# Lowpass and Bandpass Pyramids

Carlo Tomasi

It is often useful to analyze an image $I(\mathbf{x})$ at different scales. One can then form a stack of images obtained by repeatedly blurring the input image:

$$\begin{aligned} B_0 &= I \\ B_\ell &= B_{\ell-1} \otimes S_\sigma \quad \text{for} \quad \ell = 1, \ldots, L \end{aligned} \tag{1}$$

where $S_\sigma(\mathbf{x})$ is a smoothing kernel, typically a Gaussian with width $\sigma$, and the symbol '$\otimes$' denotes convolution. The larger $\sigma$, the more high-frequency information is suppressed at every level of smoothing, and analysis of $B_\ell$ reveals finer or coarser structures in the image depending on the value of the *level* $\ell$. Figure 1 shows the result of blurring an input image $L = 7$ times with a Gaussian kernel with $\sigma = 2$ pixels. Note the loss of detail at higher levels ($\ell > 0$) of the stack.

The convolution of a Gaussian with parameter $\sigma_1$ with another Gaussian with parameter $\sigma_2$ is a Gaussian with parameter $\sigma = \sqrt{\sigma_1^2 + \sigma_2^2}$. Because of this, convolving an image $\ell$ times with a Gaussian with parameter $\sigma$ is the same as convolving the same image once with a Gaussian with parameter

$$\sigma_\ell = \sqrt{\ell}\sigma$$

for each $\ell$. So we can also write

$$\begin{aligned} B_0 &= I \\ B_\ell &= B_0 \otimes S_{\sigma|_\ell} \quad \text{for} \quad \ell = 1, \ldots, L \; . \end{aligned}$$

Of course, the iterative smoothing procedure (1) is more efficient, because the kernels are smaller.

The sampling frequency of the images in the stack is high when compared to the spatial frequencies contained in the images for $\ell > 0$, so the blurred images $B_\ell$ can be sampled after filtering without significant loss of information. Without getting into the quantitative aspects of sampling and image bandwidth, it turns out that most of the image information is preserved if every time the image is blurred with a Gaussian filter with parameter $\sigma$, the image is subsampled by a factor of about $\sigma/1.6$.

When $s = \sigma/1.6$ is an integer number, it is clear what this means: filter with a Gaussian with parameter $\sigma$, then retain every $s$-th pixel in each dimension. When $s$ is not an integer, on the other hand, sampling "every $s$ pixels" entails retrieving image values between the values available in the image array. This can be done by *sub-pixel interpolation*, which requires a model for the continuous image that the array values are samples of. One of the simplest such models is the *bilinear* one, in which the underlying continuous image $I(\mathbf{x})$ is assumed to be separately linear in $x$ and $y$, the two components of $\mathbf{x}$. This model leads to *bilinear interpolation*: Let $\mathbf{x} = (x, y)$, and

$$\begin{aligned} \xi = \lfloor x \rfloor \quad &, \quad \eta = \lfloor y \rfloor \\ \Delta x = x - \xi \quad &, \quad \Delta y = y - \eta \; . \end{aligned}$$

Then,

$$
\begin{aligned}
I(\mathbf{x}) &= I(\xi,\eta)\,(1-\Delta x)\,(1-\Delta y) \\
&+ I(\xi+1,\eta)\,\Delta x\,(1-\Delta y) \\
&+ I(\xi,\eta+1)\,(1-\Delta x)\,\Delta y \\
&+ I(\xi+1,\eta+1)\,\Delta x\,\Delta y\;.
\end{aligned}
$$

We can now sample the image $I$ with any sampling period, integer or otherwise.

We encapsulate the operations of filtering followed by sampling into a single function

$$B = \text{resize}(I, \phi)$$

where the *downsampling factor* $\phi$ is a positive real number that denotes the ratio between the size of $B$ and that of $I$. For values $0 < \phi < 1$, the image shrinks. For $\phi > 1$, no filtering is performed, and the image grows larger. The filter in downsampling is Gaussian with parameter

$$\sigma = 1.6/\phi\;.$$

Replacing convolution with $S_\sigma$ with resize$(\cdot, \phi)$ where $0 < \phi < 1$ in equation (1) yields the *Gaussian pyramid*:

$$
\begin{aligned}
G_0 &= I \\
G_\ell &= \text{down}(G_{\ell-1}) \quad \text{for} \quad \ell = 1, \ldots, L\;.
\end{aligned}
\tag{2}
$$

In the last expression, we think of fixing $\phi$ to some value between 0 and 1 (for instance, $\phi = 1/2$) and define

$$\text{down}(X) = \text{resize}(X, \phi)\;.$$

We will later also need

$$\text{up}(X) = \text{resize}(X, 1/\phi)$$

where "down" and "up" use the same value of $\phi$. These two operations are called *downsampling* and *upsampling*. There is a crucial difference between the two: Downsampling blurs the input image with a Gaussian and then samples it by bilinear interpolation to make it smaller. Upsampling merely resamples the input image on a finer grid, but it does not undo the blurring. So $X$ and up$(X)$ contain the same frequencies, but the latter is bigger than the former.

Since the image shrinks at each level, it is no longer necessary to specify the maximum level $L$: once the image shrinks to a single pixel (about $\lfloor \log_{1/\phi}(\min(R, C)) \rfloor$ steps, where $R$ and $C$ are the number of rows and columns of $I$), the procedure stops.

Figure 2 shows the Gaussian pyramid for the same image input image as in Figure 1 and for $\phi = 1/2$. The input image has $R = 365$ rows and $C = 384$ columns, and the pyramid has $L = 7$ levels (plus the input image itself).

The Gaussian pyramid is said to be a *lowpass* pyramid, in that every level contains all the image frequencies *below* some value, roughly proportional to $\phi^\ell$. In contrast, the *Laplacian pyramid* is a *bandpass* pyramid, in that every level contains the image frequencies *around* a value that is roughly proportional to $\phi^\ell$.

This bandpass behavior could be achieved by replacing the image at level $\ell$ with the difference $H_\ell$ between $B_{\ell-1}$ and $B_\ell$ in the Gaussian *stack*: The first image contains frequencies below $F\phi^{\ell-1}$ (where

Figure 1: A Gaussian stack.

Figure 2: A Gaussian pyramid.

$F$ measures the highest frequencies in the original image $I$), and the second contains frequencies below $F\phi^\ell$, so their difference $H_\ell$ would contain frequencies between $F\phi^\ell$ and $F\phi^{\ell-1}$, that is, the image detail at frequencies in this band. One would then subsample $B_\ell$ and $B_{\ell+1}$ by a factor $\phi$ and repeat, so that each detail image $H_\ell$ would be smaller than $H(\ell-1)$ by a factor of $\phi$. The result is a pyramid of images $H_\ell$ each of which contains detail around lower and lower frequencies, plus a single, tiny lowpass image $B_{L+1}$ left over after the last subsampling.

How exactly does the information in the highpass pyramid relate to that in the input image $I$? To reconstruct $I$ from $H_1, \ldots, H_L$ and $B_{L+1}$, one would have to *upsample* $B_{L+1}$ and add $H_L$ to the result to obtain $B_L$:

$$B_L = H_L + \text{up}(B_{L+1}) \; .$$

This procedure could be continued until $I = B_0$. However, this would require upsampling to be the exact inverse of the sampling part of downsampling. However, the two sampling operations use bilinear interpolation over two different grids, so this requirement cannot be met precisely.

The solution is the *Laplacian pyramid*, which contains exactly the same information as the input image, and from which the input image can be reconstructed *exactly* (up to numerical rounding). Instead of subtracting two consecutive images from the Gaussian stack, the Laplacian pyramid takes two consecutive images $G_\ell$ and $G_{\ell+1}$ from the Gaussian pyramid, upsamples $G_{\ell+1}$, and makes $H_\ell$ the difference between $G_\ell$ and the upsampled $G_{\ell+1}$:

$$H_\ell = G_\ell - \text{up}(G_{\ell+1}) \; .$$

This simple change guarantees that $G_\ell$ can be reconstructed exactly from $G_{\ell+1}$ and $H_\ell$, since the last equation can be solved for $G_\ell$ to obtain he following:

$$G_\ell = H_\ell + \text{up}(G_{\ell+1}) \; .$$

Here is an algorithm for computing the Laplacian pyramid. First, set

$$L \leftarrow I \; .$$

Then, while $L_0$ is large enough, repeat the following for $\ell = 1, 2, \ldots$:

$$\begin{aligned} L_0 &\leftarrow \text{down}(L) \\ H_\ell &\leftarrow L - \text{up}(L_0) \\ L &\leftarrow L_0 \; . \end{aligned}$$

The remaining image $L$ is the lowpass residual $L_{L+1}$.

Figure 3 shows the Laplacian pyramid for the same image input image as in Figure 1 and for $\phi = 1/2$. To reconstruct the image from the Laplacian pyramid, we work backwards:

$$I \leftarrow L_{L+1}$$

and for $\ell = L, \ldots, 1$:

$$I \leftarrow \text{up}(I) + H_\ell$$

A small *caveat*: because of rounding, up$(I)$ may not yield exactly the desired image size. Because of this, we overload the definition of the up function to take as second argument the desired size of the output image instead of the scaling factor, and the last assignment becomes

$$I \leftarrow \text{up}(I, \text{size}(H_\ell)) + H_\ell \; .$$

Figure 4 shows computation of the Laplacian pyramid and reconstruction of the image from it in schematic form.

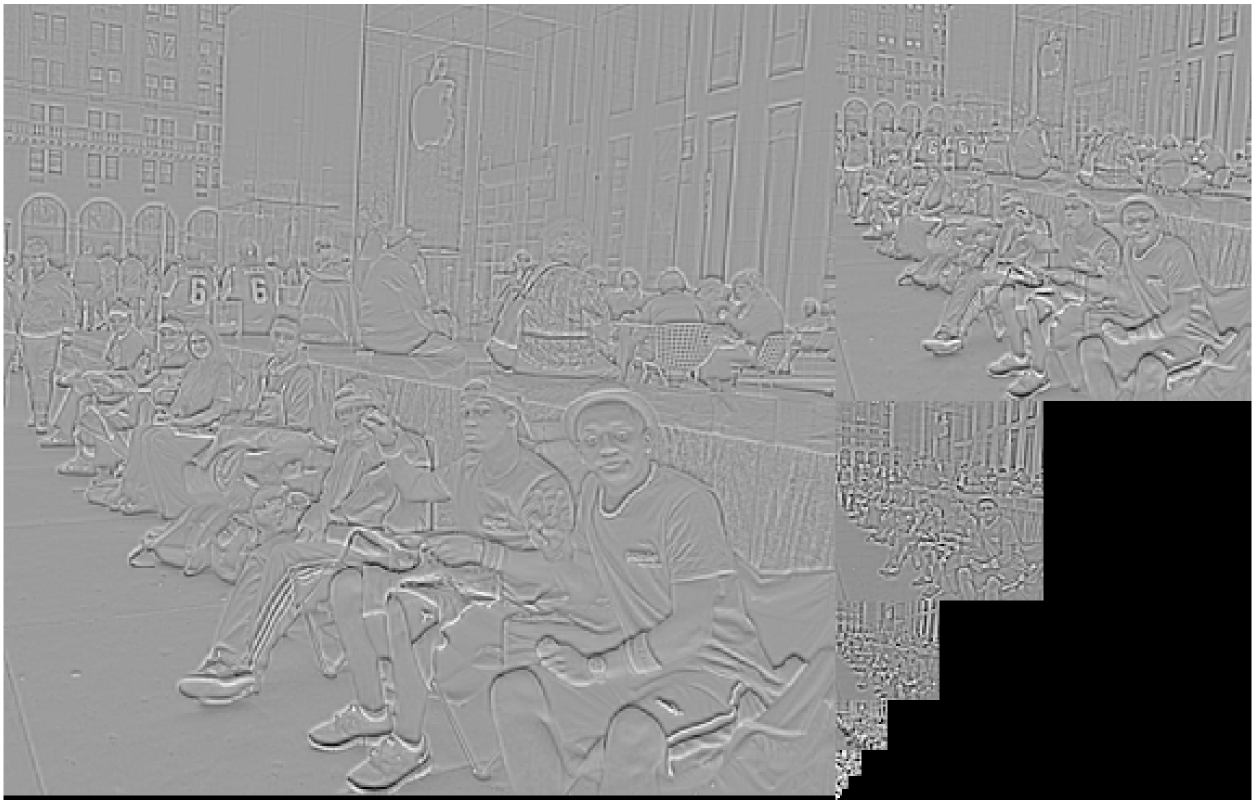

Figure 3: A Laplacian pyramid. Pixel values are positive and negative, and gray denotes zero.

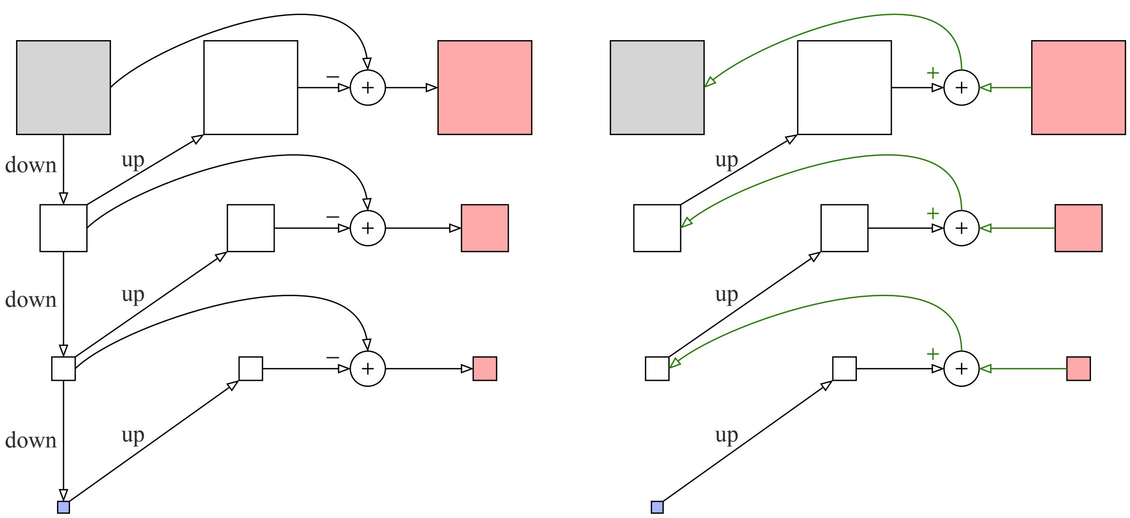


Figure 4: Construction of the Laplacian pyramid (left) and its inverse (right).