 x)G(x;\sigma) \ , \qquad \cos(k_0 x)G(x;\sigma) \ . \tag{17}$$

Let β be the *bandwidth* specified in octaves, and let the peak radian frequency be $k_0 = 2\pi/\lambda$. In order to take bandwidth into account we need a measure of extent in the frequency domain, that is of amplitude spectra. Therefore, let μ be the extent of the amplitude spectrum measured in terms of $\sigma_k = 1/\sigma$, the standard deviation of the Gaussian in (radian) frequency space. This value may be specified through the key-parameter *extent-measure*. The default value for μ is 1.0 thereby measuring Gaussian extent at one standard deviation. In order to measure the spectrum at half height instead, μ could be set to $\sqrt{\ln(4)} \approx 1.177$.

Given β, k_0, and μ, **make-1d-gabor-parts** can determine the appropriate standard deviation σ for the Gaussian envelope of the kernel. More precisely, the standard deviation of the frequency domain Gaussian, $\sigma_k = 1/\sigma$, should satisfy the following equality to achieve the desired bandwidth:

$$b = \frac{k_0 + \mu\sigma_k}{k_0 - \mu\sigma_k} \tag{18}$$

where $b = 2^{\beta}$ is the relative bandwidth. Rearranging these terms gives

$$\sigma = \frac{\mu\lambda(b+1)}{2\pi(b-1)} \tag{19}$$

With these parameters a check is done to ensure that the main region of amplitude sensitivity falls below the Nyquist rate, i.e., $k_0 + \mu\sigma_k < \pi$. This indicates whether or not the desired fundamental frequency is too high for the desired bandwidth, and if it is a warning message is issued. In general, the wavelength should satisfy $\lambda < 4b/(b+1)$. Then, with truncation-point, the kernel radius ρ can be derived as described above.

Finally, given σ and ρ the appropriate Gaussian is generated (normalized to one), and is multiplied by both a sine and a cosine function of frequency k_0 thus yielding the two desired Gabor masks. They are returned using a lisp *values* form.

Note that the standard deviation and the mask radius may also be prespecified as key-parameters. Similarly, although it is most common to specify the fundamental frequency implicitly using the *wavelength*, it is also possible to use nil for the *wavelength* parameter and to specify a *frequency* using the appropriate key-parameter. This allows the specification of zero frequency, or the use of frequency when wavelength is only implicit. In any case if the frequency is zero (zero wavelength is also interpreted as zero frequency) then σ must be supplied as a key-parameter. In such cases a Gaussian with the given standard deviation is returned as the cosine part, and nil is returned as the sine part. These options are useful when **make-1d-gabor-parts** is used by its 2-d and 3-d counterparts.

make-2d-gabor-parts *wavelength orientation &key key-parameters*

The two-dimensional case is very similar to the one-dimensional case above. The cosine and sine Gabor kernels are given by

$$\sin(\vec{k}_0 \cdot \vec{x})G(\vec{x};\sigma)\ , \qquad \cos(\vec{k}_0 \cdot \vec{x})G(\vec{x};\sigma)\ , \tag{20}$$

where $G(\vec{x};\sigma)$ is a 2-d isotropic Gaussian as in (10) but with $C_{xt} = \sigma I$. The parameters of interest are the spatial frequency $\vec{k}_0$, the *orientation* θ (specified in degrees), and the standard deviation σ.

Convolution with such masks can be performed in 1-d separable stages by exploiting the following identities:

$$\begin{aligned} \sin(\vec{k} \cdot \vec{x}) &= \sin(k_x x)\cos(k_y y) + \cos(k_x x)\sin(k_y y), \\ \cos(\vec{k} \cdot \vec{x}) &= \cos(k_x x)\cos(k_y y) - \sin(k_x x)\sin(k_y y). \end{aligned} \tag{21}$$

Given this separation, in conjunction with the separability of the Gaussian, the convolution can be performed by first applying 1-d sine and cosine Gabor filters with the appropriate frequency and standard deviation in the x-dimension, followed by the complementary sine and cosine Gabor filters in the y-dimension. Therefore, we generate the 1-d Gabor parts which, when combined, produce the 2-d Gabors.

The required inputs to the function are an *orientation* θ, and a fundamental *wavelength* λ. Here, $\vec{k}_0 = \vec{n}\ 2\pi/\lambda$ where $\vec{n} = (-\sin\theta, \cos\theta)$ is normal to the orientation. That is

to say, scaling the wavelength by sine and cosine of the orientation gives the individual wavelengths along the x and y axes,

$$\lambda_x = \frac{-\lambda}{\sin(\theta)}, \qquad \lambda_y = \frac{\lambda}{\cos(\theta)}. \tag{22}$$

These projected wavelengths can be used as input to **make-1d-gabor-parts**. Note however that the appropriate standard deviation for these calls to **make-1d-gabor-parts** must be supplied by the 2-d function based on the 2-d fundamental wavelength λ, *not* the projected wavelengths. Therefore, using λ, σ is determined as in the 1-d case (19). (To allow similar flexibility the key-parameters *bandwidth*, *extent-measure*, *trunc-pt*, *stdev*, and *mask-radius* can all be specified with the same defaults as in the 1-d case.) A frequency of zero, corresponding to 2-d Gaussian blurring, is prohibited.

Note that if θ is aligned with either of the x or y axes (i.e. an integer multiple of 90 degrees), one of the new projected frequencies will be zero. For that dimension **make-1d-gabor-parts** will return a Gaussian (of the appropriate standard deviation) for the cosine part, and nil is returned for the sine part. The four Gabor parts are returned using a *values* expression with the sine and cosine parts along the x-axis followed by those along the y-axis. The computed standard deviation is returned as the fifth entry in the *values* form so that calling routines may determine an appropriate reduced sampling rate for efficient coding (described further in Section 3.2 and 3.3 below).

make-3d-gabor-parts *spatial-wavelength orientation speed &key key-parameters*

The three-dimensional case is only slightly more complicated than the two-dimensional case. As above, the main parameters are the spatial *orientation* θ and the desired *spatial-wavelength* λ_s to which the filter should be maximally tuned. In addition the desired *speed* v is also specified. Both the temporal wavelength λ_t and the 3-d fundamental wavelength λ are, however, left implicit. The fundamental wavelength is necessary to compute the appropriate standard deviation (as described above), and the temporal wavelength is necessary to compute the 1-d temporal Gabor masks.

Temporal frequency, spatial frequency, and 1-d velocity satisfy $v = -\omega_0/k_s$. Consequently, temporal wavelength is related to the spatial wavelength and velocity as

$$\lambda_t = \frac{-\lambda_s}{v}, \tag{23}$$

with temporal frequency $\omega_0 = 2\pi/\lambda_t$. The L_2 magnitude of (ω_0, k_s) gives the fundamental (radian) frequency which then yields a fundamental wavelength (in pixels), i.e.,

$$\lambda = \frac{\lambda_s}{\sqrt{v^2+1}}. \tag{24}$$

Consequently, given the fundamental wavelength (and the same *key-parameters* and defaults as in the 1-d and 2-d cases) the appropriate standard deviation σ can be computed. This is then used, with the temporal wavelength to get the temporal 1-d Gabor components of the mask (using **make-1d-gabor-parts**). The spatial Gabor parts are obtained by calling **make-2d-gabor-parts** again with σ, plus θ and λ_s.

The 6 Gabor components are returned through a *values* statement with the sine and cosine temporal parts first, followed by the spatial parts as returned by the 2-d function. Like the 2-d case the standard deviation of the Gaussian envelope is also returned as the last argument.

Note that the sine and cosine Gabor kernels can be treated together as a linear FIR filter with complex coefficients (as in (8)), or individually as FIR filters with real coefficients. Individually, their Fourier transforms consist of two Gaussians centred at $(+\vec{k}_0, +\omega_0)$ and $(-\vec{k}_0, -\omega_0)$. For example the Fourier transform of the cosine Gabor is given by

$$\mathcal{F}[\cos(\vec{x}\cdot\vec{k}_0 + t\omega_0)\, G(\vec{x}, t; C_{xt})] = \tag{25}$$

$$\frac{1}{2}[\hat{G}(\vec{k}+\vec{k}_0, \omega+\omega_0;\ C_{xt}) + \hat{G}(\vec{k}-\vec{k}_0, \omega-\omega_0;\ C_{xt})]\ . \tag{26}$$

For sine Gabors these two Gaussians are 180 degrees out of phase, and for cosine Gabors they are in phase. Because of this phase dependence the sine Gabor is strictly band-pass; the frequency domain Gaussians cancel at the origin. For the cosine Gabors however, they sum at the origin yielding some non-zero response to the mean intensity level (or DC component).

To compute this DC sensitivity one must take into account the octave bandwidth and the measure of amplitude spectrum extent. More precisely, the amplitude spectrum at the origin is given by

$$A(\vec{k}=0) = e^{-\frac{\mu^2}{2}\left(\frac{b+1}{b-1}\right)^2}\ . \tag{27}$$

For example, for a one octave filter with extent measured at one standard deviation σ_k (in frequency space), the origin is three standard deviations from the Gaussian centres, and $A(0) = e^{-4.5} \approx 0.011$. Typically this is negligible. If not, it may be subtracted from the coefficient values obtained above giving

$$[\cos(\vec{k}_0\cdot\vec{x} + \omega_0 t) - \frac{A(0)}{2}]\ G(\vec{x}, t; \sigma)\ . \tag{28}$$

The result will be band-pass. This was not implemented but could easily be if desired.

3.2 Gaussian Smoothing

The following two functions perform 2-d and 3-d Gaussian blurring. In both, the convolution masks are assumed to be separable. Therefore, the covariance matrix is diagonal. The basic parameters are: an image; and the standard deviations corresponding to the x, y spatial dimensions and time.

Note that the Gaussian is a low-pass filter and therefore, depending on the amount of smoothing, much of the high frequency content of the signal may be significantly attenuated, in which case the resulting signal may be sampled at a coarser rate with minimal amounts of aliasing. Information theory (e.g., [Gabor, 1946], or see Section 2) suggests that for Gaussian windows the sampling interval should be 2σ so that windows just touch at one standard deviation. In frequency space, therefore, the sampling frequency is π/σ and the highest frequency

that can be uniquely represented is $\pi/2\sigma$. The amount of energy aliased in 1-d (assuming flat spectral input) will then be

$$1 - \frac{2}{\sqrt{2\pi}} \int_0^{\pi/2} e^{-k^2} dk \approx 0.02 . \tag{29}$$

The aliasing is close to 0.04 in 2-d and about 0.06 in 3-d. Therefore results of such re-sampling with respect to aliasing are not very significant.

The key-parameters in both functions allow for automatic reduction of the spatial sampling rate of the convolution output. With automatic reduction the mask is only applied on the new sampling lattice. These functions were written mainly for use with the menu hierarchy. Parameters governing fixnum calculations, precision, and possible sample-spacing are not available at present, but could be.

gaussian-2d *image stdev-x stdev-y &key (reduct-x t) (reduct-y t) into-image*

gaussian-3d *image-sequence frame-num stdev-x stdev-y stdev-t &key (reduct-x t) (reduct-y t) into-image*

In fact this routine does not output an entire convolution result, rather it outputs the result computed only at one frame. In other words, it computes a weighted linear combination of frames about the frame number specified (0 is the first frame). In order to get a time-varying convolution output this function could be applied repeatedly.

3.3 Gabor Filters

The goal here is to exploit the Gabor masks generated by the functions in Section 3.1 to obtain outputs from Gabor convolutions.

The amplitude spectra associated with the sine and cosine Gabor kernels individually are pairs of Gaussian envelopes centred at frequencies determined by the sinusoidal component (25). In linear combination, the amplitude spectra become a single Gaussian for positive frequencies, or a single Gaussian for negative frequencies (cf. Sections 2 and 3.1). In such cases sinusoidal modulation can be used to move the Gaussian back to the origin so that its output may be sampled more coarsely (at the same rate as the output of a Gaussian low-pass filter of the same standard deviation). More precisely,

$$e^{i(\vec{x}\cdot\vec{k}_0 + t\omega_0)} \, [Gabor(\vec{x}, t;\ \vec{k}_0, \omega_0, \sigma) * I(\vec{x}, t)] \tag{30}$$

has the same effective transfer function as a Gaussian smoothing function, since, using the convolution theorem, the Fourier transform of (29) is given by

$$\delta(\vec{k} - \vec{k}_0, \omega - \omega_0) * [\hat{Gabor}(\vec{k}, \omega;\ \vec{k}_0, \omega_0, \sigma)\ \hat{I}(\vec{k}, \omega)] . \tag{31}$$

Using (11) this is equivalent to

$$\hat{G}(\vec{k}, \omega; \sigma)\ \hat{I}(\vec{k} + \vec{k}_0, \omega + \omega_0) . \tag{32}$$

In other words, rather than modulate the Gaussian envelope to create the Gabor kernel, we could modulate the image and then smooth the result with the Gaussian, the output of which

may be sampled at a lower rate. In practice, the modulation with a complex exponential in (29) need not be applied, and both the cosine and sine parts should be re-sampled at the same rate. In such cases the amount of expected error is bounded by (28) (the energy aliased in the Gaussian case). The modulation must be taken into account, however, if reconstruction is considered.

In order to introduce the same overlap of the tiling in frequency space (of amplitude spectra) as with the windows in space (the kernel widths) the Gaussians should just touch at one standard deviation (see Section 2). This accounts for the choice of reduced sampling rates with the reduction factor computed as $\max(1, \lfloor 2\sigma \rfloor)$.

gabor-sin-cos-2d *image wavelength orientation &key stdev mask-radius (bandwidth 1.0) just-cos just-sin (auto-reduction t) cos-output sin-output*

As noted above, Gabor kernels with isotropic Gaussian components are separable into horizontal and vertical convolutions. Each sine or cosine Gabor output can be computed as the linear combinations to two 2-d separable convolutions (see Eq. (21)). This requires four 1-d convolutions for each, that is, eight in total. It is more efficient, however, to consider the computation of the sine and cosine outputs as a single operation. In this case they may share the same first 1-d stage of processing. The sine and cosine 1-d convolutions in the horizontal direction are common to both and may be followed by four vertical 1-d convolutions; the vertical sine and cosine 1-d Gabors are cascaded with each of the horizontal outputs. Finally, the two Gabor outputs are simply linear combinations of the four outputs of the second stage, and only six 1-d convolutions are necessary.

The basic function **gabor-sin-cos-2d** first uses **make-2d-gabor-parts** to generate the necessary 1-d masks. It then calls **gabor-2d-guts** to coordinate the application of the filters to the image in an efficient manner. A check is done to find cases in which the specified *orientation* is either vertical or horizontal (i.e., an integer multiple of 90 degrees). In such cases only three 1-d convolutions are necessary since in Eqs. (20) either k_x or k_y will be zero. As mentioned in conjunction with **make-2d-gabor-parts**, one of the 1-d sine Gabor parts will be nil, and the corresponding cosine part is a Gaussian.

In addition to the cascaded structure with shared intermediate results, I have included some control to allow for cases where the user wants only the sine or cosine output. These options can be specified by setting the appropriate key-parameter, *just-cos* or *just-sin*, to true. The function returns the sine and cosine Gabor outputs through a *values* form.

gabor-sin-cos-3d *image-seq frame-num spat-wavelength orientation speed &key key-parameters*

This function is very similar to its 2-d counterpart. The differences are much like those between the functions **make-2d-gabor-parts** and **make-3d-gabor-parts**.

Given the six 1-d Gabor components, the sine and cosine Gabor outputs each require 12 1-d convolutions in the simplest implementation suggested by the separability (e.g., see [Heeger, 1987]), i.e.,

$$\sin(\vec{k} \cdot \vec{x} + \omega t) = \sin(\omega t)\cos(k_x x)\cos(k_y y) - \sin(\omega t)\sin(k_x x)\sin(k_y y)$$

$$+ \cos(\omega t)\sin(k_x x)\cos(k_y y) + \cos(\omega t)\cos(k_x x)\sin(k_y y).$$

(33)

$$\cos(\vec{k}\cdot\vec{x} + \omega t) = \cos(\omega t)\cos(k_x x)\cos(k_y y) - \cos(\omega t)\sin(k_x x)\sin(k_y y) - \sin(\omega t)\sin(k_x x)\cos(k_y y) - \sin(\omega t)\cos(k_x x)\sin(k_y y).$$

Given this separation, in conjunction with the separability of the Gaussian, the convolution can be performed by first applying 1-d sine and cosine Gabor filters (with the appropriate frequency and standard deviation) in time, followed by those in the x-dimension, followed by those in the y-direction.

Applying the sine and cosine filters together, making use of shared intermediate results, requires only 14 1-d convolutions in total. Thus, only two temporal 1-d convolutions are required. These are computed first as they are the most expensive. Then the two x-direction masks are applied to each of the temporal outputs, and finally, the sine and cosine Gabors masks for the y-direction are applied to each of the four x-direction outputs. The actual 3-d sine and cosine Gabor outputs are formed by simple linear combinations of these 8 separable outputs (32).

Note that, as discussed in terms of the temporal convolution function, **gabor-sin-cos-3d** computes a result only at one frame in the sequence. To obtain a time-varying output use the function **gabor-sequence** as described below.

gabor-power *sin-part cos-part &key into-image*

Given the cosine and sine Gabors output, this function computes a local estimation of power. It simply calls the utility function **complex-image-power**. There is also a utility function to compute local amplitude called **complex-image-amplitude**.

gabor-phase *sin-part cos-part &key into-image*

Given the cosine and sine Gabor outputs G_c and G_s this function computes local phase as $\arctan(G_s/G_c)$ using the utility function **complex-image-phase**. The output phase values fall between $-\pi$ and π. In manipulating phase spectra further, a utility function is provided to take an image modulo 2π so that its phase values also fall between $-\pi$ and π.

gabor-sequence *image-seq nframes start-frame spat-wavelen orientation speed &key (bandwidth 1.0) (auto-reduction t)*

This function simply applies a 3-d Gabor filter (sine and cosine parts with the parameters specified as described above) to the frames specified. The results are returned in an array of length (2 *nframes*), with sine and cosine Gabor outputs for frame i + *nframes* returned as *array[2i]* and *array[2i+1]*.

gabor-power-sequence *image-seq nframes start-frame spat-wavelen orientation speed &key (bandwidth 1.0) (auto-reduction t)*

Like **gabor-sequence** this function applies a gabor filter to an image-sequence. However, it returns an array of length *nframes* containing not the sine and cosine results, but rather

the power computed using **gabor-power**. (In fact, before returning the power output, each result frame is placed through a display function to scale the results to 8 bits. So this is mainly for viewing results or for demonstration.)

3.4 Gabor Families

As described in Section 2 we are ultimately interested in families of filters providing rich image decompositions. Using the Gabor functions described above we now consider the construction of an entire family. Of special interest is the tremendous efficiency obtained by exploiting phase relationships between filters within the family and intermediate results (a hierarchical approach). The implementation of the family can be much more efficient per filter than that outlined in Section 3.3.

For each of the families considered here frequency space is divided into a set of scale-specific (isotropic) rings, each of which is divided into a set of orientation or velocity tuned channels. All the functions described below do *not* compute a complete representation across all scales. Instead, given a fundamental wavelength of interest, they compute the outputs of the set of filters that decompose the annular region of frequency space about the desired scale (a spheroid in 3-d). Given the desired bandwidth the function determines the appropriate number of orientation channels and their tuning (from Section 2). It then creates the filter masks and applies them to the image.

The application of filters comprising different scales, either with constant bandwidth or constant spatial support (varying bandwidth), can be handled by another function. From the discussion above (or the program diagnostic output) one can determine the high and low frequency cut-off for a given ring, and thereby decide on the placement of neighbouring rings.

spatial-gabor-ring *image wavelength &key (auto-reduction t) (bandwidth 1.0)*

Given the *bandwidth* β, we can determine the appropriate number of orientation channels. This is given by (see Figure 1 in Section 2.3)

$$N = \lceil \pi/\theta \rceil , \quad \text{where} \quad \theta = 2 \arcsin\left(\frac{R}{f_0}\right) . \tag{34}$$

Here, R is the radius of the amplitude spectrum ($R = \mu\sigma_k$ from Section 3.1), and f_0 is the peak frequency ($f_0 = 2\pi/\lambda$ from Section 3.1). I use the ceiling instead of the floor as in Section 2 because it may be more important to stress completeness than efficiency at this stage. By approximating θ by $2R/f_0$ we get

$$N \approx \lceil \frac{\pi^2\sigma}{\mu\lambda} \rceil = \lceil \frac{\pi(1+2^\beta)}{2(2^\beta-1)} \rceil . \tag{35}$$

Thus the preferred orientation of the n^{th} channel is simply $\theta_n = n\pi/N$. This then yields the parameters necessary for the generation of the masks.

Towards an efficient implementation of the convolutions we may take advantage of certain phase relationships. In particular, note that for $n \neq 0$ or 90 degrees, $\theta_n = 180 - \theta_{N-n}$.

Consequently, the wavelengths in the x and y dimensions will have the same magnitudes for both channels but will differ in sign; i.e.,

$$\lambda_{x,n} = \lambda_{x,N-n} \ , \qquad \lambda_{y,n} = -\lambda_{y,N-n} \ . \tag{36}$$

Consequently, the intermediate 1-d separable results necessary to compute the n^{th} orientation are identical (modulo sign change) to those for the $(N-n)^{th}$ orientation. More precisely,

$$\begin{aligned} \sin(\vec{k}_n \cdot \vec{x}) &= \sin(k_{x,n}x)\cos(k_{y,n}y) + \cos(k_{x,n}x)\sin(k_{y,n}y), \\ \cos(\vec{k}_n \cdot \vec{x}) &= \cos(k_{x,n}x)\cos(k_{y,n}y) - \sin(k_{x,n}x)\sin(k_{y,n}y). \end{aligned} \tag{37}$$

$$\begin{aligned} \sin(\vec{k}_{N-n} \cdot \vec{x}) &= \sin(k_{x,n}x)\cos(k_{y,n}y) - \cos(k_{x,n}x)\sin(k_{y,n}y), \\ \cos(\vec{k}_{N-n} \cdot \vec{x}) &= \cos(k_{x,n}x)\cos(k_{y,n}y) + \sin(k_{x,n}x)\sin(k_{y,n}y). \end{aligned} \tag{38}$$

Therefore, we need only the masks for θ_n up to and including 90 degrees, and the 6 1-d separable convolutions described in Section 3.3 for **gabor-sin-cos-2d** can be used to compute not 2, but 4 Gabor outputs at orientations θ_n and θ_{N-n}. This yields a considerable increase in efficiency. Furthermore, since all filters have the same Gaussian envelope, they should all have the same reduced sampling rates.

The function returns an array containing the convolution outputs. The sine and cosine Gabor outpus for orientation θ_n are found in *array[2n]* and *array[2n+1]*. The size of the array depends on the number of orientations which depends on the bandwidth.

space-time-gabor-ring *image-seq frame-num wavelen orientation speed &key (auto-reduction t)*

The decomposition of the spatiotemporal scale-specific frequency space region is more involved than the 2-d case. Rather than consider the general tiling problem of a spheroidal ring, consider the tiling in terms of a set of toroidal rings parallel to the spatial frequency axis, with the radius of the ring decreasing with higher temporal frequencies. Thus each ring represents a region of narrow speed tuning, but no orientation specificity. We can then use the scheme above to determine how many orientation should be in each ring. The middle ring, which contains the plane at $\omega = 0$, will contain the greatest number of orientation channels, each with a preferred speed tuning of $\vec{v}_n = 0$. Above and below come rings with somewhat fewer orientation channels that are sensitive to non-zero yet small velocities. Note that for rings with tuning to higher temporal frequencies, the principal spatial frequency tuning decreases. and Yet for all rings the bandwidth remains constant. Finally, decomposition should end nicely with one filter sensitive to high speeds and flicker.

The number of these speed-rings depends on the bandwidth; remember that all the filters have 3-d isotropic amplitude spectra. Thus, the difference in preferred orientation tuning between adjacent spectra for the middle band (zero speed case) $\Delta\theta$ determines the speed preferences for other rings. More precisely, the preferred speeds for the two rings of distance n from the middle ring are $\pm \tan(n\Delta\theta)$. Ideally the bandwidth should be set so

that for $\Delta\theta_0$ there is some n satisfying $n\Delta\theta = 90$ degrees. In this way the flicker channel is obtained in the final ring.

One appropriate bandwidth is 0.8 octaves, which is what **space-time-gabor-ring** uses. In this case there are effectively 6 toroidal rings. One tuned to zero speeds with 6 different orientations, two tuned to speeds of $\pm 1/\sqrt{3}$ (in pixels/frame) each with 5 orientations, two tuned to speeds of $\pm\sqrt{3}$ each with 3 orientations, and the flicker channel. For a given orientation the directions that correspond to positive and negative speeds can be thought of as follows: Imagine a range of orientations to which a filter is tuned, say from 30 to 60 degrees, that lie on the perimeter of a circle. Positive speeds correspond to the direction out from the centre of the circle, while negative speeds point inwards.

Like the orientation case above phase symmetries are exploited. This applies not only within each speed-ring but for rings which differ only in the sign of the speed tuning. Every two such rings share the same intermediate results, and can be combined in different linear combinations. In this case the 46 sine and cosine Gabor outputs are computed with only 75 1-d convolutions. This is over 7 times more efficient than if each had been computed independently. This is in addition to the efficiency already gained by 3-d separability.

It is useful to note that several intermediate functions were written that coordinate most of the shared intermediate results, the phase relationships, and the order in which convolutions are applied. These are **full-family-guts**, **part-family-guts**, **full-reflected-gabors**, and **part-reflected-gabors**. The former two operate in 2-d space exploiting spatial phase symmetries. Of these the second handles cases where the orientation of interest is an integer multiple of 90 degrees, in which case there are no phase-symmetries of interest and only 3 1-d convolutions are necessary. These two functions are used explicitly by **spatial-gabor-ring** for example. The latter two functions work in space-time. As above, the second function deals only with cases in which the preferred orientation is an integer multiple of 90 degrees.

4 Comments and Extensions

As noted at the end of Section 2.4 there remain many issues still to be resolved. With respect to immediate goals I see two topics for future work. The first is an extension of the current implementation to include filters with non-unit aspect ratios. This may follow the hierarchical approach outlined in [Fleet and Jepson, 1988] which should be as efficient as the functions in Section 3.4. (The necessary functions for this would be 1-d inseparable convolution, which should not be difficult.) This representational scheme should have several advantages in terms of localization, directional tuning and efficient encoding.

The second immediate topic for future research is the accurate measurement of orientation and normal velocity when given the filter outputs. The filters described above are linear and therefore yield a modulated output. It is of interest to consider how estimates of velocity and orientation may be derived from the filter outputs, that are 1) monotonic with increasing speed and orientation, and 2) more accurate than the tuning of the original filters. This stage of processing will be non-linear (e.g., see [Heeger, 1987]).

Acknowledgements

Much of this work stems from collaboration with Prof. A. Jepson and Michael Langer at the University of Toronto. I am also endebted to Prof. Bernd Neumann and Michael Mohnhaupt of the University of Hamburg for many discussions and hospitality in Hamburg. In addition, Gregory Dudek and David Rishikof gave useful comments on earlier drafts, and Roman Cuntis helped supplement my sparse knowledge of lisp. Financial support was provided in part by the German Science Foundation.

5 References

Adelson, E.H. and Bergen, J.R. (1985) Spatiotemporal energy models for the perception of motion. *J. Opt. Soc. Am. A* 2, pp. 284-299

Adelson, E.H. and Bergen, J.R. (1986) The extraction of spatiotemporal energy in human and machine vision. *IEEE Workshop on Motion*, Charleston, S.C. USA, pp. 151-156

Barnard, S.T. and Thompson, W.B (1980) Disparity analysis of images. *IEEE Trans. PAMI* 2, pp. 333-340

Binford, T.O. (1983) Figure/ground: Segmentation and aggregation. In **Physical and Biological Processing of Images**, (eds.) Braddick, O. and Sleigh, A., Spinger-Verlag, New York, pp. 304-315

Bracewell, R.N. (1978) The Fourier Transform and its Aplications. 2nd edition, McGraw Hill, New York

Burt, P.J., and Adelson, E.H. (1983) The Laplacian pyramid as a compact image code. *IEEE Trans. on Communications* 31, pp. 532-540

Crowley, J.L. and Parker, A. (1984) A representation for shape based on peaks and ridges of the difference of low-pass transform. *IEEE Trans. PAMI* 6, pp. 156-169

Crowley, J.L. and Stern, R.M. (1984) Fast computation of the difference of low-pass transform. *IEEE Trans. PAMI* 6, pp. 212-222

Daugman, J.G. (1985) Uncertainty relation for resolution in space, spatial frequency, and orientation optimized by two-dimensional visual cortical filters. *J. Opt. Soc. Am. A* 2, pp. 1160-1169.

Dreschler, L.H. and Nagel, H.H. (1982) Volumetric model and 3D trajectory of a moving car derived from monocular TV frames of a street scene. *Comput. Graphics Image Process* 20, pp. 199-228

Field, D.J. (1987) Relations between the statistics of natural images and the response properties of cortical cells. *J. Opt. Soc. Am. A* 4, pp. 2379-2393

Fleet, D.J. and Jepson, A.D. (1984) A cascaded filter approach to the construction of velocity selective mechanisms. Technical Report: RBCV-TR-84-6, (available from Dept. Computer Science, U. of Toronto)

Fleet, D.J. and Jepson, A.D. (1985a) Spatiotemporal inseparability in early vision: centre-surround models and velocity selectivity. *Computational Intelligence* 1, pp. 89-102

Fleet, D.J. and Jepson, A.D. (1985b) Velocity Extraction Without Form Interpretation. *Proc. IEEE Workshop on Computer Vision: Representation and Control*, Michigan, pp. 179-185

Fleet, D.J. and Jepson, A.D. (1988) Hierarchical construction of orientation and velocity selective filters. *IEEE Trans. PAMI* to appear. (also see Proc. SPIE 594 (Image Coding), pp. 10-20, 1985; and Technical Report: RBCV-TR-85-8)

Gabor, D. (1946) Theory of communication. *J. IEE* 93, pp. 429-457

Gallager, R. (1968) Information Thoery and Reliable Communication. Wiley and Sons, New York.

Geisler, W.S. and Hamilton, D.B. (1986) Sampling-theory analysis of spatial vision. *J. Opt. Soc. Am. A* 3, pp. 62-69

Heeger, D.J. (1987) Model for the extraction of image flow. *J. Opt. Soc. Am. A* 4, pp 1455-1471

Heeger, D.J. and Pentland, A.P. (1986) Seeing structure through chaos. *Proc. IEEE Workshop on Motion: Representation and Analysis*, Charleston, pp. 131-136

Horn, B.K.P. (1986) Robot Vision, MIT Press, Cambridge

Horn, B.K.P. and Schunck, B.G. (1981) Determining optic flow. *Artificial Intelligence* 17, pp. 185-204

Koenderink, J.J. (1984) The structure of images. *Biological Cybernetics* 50, pp. 363-370

Langer, I.M.S. (1988) On efficient representation of natural images. M.Sc. Thesis, Dept. of Computer Science, University of Toronto

Nagel, H.H. (1983) Displacement vectors derived from second-order intensity variations in image sequences. *Comput. Graphics and Image Proc.* 21, pp. 85-117

Sakitt, B. and Barlow, H.R. (1981) A model for the economical encoding of the visual image in cerebral cortex. *Biological Cybernetics* 43, pp. 97-108

Schunck, B.G. (1985) Image flow: fundamentals and future research. *Proc. CVPR*, San Francisco, June, 1985, pp. 561-571

Shannon, C.E. (1949) Mathematical theory of communication. University of Illinois Press, Chicago.

Slepian, D. (1983) Some comments on Fourier analysis, uncertainty and modelling. *Siam Review* 25, pp. 379-393

Watson, A.B. (1983) Detection and recognition of simple spatial forms. In **Physical and Biological Processing of Images**, (eds.) Braddick, O. and Sleigh, A., Spinger-Verlag, New York

Watson, A.B. and Ahumada, A.J. (1985) Model of human visual-motion sensing. *J. Opt. Soc. Am. A* 2, pp. 322-342

6 Appendix: Implementation Details

For the current implementation (unlike IMAGE-CALC) I assume that images, by default, consist of 32-bit 2's complement fixed-point numbers. Mask (or kernel) coefficients are generally assumed to be floating-point numbers between minus one and one. This need not always be the case as discussed below. Like IMAGE-CALC, one convention often used with the functions described below is the specification of a particular image into which the output should be placed (often through the key-parameter *into-image*). If such key-parameters are *not* used then a new image is created to hold the result, which by default has 32-bit fixnum pixels that are initialized to 0. Different pixel types (e.g. floating-point or 8-bit fixnum), and different initial values may be specified in general by providing such "result" images directly using key-parameters.

Most image manipulation functions return an image as a result (independent of whether the user specifies an image into which the result should be placed). Often, however, functions return several results. This is true of mask generation functions as well as image operators. In doing so, the lisp *values* form is often used to return the function outputs. Therefore, such functions should be called within a *multiple-value* or a *multiple-value-bind* expression.

6.1 Convolution Functions

6.1.1 Basic Convolution Functions

The following functions form the basis of the more sophisticated convolution routines. They are written assuming that they will be called primarily by other routines, and not used directly at the top level. Therefore, most of their parameters must be specified directly, rather than being left as optional or key-parameters.

In all but **convolve-image-sequence** there is an optional parameter which, if specified should be an image into which the output is placed. In **convolve-image-sequence** this is done through the key parameter *into-image*. (Perhaps, for consistency, they should all be key parameters.)

convolve-1d-rows *image mask fc pr ss rf &optional into-image*

This is the basis for most the convolution software; it was adapted from a similar, undocumented function found in the IMAGE-CALC file 'gauss-image.lisp'. The principal inputs are an image and a kernel (mask). The kernel is applied to the image in the horizontal direction (along the rows parallel to the x-axis). The result is transposed, ready for another horizontal convolution to complete a 2-d separable operation.

The convolution is performed most efficiently one row at a time. Therefore, the rows are first extracted from the image, after-which, each is extended by placing a border of pixels at each end. This ensures that the mask may be applied up to the actual border of the original image. (The intensities of the border pixels are the same as those along the boundary of the original image to ensure that abrupt intensity changes are not found at image boundaries.) The mask is then applied along the row yielding a 1-d array of output that is then placed into the columns of the result image; this implicitly performs the transposition.

The computation may be done using fixed-point arithmetic with fixed-point output, or floating-point arithmetic with either fixed- or floating-point output. If fixed-point arithmetic is specified (via the boolean parameter *fc* denoting fixnum-calculation), then the number of significant digits for the mask coefficients must be specified (via the parameter *pr* denoting precision).

Also, there is a reduction factor (*rf*) that indicates the placement interval at which the mask is applied to the image, so that a result is not computed at every pixel. This is equivalent to specifying a sampling function that is applied (multiplicatively) to the convolution output. for example, if the reduction factor is 2, the output image (in the horizontal dimension) is half the size. Given the reduction of the effective fold-over rate accompanied by low-pass filtering, this permits a combination of convolution and subsampling for more efficient computation and coding in one step.

Finally, a sample spacing parameter (*ss*) is specified, which, if other than 1, specifies that the filter coefficients are not paired with adjoining pixels, but with pixels spaced by *ss*. Naturally this is normally 1. In a multiple-cascade approach, like that of Burt and Adelson (1983), Crowley and Stern (1984), or Fleet and Jepson (1988), this is often useful.

convolve-1d-rows-symm-mask *image mask fc pr ss rf &optional into-image*

This is largely identical to **convolve-1d-rows** above although it exploits assumed symmetry in the kernel.

convolve-1d-rows-antisymm-mask *image mask fc pr ss rf &optional into-image*

This is largely identical to **convolve-1d-rows** above although it exploits assumed odd-symmetry in the kernel.

convolve-image-sequence *image-seq frame-num mask fc pr ss &key zero-past into-image*

This takes 3 primary inputs: an image-sequence (which is assumed to be an array of images), a frame number, and a mask. The function returns a weighted sum of images from the image-sequence with the weights given by the mask, centred at the frame specified. It amounts to the temporal convolution output computed only at the specified frame. (Therefore, it should probably be called a linear summation routine instead of convolution, where convolution would be a subsequent routine calling this function repeated for different frames.)

The other inputs, namely *fc*, *pr*, and *ss*, are functionally similar to those with the other convolution functions above. Note that there is no reduction factor because the result is only computed at one frame anyway.

If the sequence does not have enough images for the size of the mask specified then the frame-number given was either close to the beginning or close to the end of the sequence. In such cases, by default, the function assumes that all images before time 0 are equal to frame 0 (i.e., it assumes a static past), and that all images beyond the final frame are equal to the final frame (i.e., it assumes a static future). However, the user may also

specify that the past and future pixel values should be zero by setting the boolean key parameter *zero-past* to true.

The main workings of the convolve-image-sequence routine are actually contained in two other functions that are reasonably involved. The goal was to adapt the IMAGE-CALC **image-point-operator** macro to take a linear combination of an arbitrary number of images. Therefore, dynamically (when given the size of mask and therefore the number of images) a new image-point-operator instance is generated, and compiled with the appropriate number of arguments. Therefore each call to **convolve-image-sequence** will be slow the first time with a specific number of images. Subsequently it retains the compiled version on the function's property list for future use.

6.1.2 General Convolutions

convolve-1d *image mask &key row column into-image display-image display-only transpose-result (fixnumcalc t) (precision 5) (sample-spacing 1) (reduction-factor 1)*

This is a friendlier function than the 1-d convolution routines above. With respect to the key-parameters: 1) either *row* or *column* should be specified as true, both not both; 2) If an 8-bit display version of the result is desired as output in addition to the 32-bit result, the appropriate image type should be provided using the *display-image* key-parameter into which the output will be placed; 3) if only the display version is required then *display-only* should be set to true, in which case the function returns the 8-bit image instead of a 32-bit image which is the default (a *display-image* may still be supplied for the output); 4) the option of transposing the result to allow for subsequent completion of a separable 2-d convolution is allowed with the boolean key-parameter *transpose-result*; 5) the last four key-parameters are functionally similar to those described in Section 3.1, but provide default values and therefore need not be specified explicitly.

convolve-2d-separable *image row-mask col-mask &key key-parameters*

This, like **convolve-1d** is a more comfortable routine that enables 2-dimensional separable convolution. Therefore, in addition to the input image, both a row mask and a column mask must be given. Otherwise, its key-parameters are similar to those of **convolve-1d**, except for *row*, *column*, and *transpose-result* of which it has no use.

Thus far, a separable 3-d convolution, and general 2-d and 3-d convolutions have not been written. To do so shouldn't be hard at all. For the general case however, the first step should be the construction of a *bordered* image as described in the 1-d case to allow the mask to be applied up to the border. A function in the utility-fns file already exists to do this as described in Section 3.6.2. The remaining code can be adapted from **convolve-1d-rows**.

6.2 Utility/Menu Functions

6.2.1 Timing and Debugging

These functions (and global variables) have been used to display diagnostic information about the time taken by various functions and intermediate values of various parameters, mask

coefficients, etc. The functions of interest are: **dbg-on, dbg-off, timer-on,** and **timer-off.** They are called with no arguments.

6.2.2 Image Operators

rescale-image *image scale-factor &key into-image*

This function simply rescales all values in the image by the scale-factor.

shift-intensity *image off-set &key into-image*

This function adds the *off-set* to all intensity values in the image. If the input *image* has floating point pixel values so will the result.

set-image-intensity *image new-value*

This acts like an initialization function setting all pixels values of the image to new-value.

bordered-image *image x-size y-size &optional (x-offset 0) (y-offset 0)*

The adds a border to the existing image. The returned image will be larger horizontally by 2 times x-size, and vertically by 2 times y-size. By default the old image is placed in the centre. However, using the key-parameters it may be shifted towards one or two of the bounaries.

make-image-copy *image &key into-image*

This function is used to avoid problems in using IMAGE-CALC's copy-image function with other than 8-bit fixnum images this one was written.

extract-image-window *image x-start y-start &key x-dim y-dim into-image unmake-original*

A window is extracted from *image* and returned as an image. The lower left-hand point of *image* for the window is given by *(x-start , y-start)*. The size of the window can be specified by the key-parameters *x-dim* and *y-dim*, or the size of *into-image*. Otherwise by default it is the remaining portion of *image*. If *unmake-original* is true then the original input image will be discarded and expunged.

image-ipo-add *image-a image-b &key into-image*

Useful image-point-operators were written to perform frequent operations on images with no re-scaling of results as is done by several IMAGE-CALC functions. The two functions below are similar.

image-ipo-sub *image-a image-b &key into-image*

image-ipo-mult *image-a image-b &key into-image*

rotate-image *image orientation &key interpolation*

This function creates a new image from the given image by rotating it counter-clockwise. Orientation is specified in degrees.

6.2.3 Image Generators

make-2d-impulse *&key into-image (intensity 100000)*

This simply returns an image which has been initialized to zero, except for the centre pixel which is set to the value of 'intensity'. A 3-d impulse function can be formed implicitly using a 2-d impulse function, and specifying *zero-past* when calling **convolve-image-sequence**.

make-disk *radius &key x-centre y-centre (image-size 128) (intensity 500)*

This creates an image with a bright disk on a dark background. The radius is required as input. The key-parameters may be used to move the location of the disk from the centre, the size of the output image created, and the intensity of the disk.

make-bar *x-width y-width orientation &key x-centre y-centre (image-size 128) (intensity 500)*

This creates an image with a bright rectangular region on a dark background with the given orientation. The key-parameters may be used to move the location of the bar from the centre, the size of the output image created, and the intensity of the bar.

make-sinusoid *wavelength orientation &key phase (amplitude 250) (image-size 128)*

This function is similar to **make-bar**. The sinusoid is generated in the horizontal direction, and then rotated using the function **rotate-image**.

make-disk-sequence *n-frames disk-radius x0 y0 dx dy*

Returned is an array of images with a disk of radius *disk-radius* translating with speed *(dx, dy)*, centred at an initial location of *(x0, y0)*.

make-translation-sequence *image n-frames dx dy*

Returned is an array of images (all of the same size) with the image translating for *n-frames* with speed *(dx, dy)*.

6.2.4 Mask Functions

check-mask *mask string* This is used often when the debugger is no. It simply displays information about the mask, including the coefficient values, the sum of the coefficient values (and the absolute coef values), and their standard deviation.

rescale-mask *mask scale-factor &key into-mask*

This rescales the mask given by scale-factor. If into-mask is unspecified then a new floating-point array is created into which the result is placed. As well into-mask may be the same as the input mask, thus doing the scaling in place. In all cases the new mask is returned by the function.

make-fixnum-mask *old-mask &optional (precision 6) &key (remove-zeros t)*

This returns a new mask in which the coefficients are all fixnum. The values of the old mask input are scaled to the specified number digits of precision and then rounded to determine the new mask coefficients. If remove-zeros is specified, then the new mask will trailing and leading zeros, thus creating a smaller mask.

6.2.5 Display Functions

make-display-image *image &key assumed-min-max display-image (do-scaling t)*

I had some problems displaying results from 32-bit computations. Therefore, for expedience and consistency, a display routine was written to allow for rescaling of output results into 8-bit images for display purposes. The function returns the scaled copy of the original.

By default, for images containing only positive values, they are scaled from 0 to 255. For images with positive and negative values, the results are scaled to fall within the range -127 to +127. Using the key parameters *assumed-min* and *assumed-max* one can scale a set of outputs according to the same minimum and/or maximum values. In other words specifying *assumed-max* or *assumed-min* overrides the minimum or maximum found from the input image. Also, by setting the *do-scaling* key-parameter to nil, the results will simply be truncated at 0 and 255 with no rescaling.

display-image-sequence *image-array*

There is an undocumented function in IMAGE-CALC to display lists of images. The function was adapted for use by the menu to display image sequences supplied as arrays.

draw-vector-field *orient-image ampl-image max-ampl x0 y0 wx wy vect-radius disp-pane*

This function assumes that *orient-image* is an image containing orientation values (estimates) in degrees. *Ampl-image* is assumed to contain non-zero confidence values. The parameters *x0, y0, wx* and *wy* specify the bottom left coordinates and widths of region (window) of *orient-image* to be displayed. Finally, *vect-radius* gives the room necessary for each vector (therefore the expansion factor), and *disp-pane* must be an image-calc pane (e.g., use **get-imagecalc-pane**). Normally this function is accessed through the menu system.

6.2.6 Input-Output Functions

input-image-sequence *path-name number-frames frame-start-index*

This function, normally accessed through menus, will load a sequence of images and return them in an array of length *number-frames*. The images should be arranged so that the i^{th} image is found with file name "pathname[i+*frame-start-index*].image". In other words, an extension of ".image" is assumed, as is the numbering as a suffice onto the basic file name assumed of all image common to pathname. This function is most easily called using the menu.

output-image-sequence *image-array number-frames pathname &key (array-start-index 0) (frame-start-index 1)*

Much like **input-image-sequence** this places an array of images to disk under the basic pathname specified, an index number, and the extension ".image". The function takes the specified number of images from the array from index *array-start-index* to *array-start-index* + *number-frames* and begins the suffice file numbering with *frame-start-index*.

write-image-to-file *image pathname*

This write the pixels values of the image to the specified file in ascii format so that it is readable as a text file.

6.2.7 Menu Hierarchy

After invoking IMAGE-CALC the middle button provides access to a variety of the function described here through menu control. In this section I simply list those functions available in the menu hierarchy. The top level menu (when middle button hit) allows access to four subordinate menus:

Gaussians Gabors I/O Display

These four menus contain the following functions:

- Gaussian Menu

  2-D Gaussian Filter

  3-D Gaussian Filter

  2-D Impulse Response

- Gabor Menu

  2-D Gabor Filter

  2-D Impulse Response

  3-D Gabor Filter

  3-D Impulse Response

  Gabor Power

- I/O Menu

  Input Image

  Input Sequence

  Output Sequence

- Display Menu

  Display Image

  Display Sequence

  Display Vector Field