Contents

1 Temporal filters

Although adding time might seem like a trivial extension from 2D signals to 3D signals, and in many aspects it is, there are some properties of how the world behaves that make sequences to be different from arbitrary 3D signals. In 2D images most objects are bounded occupying compact and well defined image regions. However, in sequences, objects do not appear and disappear instantaneously unless they get occluded behind other objects or enter or exit the scene through doors or the image boundaries. So, the behavior of object across time t is very different than their behavior across space n, m. In time, objects move and deform defining continuous trajectories that have no beginning and never end.

1.1 Modeling sequences

Sequences will be represented as functions $f(x, y, t)$, where x, y are the spatial coordinates and t is time. As before, when processing sequences we will work with the discretized version that we will represent as $f[n, m, t]$, where n, m are the pixel indices and t is the frame number. Discrete sequences will be bounded in space and time, and can be stored as arrays of size $N \times M \times P$.

Figure 1.1 illustrates this with one sequence shown in fig. 1.1.a. This sequence has 90 frames and shows people on the street walking parallel to the camera plane and at different distances from the camera. Fig. 1.1.b shows the space-time array $f[n, m, t]$. When we look at a picture we are a looking at a 2D section, $t = constant$, of this cube. But it is interesting to look at sections along other orientations. Fig. 1.1.c and d show sections for $m = constant$ and $n = constant$ respectively. Although they are also 2D images, their structure looks very different from the images we are used to seeing. Fig. 1.1.c shows an horizontal section that is parallel to the direction of motion of the people walking. Here we see straight bands with different orientations. This bands appear to occlude each other. Each band corresponds to one person and its orientation is given by the speed of walk and the direction of motion. Fig. 1.1.d looks like a foto-finish photograph as the ones used in sporting races. In both images (d) and (c), static objects appear as vertical stripes in (b), and horizontal stripes in (d).

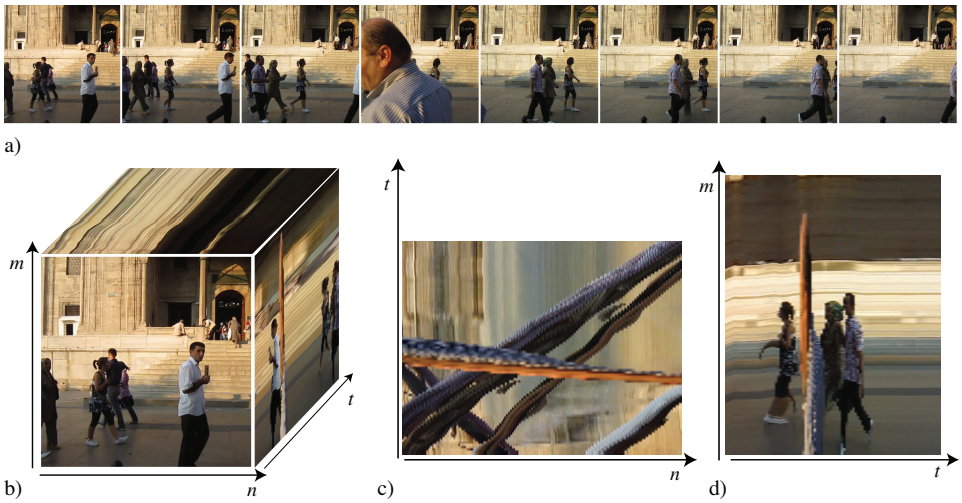

Figure 1.1
a) 8 Frames from a sequence with people walking. The frames are shown at regular time intervals. The full sequence had 90 frames (corresponding to 3 seconds of video). b) Space-time array, $f[n, m, t]$ of size $128 \times 128 \times 90$. c) Section for $m = 50$, d) Section for $n = 75$. Static objects appear as straight lines.

One special sequence is when the image has a global motion with constant velocity (v_x, v_y). In such a case we can write:

$$f(x, y, t) = f_0(x - v_x t, y - v_y t) \tag{1.1}$$

where $f_0(x, y) = f(x, y, 0)$ is the image being translated, and v_x and v_y are constants. We use continuous functions because it allows us to deal with any velocity values. This function assumes also that the brightness of the pixels does not change while the scene is moving (constant brightness assumption). The FT of this globally moving image is:

$$F(w_x, w_y, w_t) = F_0(w_x, w_y)\delta(w_t - v_x w_x - v_y w_y) \tag{1.2}$$

The continuous FT of the sequence is equal to the product of the 2D FT of the static image $f_0(x, y)$ and a delta wall. To better understand this function let's look at a simple example on only one spatial dimension, as shown in figure 1.2.

Figure 1.2 shows the FT for a sequence with one spatial dimension, $f(x, t)$, that contains a blurry rectangular pulse moving at three different speeds towards the left. Fig. 1.2.a shows a sequence when the pulse is static and fig. 1.2.d shows its FT. Across the spatial frequency w_x, the FT is approximately a sinc function. Across the temporal frequency w_t, as the signal is constant, its FT is a delta function. Therefore the FT is a sinc function contained inside a delta wall in the line $w_t = 0$. Fig. 1.2.b shows the same rectangular

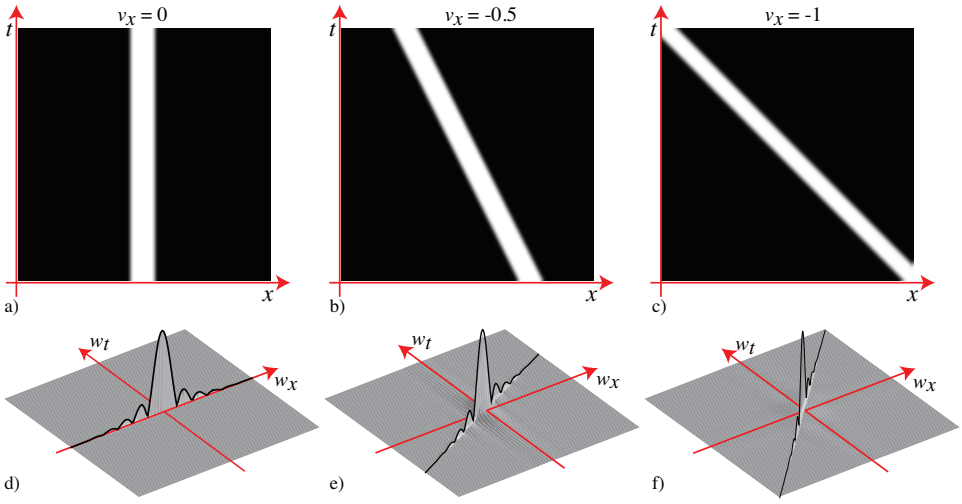

Figure 1.2
a) A sequence with one spatial dimension showing a static rectangular pulse. b) The rectangular pulse moves to the left at a speed $v = -0.5$ and c) moving towards the left, $v = -1$. As we work with discretized signals, speed units are in pixels per frame.

pulse moving towards the left, $v_x = 0.5$. Fig. 1.2.e shows the sinc function but skewed a long the frequency line $w_t + 0.5w_x = 0$. Note that this is not a rotation of the sinc function from fig. 1.2.d as the locations of the zeros lie at the same w_x locations. Fig. 1.2.c shows the pulse moving at a faster speed resulting in a larger skewing of its FT, Fig. 1.2.f.

In general, sequences will be more complex, but the properties of a globally moving image are helpful to understand local properties in sequences. We can also write models for more complex sequences. For instance, a sequence containing a moving object over an static background can be written as:

$$f(x, y, t) = b(x, y)(1 - m(x - v_x t, y - v_y t)) + o(x - v_x t, y - v_y t)m(x - v_x t, y - v_y t) \quad (1.3)$$

where $b(x, y)$ is the static background image, $o(x, y)$ is the object image moving with speed (v_x, v_y), and $m(x, y)$ is a binary mask that moves with the object and that models the fact that the object occludes the background. We let to the reader the work of computing its FT and visualizing the effect of the mask in the Fourier domain.

1.2 Temporal filters

Linear spatio-temporal filters can be written as spatio-temporal convolutions between the input sequence and a convolutional kernel (impulse response). Discrete spatiotemporal filters have an impulse response $h[n, m, t]$. The extension from 2D filter to spatio-temporal

filters does not has any additional complications. We can also classify filters as low-pass, high-pass, etc. But in the case of time, there is another attribute used to characterize filters: causality.

- Causal filters: these are filters with output variations that only depend of the past values of the input. This puts the following constraint: $h[n, m, t] = 0$ for all $t < 0$. This means that if the input is an impulse at $t = 0$, the output will only have non-zero values for $t > 0$. If this condition is satisfied, then the filter output will only depend on the input's past for all possible inputs.
- Non-causal filters: when the output has dependency on future inputs.
- Anti-causal filters: this is the opposite, when the output only depends on the future: $h[n, m, t] = 0$ for all $t > 0$.

Many filters are non-causal and have both causal and anti-causal components (e.g., a Gaussian filter). Note that non-causal filters can not be implemented in practice and, therefore, any filter with an anti-causal component will have to be approximated by a purely causal filter by bounding the temporal support and shifting in time the impulse response.

In this chapter, we have written all the filters as convolutions. However, some filters are better described as difference equations (this is specially important in time). An example of a difference equation is:

$$g[n, m, t] = f[n, m, t] + \alpha\, h[n, m, t - 1] \tag{1.4}$$

where the output g at time t depends on the input at time t and the output at the previous time instant $t - 1$ multiplied by a constant α. We can easily evaluate the impulse response, $h[n, m, t]$, of such a filter by replacing $f[n, m, t]$ with an impulse, $\delta[n, m, t]$. The impulse response is:

$$h[n, m, t] = \alpha^t \delta[n, m]\, u[t] \tag{1.5}$$

where $u[t]$, called the Heaviside step function, is:

$$u[t] = \begin{cases} 0 & \text{if } t < 0 \\ 1 & \text{otherwise} \end{cases} \tag{1.6}$$

Most filters described by differences equations have an impulse response with infinite support. They are called IIR (Infinite Impulse Response) filters. IIR filters can be further classified as stable and unstable. Stable filters are the ones that given a bounded input, $|f[n, m, t]| < A$, produce a bounded output, $|g[n, m, t]| < B$. For this to happen, the impulse response has to be bounded. In unstable filters, the amplitude of the impulse response diverges to infinity. In the previous example, the filter is stable if and only if $|\alpha| < 1$.

Let's now describe some spatio-temporal filters.

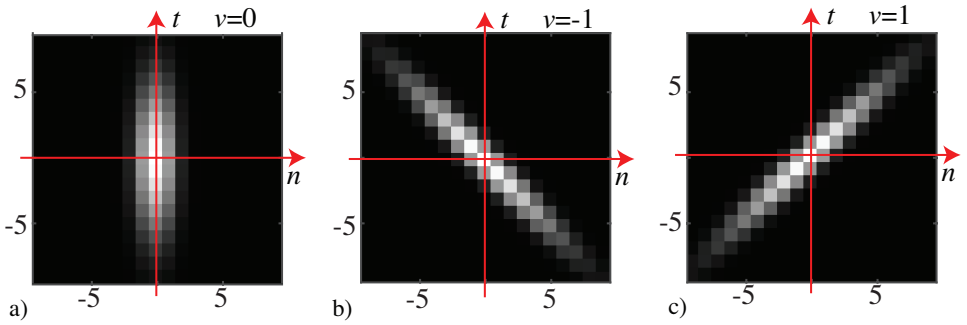

Figure 1.3
a) Spatio-temporal Gaussian with $\sigma = 1$ and $\sigma_t = 4$. b) Same gaussian parameters but skewed by the velocity vector $v_x = -1, v_y = 0$ pixels/frame, c) and $v_x = 1, v_y = 0$ pixel/frame.

1.2.1 Temporal Gaussian

As with the spatial case, we can define the same low-pass filters: the box filter, triangular filters, etc. As an example, let's focus on the Gaussian filter. The spatio-temporal gaussian is trivial extension of the spatial gaussian filter we have seen in section **??**:

$$g(x, y, t; \sigma_x, \sigma_t) = \frac{1}{(2\pi)^{3/2}\sigma_x^2\sigma_t} \exp -\frac{x^2 + y^2}{2\sigma_x^2} \exp -\frac{t^2}{2\sigma_t^2} \tag{1.7}$$

Where σ_x is the width of the gaussian along the two spatial dimensions, and σ_t is the width on the temporal domain. As the units for t and x, y are unrelated, it does not make sense to set all the σs to have the same value.

We can discretize the continuous gaussian by taking samples and building a 3D convolutional kernel. We can also use the binomial approximation. The 3D Gaussian is separable so it can be implemented efficiently as a convolutional cascade of 3 one dimensional kernels. Figure 1.3.a shows a spatio-temporal gaussian. The temporal gaussian is a non-causal filter, therefore it is not physically realizable. This is not a problem when processing a video stored in memory. However, if we are processing an streamed video, we will have to bound and shift the filter to make it causal which will result in a delay in the output.

Figure 1.4.a shows one sequence and figure 1.4.b shows the sequence filtered with the gaussian from figure 1.3.a.This Gaussian has a small spatial width, $\sigma = 1$, and a large temporal width, $\sigma_t = 4$ so the sequence is strongly blurred across time. The moving objects show motion blur and are strongly affected by the temporal blur, while the static background is only affected by the spatial width of the gaussian.

How could we create a filter that keeps sharp objects that move at some velocity (v_x, v_y) while blurring the rest? Figure 1.4.c shows the desired output of such a filter. The bottom image shows one frame for a sequence filtered with a kernel that keeps sharp objects moving left at 1 pixel/frame while blurring the rest. This filter can be obtained by skewing the

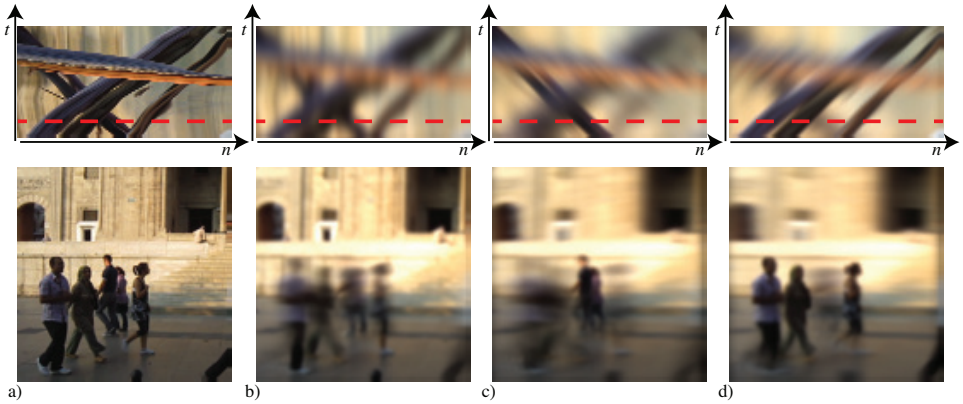

Figure 1.4
a) One frame from the input sequence and the space-time section (on top). b) Output when convolving the the gaussian from fig. 1.3.a. c) Output of the convolution with fig. 1.3.b, and d) output of the convolution with fig. 1.3.c.

gaussian:

$$g_{v_x,v_y}(x,y,t) = g(x - v_x t, y - v_y t, t) \tag{1.8}$$

This directional blur is not a rotation of the original Gaussian as the change of variables is not unitary, but the same effect could be obtained with a rotation. Figure 1.4.c shows the effect when $v_x = -1, v_y = 0$. The gaussian is shown in fig. 1.3.b. The space-time section shows how the sequence is blurred everywhere except one oriented bad corresponding to the person walking left. Figure 1.4.d shows the effect when $v_x = 1, v_y = 0$. The output of this filter looks as if the camera was tracking one of the objects while the shutter was open, producing a blurry image of all the other objects.

1.2.2 Temporal derivatives

Spatial derivatives were useful to find regions of image variation such as object boundaries. Temporal derivatives can be used to locate moving objects. We can approximate a temporal derivative for discrete signals as:

$$f[m,n,t] - f[m,n,t-1] \tag{1.9}$$

As in the spatial case, it is useful to compute temporal derivatives of spatio-temporal gaussians:

$$\frac{\partial g}{\partial t} = \frac{-t}{\sigma_t^2} g(x,y,t) \tag{1.10}$$

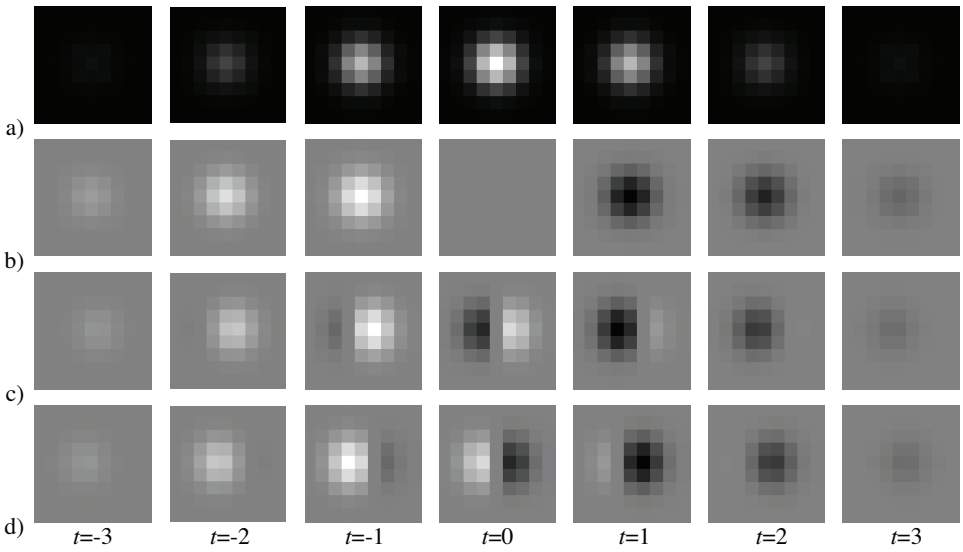

Figure 1.5
Visualization of the space-time Gaussian. The gaussian has a width of $\sigma^2 = \sigma_t^2 = 1.5$, and has been discretized as a 3D array of size $7 \times 7 \times 7$. Each image shows one frame. a) Gaussian b) The partial derivative of the Gaussian with respect to t. c) Derivative along $v = (1, 0)$ pixels/frame. d) $v = (-1, 0)$ pixels/frame.

where $g(x, y, t)$ is the Gaussian as written in eq. 1.7. We can compute the spatio-temporal gradient of a gaussian:

$$\nabla g = \Big(g_x(x, y, t), g_y(x, y, t), g_t(x, y, t)\Big) = \Big(-x/\sigma^2, -y/\sigma^2, -t/\sigma_t^2\Big)\, g(x, y, t) \tag{1.11}$$

What should we do if we want to remove only the objects moving at a particular velocity?

In the case of a moving image with velocity (v_x, v_y), the sequence is $f(x, y, t) = f_0(x - v_x t, y - v_y t)$, we can compute the temporal derivative of $f(x, y, t)$ as:

$$\frac{\partial f}{\partial t} = \frac{\partial f_0}{\partial t} = -v_x \frac{\partial f_0}{\partial x} - v_y \frac{\partial f_0}{\partial y} \tag{1.12}$$

If we compute the gradient of the gaussian along the vector $\Big[1, v_x, v_y\Big]$:

$$h(x, y, t; v_x, v_y) = g_t + v_x g_x + v_y g_y = \nabla g \left(1, v_x, v_y\right)^T \tag{1.13}$$

and we convolve it with $f_0(x - v_x t, y - v_y t)$ we get a zero output (using eq. 1.12):

$$f_0(x - v_x t, y - v_y t) \circ h = f_0(x - v_x t, y - v_y t) \circ \Big(g_t + v_x g_x + v_y g_y\Big) = \left(\frac{\partial f_0}{\partial t} + v_x \frac{\partial f_0}{\partial x} + v_y \frac{\partial f_0}{\partial y}\right) \circ g = 0 \tag{1.14}$$

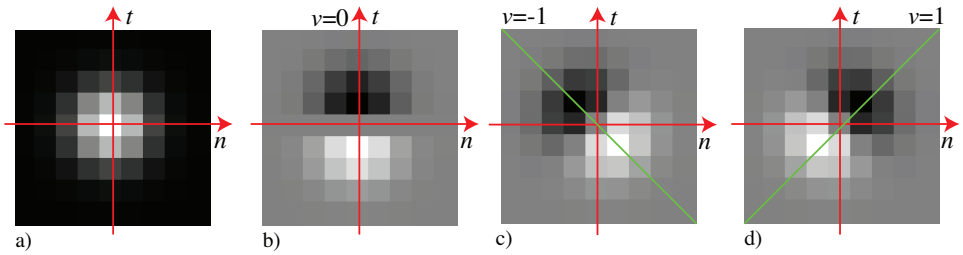

Figure 1.6
Spatio-temporal Gaussian $g[n,t]$ and derivatives. a) Gaussian with $\sigma^2 = 1.5$. b) Partial derivative with respect to t. c) Partial derivative along $(1,-1)$, d) Partial derivative along $(1,1)$.

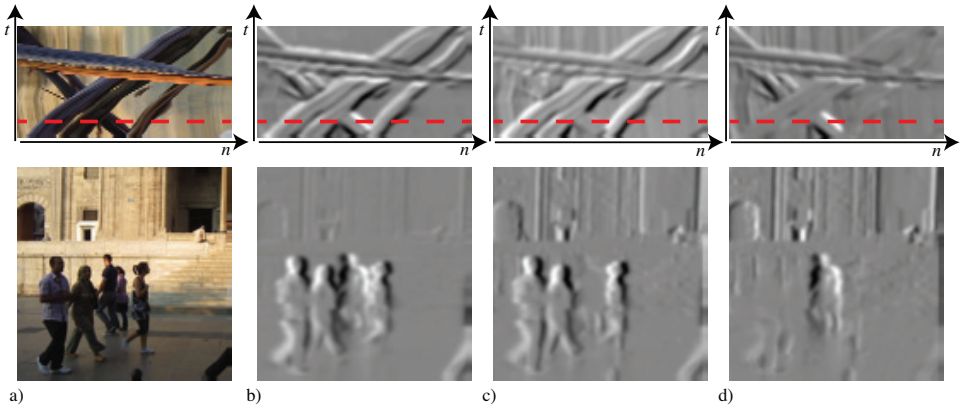

Figure 1.7
a) Input sequence. b) $v_x = v_y = 0$, c) $v_x = 1$ pixels/frame. d) $v_x = -1$ pixel/frame.

The filter h is shown in figure 1.5 as a sequence for different velocities. In the example shown in the figure, the gaussian (fig. 1.5.a) has a width of $\sigma^2 = \sigma_t^2 = 1.5$, and has been discretized as a 3D array of size $7 \times 7 \times 7$. Figures b,c and d show the filter h for different velocities: $(v_x, v_y) = (0,0)$, $(1,0)$ and $(-1,0)$. Fig. 1.6 shows the corresponding space-time sections of the same spatio-temporal gaussian derivatives. Both visualizations are equivalent and help to understand how the filter works.

Such a filter will cancel any objects moving at the velocity (v_x, v_y). By using different filters, each one computing derivatives a long different space-time orientations, we

1.2.3 Spatiotemporal Gabor filters

Just as we did with Gaussian derivatives, extending Gabor filter for motion analysis is a direct generalization of the x-y 2D Gabor function to a x-y-t 3D Gabor function. Figure 1.8.a

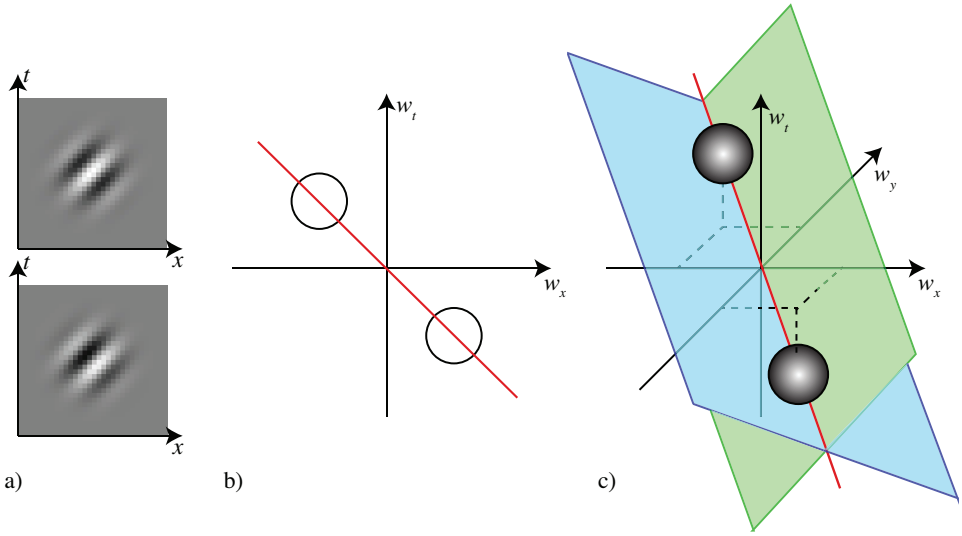

Figure 1.8
Space-time Gabor filters. a) Cosine and sine *x-t* Gabor filter, and b) the sketch of its transfer function. c) Sketch of the transfer function of a spatiotemporal Gabor filter in 2 spatial dimensions (*x-y-t*).

shows a x-t (cosine and sine) Gabor function in 1 spatial dimension, and Figure 1.8.b shows a sketch of its Fourier transform. This function is selective to signals translating to the right with a speed $v = 1$, i.e. $i(x - t)$. The red line in Figure 1.8.b shows Dirac line that contains the energy of the moving signal.

In 2 spatial dimensions, Figure 1.8.c shows the sketch of the Gabor transfer function. Note that the x-y-t Gabor filter is not selective to velocity. If we have a 2D moving signal $i(x - v_x t, y - v_y t)$, the Fourier transform is contained inside a Dirac plane. Therefore, there are an infinite number of planes that will pass by the frequencies of the Gabor filter. All those planes intersect the red line shown in Figure 1.8.c. A single Gabor filter can not disambiguate the input velocity.

Equation of the velocity as a function of the frequency tuning.

1.3 Velocity-tuned filters

How can we measure input velocity? There are many different approaches in the computer vision community for measuring motion. Here we show that it is possible to measure motion even with the simple processing machinery that we've developed so far.

We can use quadrature pairs of oriented filters in space-time to find motion speed and direction in the video signal. We just need to find the space-time orientation of strongest response. Figure 1.8.a shows a set of Gabor filters sampling the space-time frequency

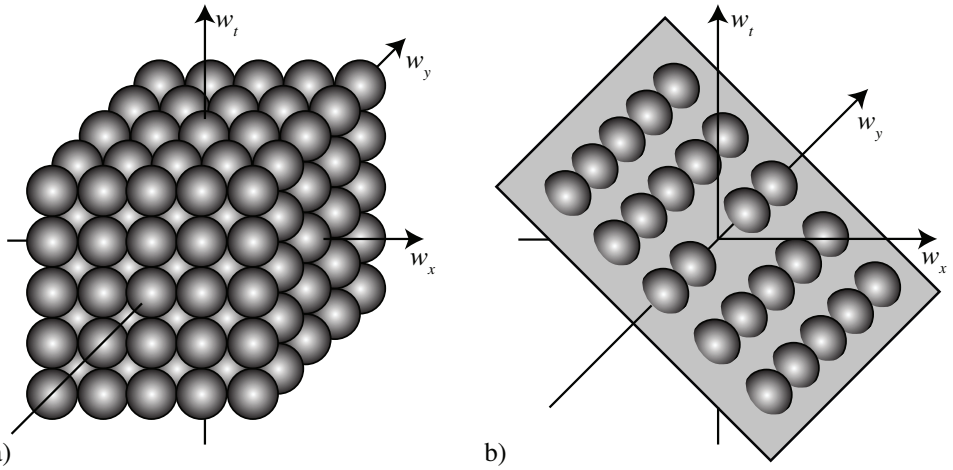

Figure 1.9
a) Space-time Gabor filters tiles. b) Set of Gabor filters selective to a particular velocity.

domain. When the input contains a moving signal, we can use a set of filters to identify the plane in the Fourier domain that contains the input energy. Figure 1.8.b shows the subset of filters that have the strongest output for a particular input motion.

As an illustration, Figure 1.10 shows one possible architecture to create velocity-selective units. The first layer is composed by Gabor filters (cosine and sine) which are frequency-selective units. The outputs are combined according to different planes in the Fourier domain to create velocity-selective outputs.

Given an input sequence, one can estimate velocity by looking at the velocity-tuned unit with the strongest response.

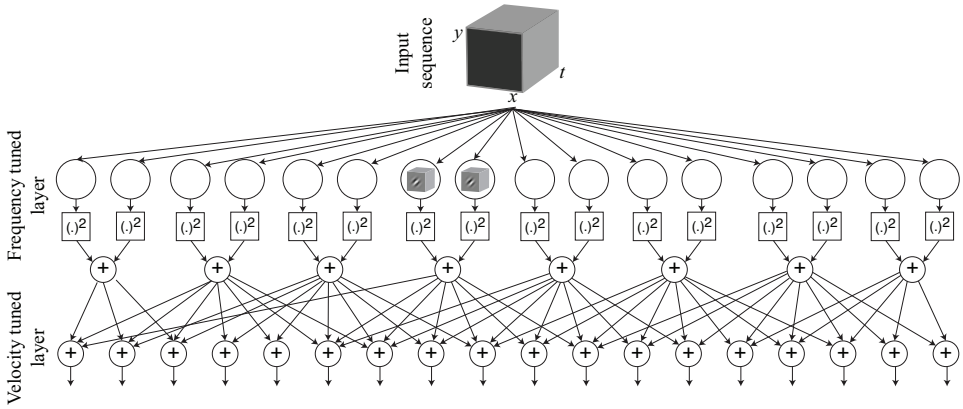

Figure 1.10

Architecture to create velocity-selective units. The first layer is composed by space-time Gabor filters (cosine and sine) which are frequency-selective units. Here we represent the impulse response of each filter by a small x-y-t cube. For each quadrature pair we compute the amplitude. Then amplitude outputs are combined according to form different planes in the Fourier domain to create velocity-selective outputs. A normalization layer can be added to normalize the outputs by dividing every output by the sum of all the amplitudes (not shown). The full architecture is non-linear.

Bibliography