Linear Filtering

Goal: Provide a short introduction to linear filtering that is directly relevant for computer vision.

Here the emphasis is on:

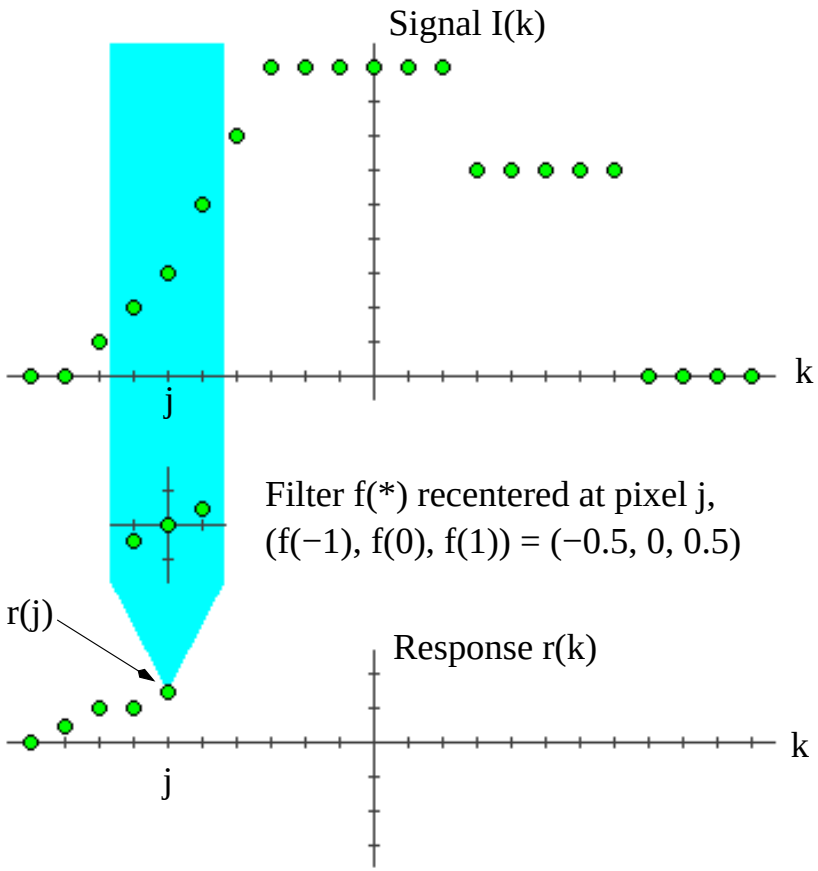
- the definition of correlation and convolution,
- using convolution to smooth an image and interpolate the result,
- using convolution to compute (2D) image derivatives and gradients,
- computing the magnitude and orientation of image gradients.

We discuss how the filters we use in 2D images can be extended to compute spatiotemporal gradients for videos.

Finally, we introduce the concept of the gradient tensor, along with its magnitude and spatial orientation. We show the application of gradient tensors to colour images.

Readings: Szeliski, Section 3.2 [optional: 3.4-5].

Correlation for 1D Signals



The **correlation** of the filter $f(k)$ with the image $I(k)$ is the new signal $r(k)$ defined by

$$r(j) \equiv \sum_{s=-K}^{K} f(s)I(j+s).$$

Here $f(k)$ is assumed to be zero for $|k|$ larger than the filter radius K.

Figure is from James Sewart's filtering applet: `http://research.cs.queensu.ca/~jstewart/applets/filter/filter.html`

Correlation and Convolution

Defn. The correlation of a 2D filter f with a monochromatic (i.e., grayscale) image I:

$$\textbf{Correlation: } r(j,k) \equiv \sum_{s=-K}^{K} \sum_{t=-K}^{K} f(s,t) I(j+s, k+t).$$

Here $f(j,k)$ is zero for $|j|$ or $|k|$ larger than the filter radius K.

Intuitively, to compute the correlation response $r(j,k)$,

- re-center the filter so that $f(0,0)$ is "aligned" with $I(j,k)$ (see 1D example on previous page),
- pointwise multiply the filter and the image values at each pixel (for which $f \neq 0$),
- sum these products to form $r(j,k)$.

Defn. The convolution of a 2D filter f with an image I:

$$\textbf{Convolution: } r(j,k) \equiv \sum_{s=-K}^{K} \sum_{t=-K}^{K} f(-s,-t) I(j+s, k+t). \tag{1}$$

We denote convolution as $r = f * I$. Note $f * I$ is the same as correlation using the flipped filter kernel, i.e., $g(j,k) = f(-j,-k)$. For historical reasons we emphasize convolution over correlation.

Image Boundary Treatments

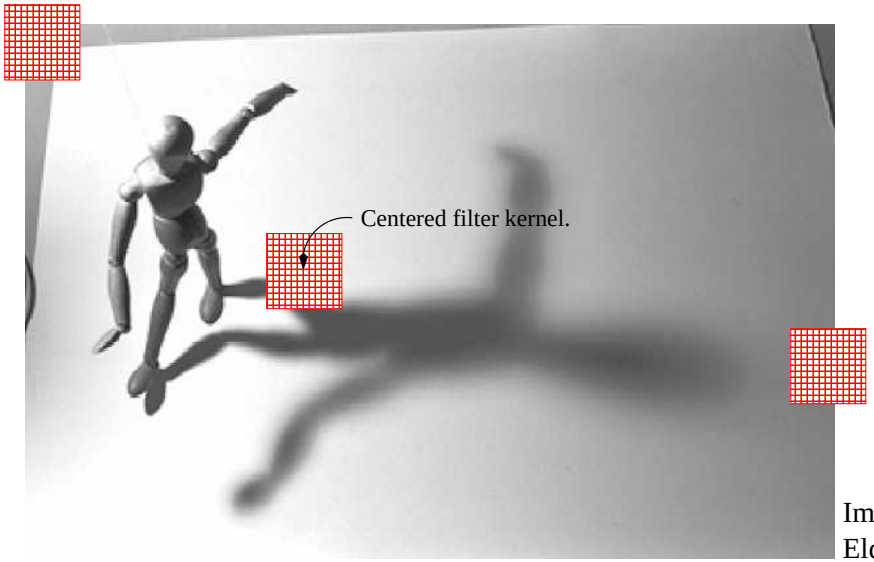


Image from
Elder et al, 1998.

When the filter kernel extends beyond the edge of the image, we use a *boundary treatment*.

Options for this include (see `upConv` in the Matlab `iseToolbox`):

- **repeat**, use the nearest image pixel within the image boundary;
- **reflect** the image across its boundaries;
- **zero**, use zero for any pixel beyond the image boundary;
- **dont-compute** the response when the re-centered filter extends beyond the filter boundary.

We often use the repeat rule, but care must be taken interpreting the response near the image boundary.

Separable Filter Kernels

The direct implementation of the convolution of a $L \times L$ filter kernel with an $N \times M$ image requires $O(L^2NM)$ operations.

The same convolution can also be obtained using the Fast Fourier Transform (typically with a periodic boundary treatment) in $O(NM \log(NM))$ operations.

We can often use **separable** filter kernels which, by definition, can be written as a product of two functions of only one variable each, that is,

$$f(j,k) = f_1(j)f_2(k).$$

It can be shown that $r = f * I = f_1 * (f_2 * I)$. These two successive 1D convolutions require only $O(LNM)$ operations (compared to $O(L^2NM)$ for the direct approach) and allow for the use of all the previous boundary treatments.

Due to the use of *image pyramids* (see Szeliski, Sec. 3.5) the size, L, of the filter kernels can typically be kept reasonably small.

Kernels with Real-Valued Arguments

Consider a filter kernel $f(\vec{x})$ defined over a continuous range of values, $\vec{x} = (x, y)^T \in \mathbb{R}^2$. As before, assume $f(\vec{x}) = 0$ outside the range $[-K, K] \times [-K, K]$.

Given such a kernel, it is useful to change variables in the definition of convolution, namely equation (1). In particular, replace (j, k) by real-valued variables (x, y), and set $m = x + s$, and $n = y + t$ (both integer-valued). With this change of variables in, we find from (1) that

$$r(\vec{x}) \equiv \sum_{m,n \in \mathbb{Z}^2, f \neq 0} f(x - m, y - n) I(m, n). \tag{2}$$

Here the image $I(j, k)$ is always evaluated at integer-valued points (i.e., pixels or "grid points"), but the result $r(\vec{x})$ can now be evaluated for real-valued points.

The idea in (2) is simply to center the flipped kernel at any real-valued $\vec{x}$ rather than only at integer-valued pixel coords.

Applications of Kernels with Real-Valued Domains

Filtering and Interpolation. Let the filtered image be $r(\vec{x}) = f(\vec{x}) * I$. Since $\vec{x}$ here is a continuous variable, we can clearly sample $r(\vec{x})$ at a higher rate than the original image. That is, we can *interpolate* the filtered image $r(\vec{x})$.

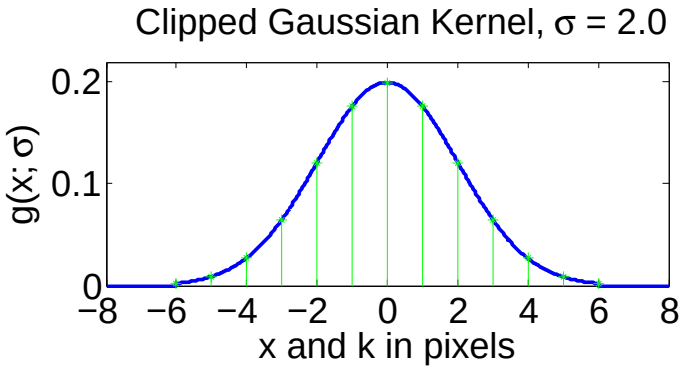
Note, we are interpolating $r(\vec{x})$, not the original image $I(j,k)$. To do the latter we need to ensure that the filter kernel is chosen such that $r(j,k) = I(j,k)$. (Optional further reading.)

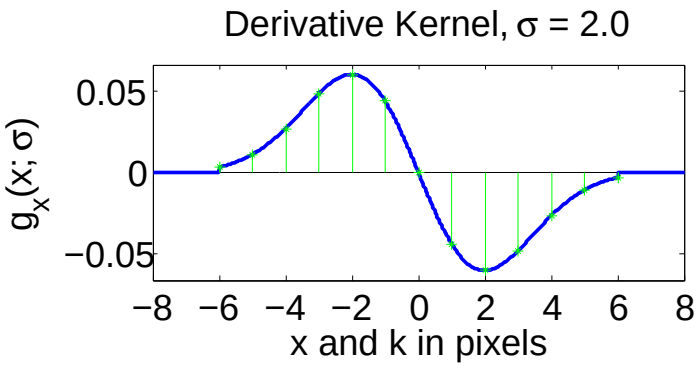
Spatial Deriviatives. It also follows that the x and y derivatives of $r(\vec{x})$, namely $r_x(\vec{x})$ and $r_y(\vec{x})$, satisfy

$$r_x(\vec{x}) = f_x * I, \quad \text{and } r_y(\vec{x}) = f_y * I,$$

where $f_x(\vec{x})$ and $f_y(\vec{x})$ are the x and y derivatives of the kernel $f(\vec{x})$.

Example: Clipped Gaussian Filter





For example, consider the 1D clipped Gaussian kernel:

$$g(x;\sigma) = \begin{cases} Ce^{-x^2/(2\sigma^2)}, & \text{for } -K \le x \le K, \\ 0, & \text{otherwise.} \end{cases} \tag{3}$$

Here $x \in \mathbb{R}$ and

- $\sigma \ge 1$ is the scale parameter, it is proportional to the width of the kernel in x,
- $K > 0$ is an integer specifying the radius in which the kernel is non-zero, typically $K \approx 3\sigma$,
- $1/C = \sum_{k=-K}^{K} e^{-x^2/(2\sigma^2)}$, which normalizes $g(x;\sigma)$ to sum to one over all integer arguments.

The kernel $g(x;\sigma)$ and its first derivative $g_x(x;\sigma)$ (ignoring the discontinuities) are shown above.

Image Smoothing

Convolution with the 2D clipped Gaussian filter kernel, $g(\vec{x};\sigma) \equiv g(x;\sigma)g(y;\sigma)$, blurs the image:

Using:
$\sigma = 2$

The kernel $g(\vec{x};\sigma)$ is shown in the bottom right corner of the smoothed image, and is circularly symmetric (modulo the clipping at $\pm K$).

Image Derivatives

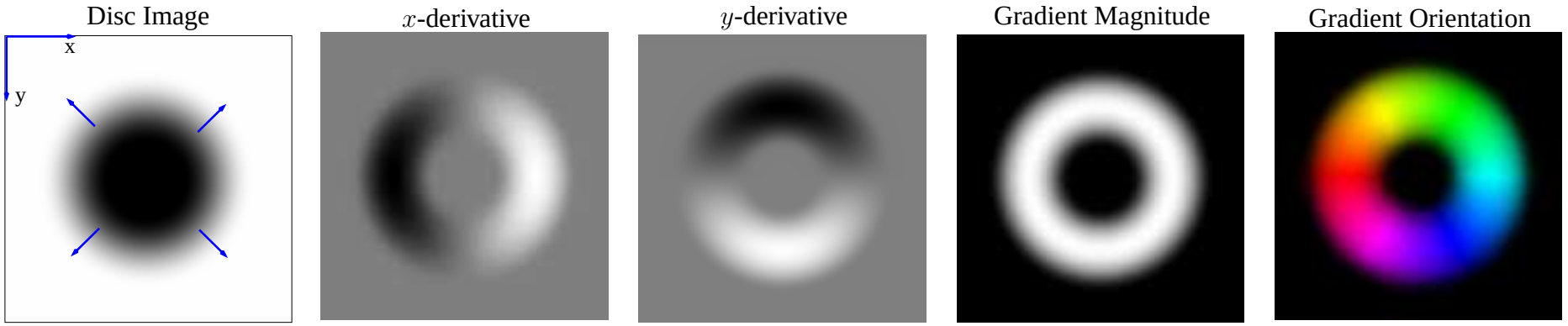
The blurred image along with its x and y derivatives are shown below. The dominant grey tone in the derivative images denotes a response of zero, with positive (negative) values being lighter (darker).

The x and y derivative filters are $g_x(\vec{x}) = g_x(x)g(y)$ and $g_y(\vec{x}) = g(x)g_y(y)$ (see the inlaid images in the bottom-right corners).

Image Gradient

The *gradient* of an image $r(\vec{x})$ is defined to be the following 2-vector at every point $\vec{x}$,

$$\vec{\nabla} r(\vec{x}) \equiv \begin{pmatrix} r_x(\vec{x}) \\ r_y(\vec{x}) \end{pmatrix}. \tag{4}$$



The magnitude of the gradient $S(\vec{x})$ is defined to be the vector 2-norm of the gradient, namely $S(\vec{x}) = ||\vec{\nabla} r(\vec{x})|| = (r_x(\vec{x})^2 + r_y(\vec{x})^2)^{1/2}$. The magnitude is zero only if both the x and y derivatives of $r(\vec{x})$ vanish.

When $||\vec{\nabla} r(\vec{x})||$ is nonzero, the gradient (vector) at $\vec{x}$ is in the "steepest ascent" direction (i.e., the direction $r(\vec{x})$ increases the fastest). We define the gradient angle as $\theta = \text{atan2}(r_y(\vec{x}), r_x(\vec{x}))$, and the gradient direction to be $\vec{u}(\theta) = (cos(\theta), sin(\theta))^T = \vec{\nabla} r(\vec{x}) / ||\vec{\nabla} r(\vec{x})||$.

Image Gradient

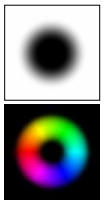
The gradient identifies the local magnitude and direction of variation in the blurred image $r(\vec{x})$.

Original Image

Gradient Magnitude

Gradient Orientation

Colour Key



The only remaining parameter here is the scale parameter σ (we used $\sigma = 2$). As σ increases the original image is increasingly blurred, and detail is lost. On the other hand, the responses are less sensitive to image noise, and will have better performance on smooth ramps in the image.

Image Gradient: Fine Scale

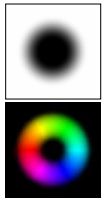
The results for $\sigma = 1$ are shown below.

Original Image

Gradient Magnitude

Gradient Orientation

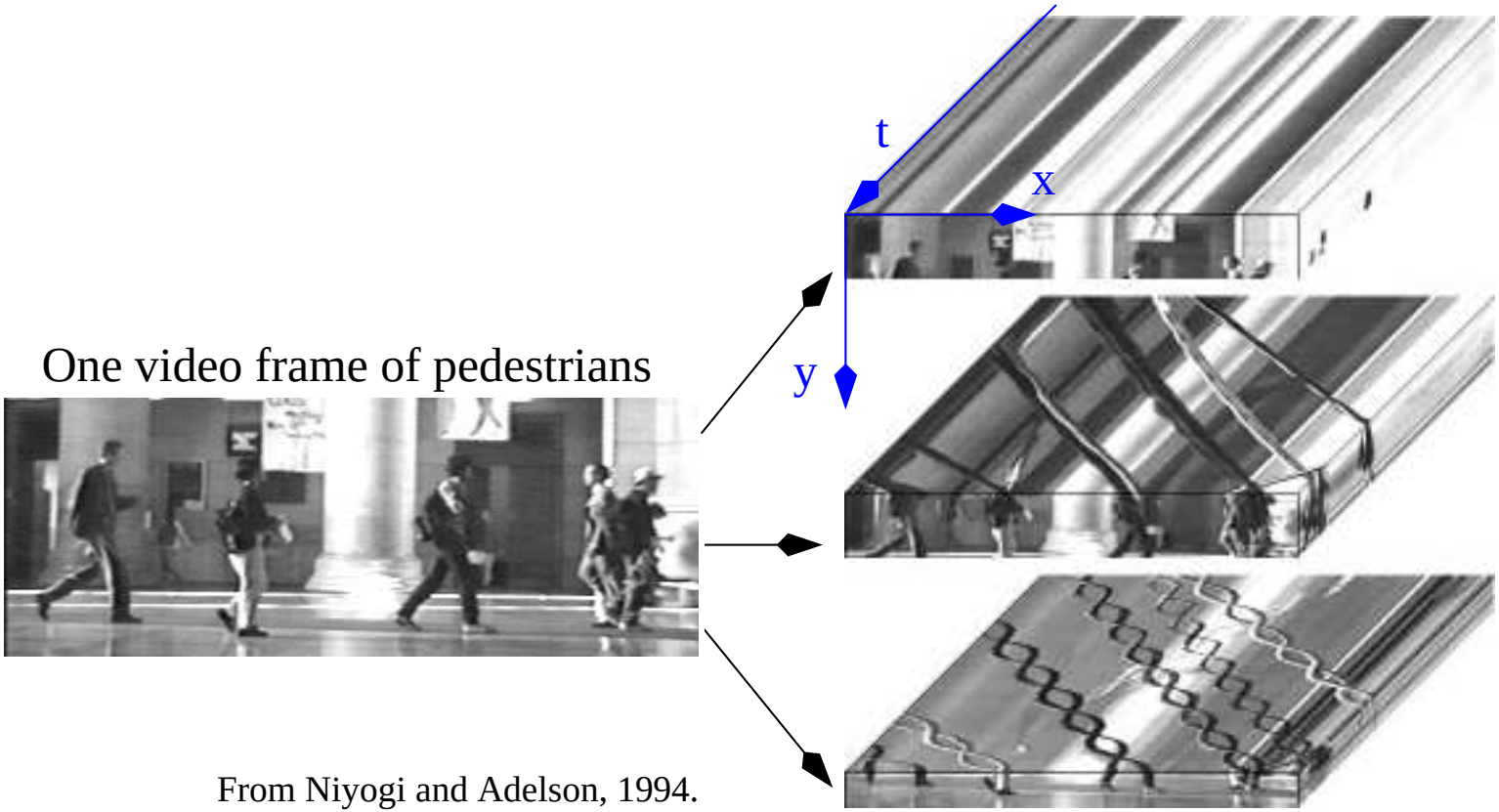
Colour Key



Compare these results with those for $\sigma = 2$ (previous slide). While the detail is significantly sharper here (for $\sigma = 1$), the gradient directions are more sensitive to image noise (e.g., on the jagged edges and within the soft shadows in the original image). See Elder et al, 1998, for automatic scale selection.

Spatiotemporal Images

A sequence of video images forms a space-time volume image.



One video frame of pedestrians

From Niyogi and Adelson, 1994.

Gradients, gradient magnitude and orientation, can all be measured as before, for example, using a separable filter $g(\vec{x}, t) = g(x; \sigma)g(y; \sigma)g(t; \sigma)$ along with its x, y, and t derivatives. Gradients again point in the "steepest ascent" direction, perpendicular to 2D surfaces of constant brightness in $(\vec{x}, t)$.

Multispectral Images

Consider an n-dimensional image $\vec{I}(\vec{x}) = (I_1(\vec{x}), \ldots, I_n(\vec{x}))^T$. For example, an RGB image has $n = 3$. We could form a $n = 4$ dimensional image by adding an infrared channel, say.

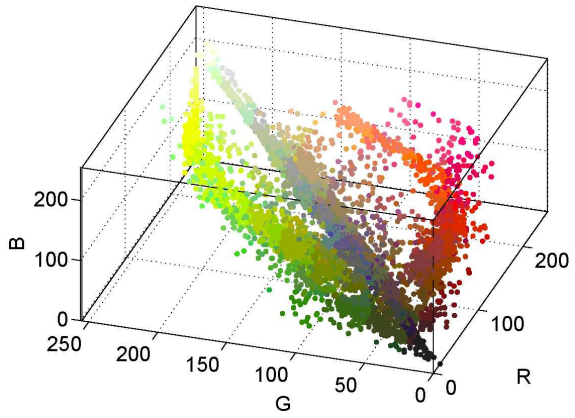
Hats Image

Red (R) Component

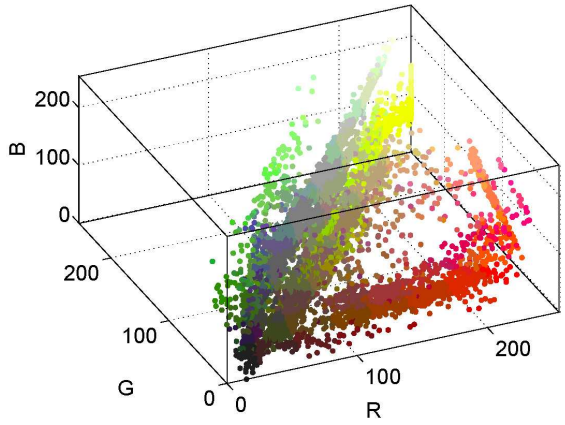
Green (G) Component

Blue (B) Component

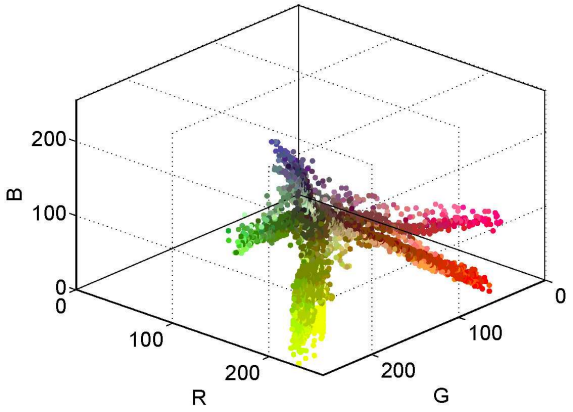
RGB Space



RGB Space



RGB Space (View Along Gray Axis)



RGB-space is a cube, with each side of length 255 (for 8-bit images). The gray axis is the diagonal from $(0, 0, 0)^T$ (Black) to $(255, 255, 255)^T$ (White).

Projecting Multispectral Images to Monochromatic

For an n-dimensional image $\vec{I}(\vec{x}) = (I_1(\vec{x}), \ldots, I_n(\vec{x}))^T$ our previous filtering techniques can be applied to any monochromatic image $h(\vec{I}(\vec{x}))$, where h is a (smooth) mapping from $\mathbb{R}^n$ to $\mathbb{R}$.

For example, a simple form for h is a linear mapping,

$$h(\vec{I}) = \vec{w}^T \vec{I},$$

where $\vec{w}$ is a constant n-vector (aka the projection direction).

Suitable projection directions $\vec{w}$ can be selected to highlight different aspects of the original image $\vec{I}$ in the monochromatic image $\vec{w}^T \vec{I}(\vec{x})$.

We illustrate the monochrome images obtained from several different projection directions in the next two slides.

Colour-Opponent Channels

For example, consider choosing projection directions to (roughly) decorrelate a typical RGB image $\vec{I} = (I_R, I_G, I_B)^T$. Define

$$\begin{pmatrix} I_{Br} \\ I_{YB} \\ I_{RG} \end{pmatrix} = C \begin{pmatrix} I_R \\ I_G \\ I_B \end{pmatrix}, \quad \text{where } C = \begin{pmatrix} 1/\sqrt{3} & 1/\sqrt{3} & 1/\sqrt{3} \\ 1/\sqrt{6} & 1/\sqrt{6} & -2/\sqrt{6} \\ 1/\sqrt{2} & -1/\sqrt{2} & 0 \end{pmatrix} \begin{matrix} \leftarrow \text{ Brightness} \\ \leftarrow \text{ R + G - 2B, aka Y-B} \\ \leftarrow \text{ R - G} \end{matrix} \quad (5)$$

This matrix C is a unitary transformation of the RGB coordinates, i.e., $C^T C = I$ (note $det(C) = -1$). This transformation very roughly decorrelates the RGB channels (see `colourTutorial.m` in the `utvis` toolbox).

Note each component of the transformed image vector, say I_{YB}, has the form $\vec{w}^T \vec{I}(\vec{x})$ where $\vec{w}^T$ is the corresponding row of C.

Caution: A significant amount of research has been done on colour spaces. The mapping C above is greatly simplified and is not meant to be an optimal choice. (For more information about colour spaces, begin by looking up "Color model" in Wikipedia.)

Example: Colour Opponent Channels

These colour opponent channels are illustrated in the following example.

Hats Image

Red (R) Component

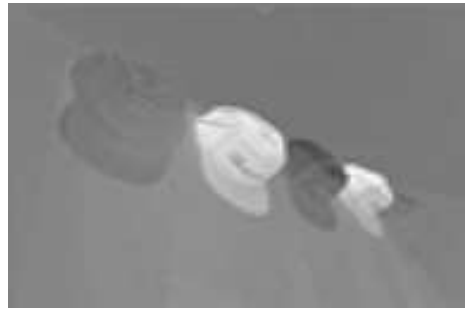
Green (G) Component

Blue (B) Component

Brightness Component

Y-B Component

R-G Component



Note the R and G images appear very similar, indicating a typically strong correlation. Also note that, in each of the monochromatic images, there are small image regions where the boundaries of the hats have low contrast. For these regions the gradients will be either small or dominated by other structure in the region. Thus it may be hard to find a single projection direction, $\vec{w} \in \mathbb{R}^3$, for which the hat boundaries all have large gradients in the single monochromatic image $\vec{w}^T \vec{I}(\vec{x})/||\vec{w}||$.

Image Gradient Tensors: Magnitude and (Spatial) Direction

Alternatively, it is possible to consider all n channels of $\vec{I}(\vec{x})$ simultaneously.

Consider the Jacobian $J(\vec{x}) = \frac{\partial \vec{I}}{\partial \vec{x}}$. This is an $n \times 2$ matrix (aka tensor), with the k^{th} row given by the x and y derivatives of I_k, that is, $(J_{k,1}(\vec{x}), J_{k,2}(\vec{x})) = [\vec{\nabla} I_k(\vec{x})]^T = (I_{k,x}(\vec{x}), I_{k,y}(\vec{x}))$.

The gradient tensor magnitude, $S(\vec{x})$, and direction, $\vec{u}(\theta(\vec{x}))$, are defined using

$$S(\vec{x}) \;=\; \max_{0 \le \phi < \pi} ||RJ(\vec{x})\vec{u}(\phi)||, \;\; \theta(\vec{x}) \;=\; \arg\max_{0 \le \phi < \pi} ||RJ(\vec{x})\vec{u}(\phi)||, \tag{6}$$

where $\vec{u}(\phi) = (cos(\phi), sin(\phi))^T$, $||\cdot||$ denotes the usual 2-norm, and R is a selected $n \times n$ weighting matrix. The gradient direction is then $\vec{u}(\theta(\vec{x}))$ (aka, the right principal vector of the weighted Jacobian, $RJ(\vec{x})$). See DiZenzo, 1986, for a closed form solution of $\theta(\vec{x})$.

Note that, in contrast to the gradient direction in monochromatic images, the gradient tensor direction is only defined up to a sign. That is, both $\vec{u}(\theta(\vec{x}))$ and $-\vec{u}(\theta(\vec{x}))$ are solutions.

Image Gradient Tensors: Left Principal Vector

The left principal vector of the weighted Jacobian $RJ(\vec{x})$ is defined to be

$$\vec{w}(\vec{x}) = \frac{1}{S(\vec{x})} RJ(\vec{x})\vec{u}(\theta(\vec{x})). \tag{7}$$

This $\vec{w}(\vec{x}) \in \mathbb{R}^n$ is, in a sense, a locally optimal choice for the projection vector $\vec{w}$ discussed above.

To see this optimality, consider the monochromatic image $H_w(\vec{x}) = \vec{w}^T R\vec{I}(\vec{x})/||\vec{w}||$, where $\vec{w}$ is a constant vector. Here we have divided by $||\vec{w}||$ to avoid a trivial dependence of the gradient magnitude of $H_w(\vec{x})$ on the magnitude of $\vec{w}$

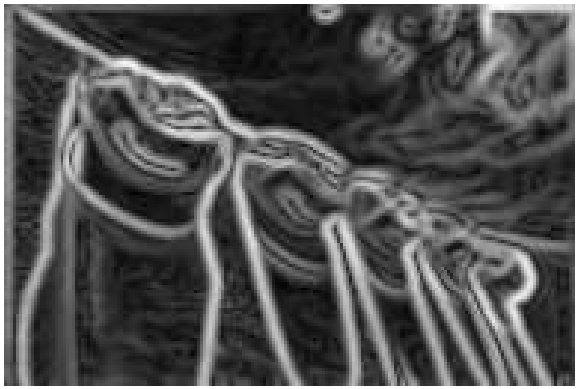
Then it follows that, at every $\vec{x}$, the left principal vector *maximizes* the (monochromatic) gradient magnitude over all images of the form $H_w(\vec{x})$. We omit the proof of this (but hint that the interested reader should consider the singular value decomposition (SVD) of the gradient tensor $RJ(\vec{x})$).

For scalar images (i.e., $n = 1$ and $R = 1$), this definition of the gradient tensor magnitude and direction agrees with our previous definition, with the one exception that the gradient tensor direction is independent of sign. In particular, it can point in the steepest ascent or descent direction.

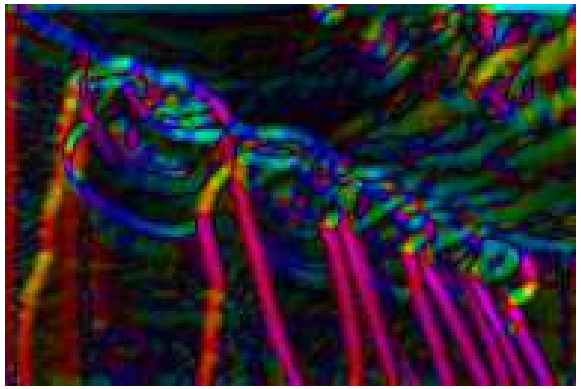
Example Using Gradient Tensors

Brightness Image

Gradient Magnitude

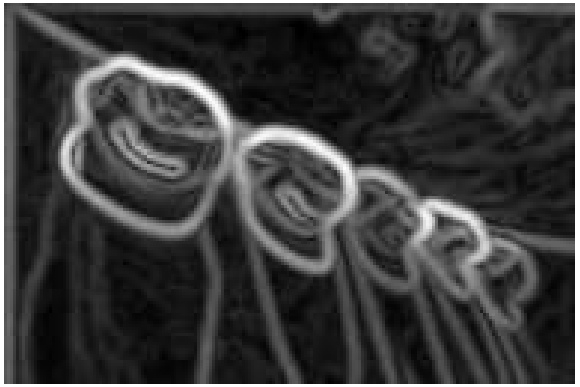


Gradient Orientation

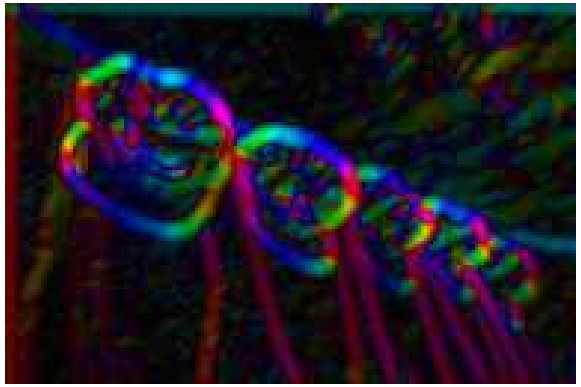


Original Image

Gradient Tensor Magnitude



Gradient Tensor Orientation



Here we used $R = \text{diag}([1, 4, 4])C$ (borrowing diag(·) from Matlab) to emphasize the colour opponent channels over the brightness channel. Note that the hat boundaries are all clearly defined in the gradient tensor approach, which is not true of the results from the brightness image alone. This illustrates the adaptive selection of the left principal direction $\vec{w}(\vec{x})$ that is inherent in this approach.

Further Information

Linear filtering is a rich subject, of which we have (intentionally) given only a brief sketch. Important subjects that students can entirely skip in this course include:

- Fourier analysis,
- sampling, and aliasing,
- signal interpolation (more generally),
- upsampling and downsampling signals,
- filter design,

The textbook by Castleman, 1995, provides an excellent introductory treatment of most of these topics (for any student interested in further study outside of this course).

Alternatively, Sections 3.2 and 3.4-6 of the textbook by Szeliski provide more information, while Section 3.8 provides an excellent set of pointers into recent literature.

References

Castleman, K.R., **Digital Image Processing**, Prentice Hall, 1995

Silvano Di Zenzo, A Note on the Gradient of a Multi-Image, *Computer Vision, Graphics, and Image Processing*, 33, 1986, pp.116-125.

James Elder and Steven Zucker, Local Scale Control for Edge Detection and Blur Estimation, *IEEE Transactions on PAMI*, 20(7), 1998, pp.699-716.

Sourabh Niyogi and Edward Adelson, Analyzing Gait With Spatiotemporal Surfaces, *In IEEE Workshop on Motion of Non-Rigid and Articulated Objects*, 1994, pp.64-69.