THE GAUSSIAN PYRAMID

The first step in Laplacian pyramid coding is to low-pass filter the original image g_0 to obtain image g_1. We say that g_1 is a "reduced" version of g_0 in that both resolution and sample density are decreased. In a similar way we form g_2 as a reduced version of g_1, and so on. Filtering is performed by a procedure equivalent to convolution with one of a family of local, symmetric weighting functions. An important member of this family resembles the Gaussian probability distribution, so the sequence of images g_0, g_1, …, g_n is called the Gaussian pyramid.[1]

A fast algorithm for generating the Gaussian pyramid is given in the next subsection. In the following subsection we show how the same algorithm can be used to "expand" an image array by interpolating values between sample points. This device is used here to help visualize the contents of levels in the Gaussian pyramid, and in the next section to define the Laplacian pyramid.

Gaussian Pyramid Generation

Suppose the image is represented initially by the array g_0 which contains C columns and R rows of pixels. Each pixel represents the light intensity at the corresponding image point by an integer I between 0 and K — 1. This image becomes the bottom or zero level of the Gaussian pyramid. Pyramid level 1 contains image g_1, which is a reduced or low-pass filtered version of g_0. Each value within level 1 is computed as a weighted average of values in level 0 within a 5-by-5 window. Each value within level 2, representing g_2, is then obtained from values within level 1 by applying the same pattern of weights. A graphical representation of this process in one dimension is given in Fig. 1. The size of the weighting function is not critical [2]. We have selected the 5-by-5 pattern because it provides adequate filtering at low computational cost.

The level-to-level averaging process is performed by the function REDUCE.

$$g_k = \text{REDUCE}\,(g_{k-1}) \tag{1}$$

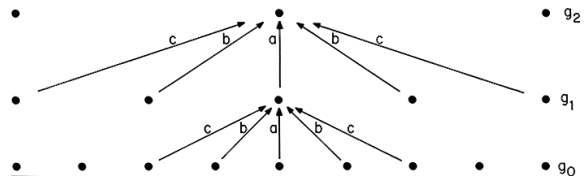
which means, for levels $0 < l < N$ and nodes i, j, $0 \leq i < C_l$, $0 \leq j < R_l$,

$$g_l(i,j) = \sum_{m=-2}^{2} \sum_{n=-2}^{2} w(m,n)\, g_{l-1}(2i+m, 2j+n).$$

Here N refers to the number of levels in the pyramid, while C_l and R_l are the dimensions of the lth level: Note in Fig. 1 that the density of nodes is reduced by half in one dimension, or by a fourth in two dimensions from level to level. The dimensions of the original image are appropriate for pyramid construction if integers M_C, M_R, and N exist such that $C = M_C 2^N + 1$ and $R = M_R\, 2^N + 1$. (For example, if M_C and M_R are both 3 and N is 5, then images measure 97 by 97 pixels.) The dimensions of g_l are $C_l = M_C\, 2^{N-l} + 1$ and $R_l = M_R\, 2^{N-l} + 1$.

[1] We will refer to this set of low-pass filtered images as the Gaussian pyramid, even though in some cases it will be generated with a trimodal rather than unimodal weighting function.

GAUSSIAN PYRAMID



g_0 = IMAGE

g_L = REDUCE [g_{L-1}]

Fig 1. A one-dimensional graphic representation of the process which generates a Gaussian pyramid Each row of dots represents nodes within a level of the pyramid. The value of each node in the zero level is just the gray level of a corresponding image pixel. The value of each node in a high level is the weighted average of node values in the next lower level. Note that node spacing doubles from level to level, while the same weighting pattern or "generating kernel" is used to generate all levels.

The Generating Kernel

Note that the same 5-by-5 pattern of weights w is used to generate each pyramid array from its predecessor. This weighting pattern, called the generating kernel, is chosen subject to certain constraints [2]. For simpticity we make w separable:

$$w(m,n) = \hat{w}(m)\ \hat{w}(n).$$

The one-dimensional, length 5, function $\hat{w}$ is normalized

$$\sum_{m=-2}^{2} \hat{w}\,(m) = 1$$

and symmetric

$$\hat{w}(i) = \hat{w}(-i) \text{ for } i = 0, 1, 2.$$

An additional constraint is called equal contribution. This stipulates that all nodes at a given level must contribute the same total weight (=1/4) to nodes at the next higher level. Let $\hat{w}(0) = a$, $\hat{w}\,(-1) = \hat{w}\,(1) = b$, and $\hat{w}(-2) = \hat{w}(2) = c$ in this case equal contribution requires that $a + 2c = 2b$. These three constraints are satisfied when

$$\hat{w}(0) = a$$

$$\hat{w}(-1) = \hat{w}(1) = 1/4$$

$$\hat{w}(-2) = \hat{w}(2) = 1/4 - a/2.$$

Equivalent Weighting Functions

Iterative pyramid generation is equivalent to convolving the image g_0 with a set of "equivalent weighting functions" h_l:

$$g_l = h_l \oplus g_0$$

or

$$g_l(i,j) = \sum_{m=-M_l}^{M_l} \sum_{n=-M_l}^{M_l} h_l(m,n)\, g_0(i2^l + m \bullet j2^l + n).$$

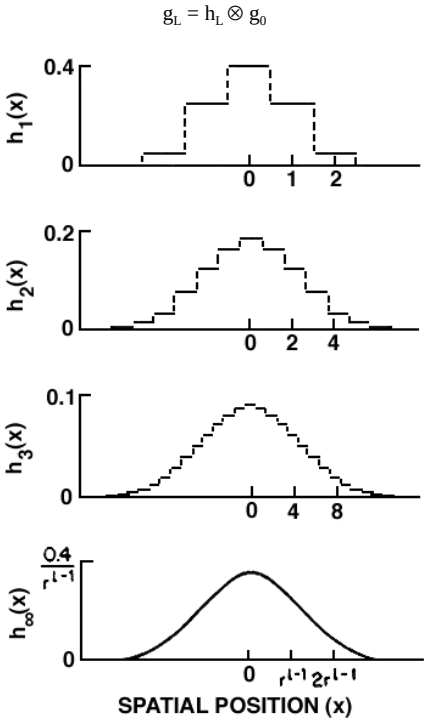


Fig. 2. The equivalent weighting functions $h_l(x)$ for nodes in levels 1, 2, 3, and infinity of the Gaussian pyramid. Note that axis scales have been adjusted by factors of 2 to aid comparison Here the parameter a of the generating kernel is 0.4, and the resulting equivalent weighting functions closely resemble the Gaussian probability density functions.

The size M_l of the equivalent weighting function doubles from one level to the next, as does the distance between samples.

Equivalent weighting functions for Gaussian-pyramid levels 1, 2, and 3 are shown in Fig. 2. In this case $a = 0.4$. The shape of the equivalent function converges rapidly to a characteristic form with successively higher levels of the pyramid, so that only its scale changes. However, this shape does depend on the choice of a in the generating kernel. Characteristic shapes for four choices of a are shown in Fig. 3. Note that the equivalent weighting functions are particularly Gaussian-like when $a = 0.4$ When $a = 0.5$ the shape is triangular; when $a = 0.3$ it is flatter and broader than a Gaussian. With $a = 0.6$ the central positive mode is sharply peaked, and is flanked by small negative lobes.

Fast Filter

The effect of convolving an image with one of the equivalent weighting functions h_l is to blur, or low-pass filter, the image. The pyramid algorithm reduces the filter band limit by an octave from level to level, and reduces the sample interval by the same factor. This is a very fast algorithm, requiring fewer computational steps to compute a set of filtered images than are required by the fast Fourier transform to compute a single filtered image [2].

Example: Fig. 4 illustrates the contents of a Gaussian pyramid generated with a = 0.4. The original image, on the far left, measures 257 by 257. This becomes level 0 on the pyramid. Each higher level array is roughly half as large in each dimension as its predecessor, due to reduced sample density.

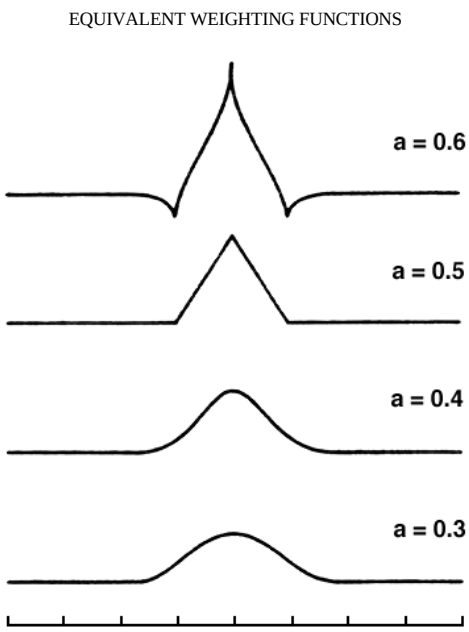


Fig. 3. The shape of the equivalent weighting function depends on the choice of parameter a. For $a = 0.5$, the function is triangular; for $a = 0.4$ it is Gaussian-like, and for $a = 0.3$ it is broader than Gaussian. For $a = 0.6$ the function is trimodal.

Gaussian Pyramid Interpolation

We now define a function EXPAND as the reverse of REDUCE. Its effect is to expand an $(M + 1)$-by-$(N + 1)$ array into a $(2M + 1)$-by-$(2N + 1)$ array by interpolating new node values between the given values. Thus, EXPAND applied to array g_l of the Gaussian pyramid would yield an array $g_{l,1}$ which is the same size as g_{l-1}.

Let $g_{l,n}$ be the result of expanding g_l n times. Then

$$g_{l,0} = g_l$$

and

$$g_{l,n} = \text{EXPAND}\,(g_l, n\text{—}1)$$

By EXPAND we mean, for levels $0 < l \leq N$ and $0 \leq n$ and nodes i, j, $0 \leq i < C_{l-n}$, $0 \leq j < R_{l-n,}$

$$g_{l,n}(ij) = 4 \sum_{m=-2}^{2} \sum_{n=-2}^{2} w(m,n) \cdot g_{l,n-1}\left(\frac{i-m}{2}, \frac{j-n}{2}\right). \quad (2)$$

Only terms for which $(i\text{—}m)/2$ and $(j\text{—}n)/2$ are integers are included in this sum.

If we apply EXPAND l times to image g_l, we obtain $g_{l,l}$, which is the same size as the original image g_0, Although full expansion will not be used in image coding, we will use it to help visualize the contents of various arrays within pyramid structures. The top row of Fig. 5 shows image $g_{0,0}$, $g_{1,1}$, $g_{2,2}$, … obtained by expanding levels of the pyramid in Fig. 4. The low-pass filter effect of the Gaussian pyramid is now shown clearly.

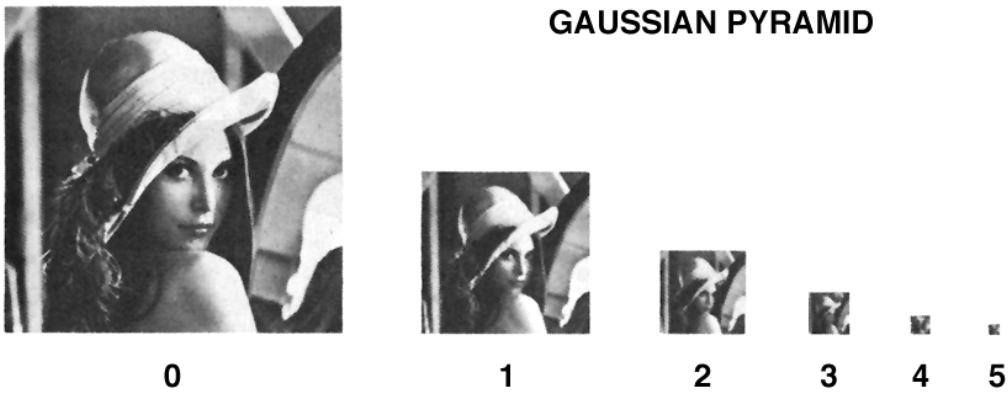


Fig. 4. First six levels of the Gaussian pyramid for the "Lady" image The original image, level 0, meusures 257 by 257 pixels and each higher level array is roughly half the dimensdons of its predecessor. Thus, level 5 measures just 9 by 9 pixels.

THE LAPLACIAN PYRAMID

Recall that our purpose for constructing the reduced image g_1 is that it may serve as a prediction for pixel values in the original image g_0. To obtain a compressed representation, we encode the error image which remains when an expanded g_1 is subtracted from g_0. This image becomes the bottom level of the Laplacian pyramid. The next level is generated by encoding g_1 in the same way. We now give a formal definition for the Laplacian pyramid, and examine its properties.

Laplacian Pyramid Generation

The Laplacian pyramid is a sequence of error images L_0, L_1, …, L_N. Each is the difference between two levels of the Gaussian pyramid. Thus, for $0 \leq 1 < N$,

$$L_l = g_l - \text{EXPAND}(g_{l+1})$$

$$= g_l - g_{l+1.1}.$$

Since there is no image g_{N+1} to serve as the prediction image for g_N, we say $L_N = g_N$.

Equivalent Weighting Functions

The value at each node in the Laplacian pyramid is the difference between the convolutions of two equivalent weighting functions h_l, h_{l+1} with the original image. Again, this is similar to convolving an appropriately scaled Laplacian weighting function with the image. The node value could have been obtained directly by applying this operator, although at considerably greater computational cost.

Just as we may view the Gaussian pyramid as a set of low-pass filtered copies of the original image, we may view the Laplacian pyramid as a set of bandpass filtered copies of the image. The scale of the Laplacian operator doubles from level to level of the pyramid, while the center frequency of the passband is reduced by an octave.

In order to illustrate the contents of the Laplacian pyramid, it is helpful to interpolate between sample points. This may be done within the pyramid structure by Gaussian interpolation. Let $L_{l,n}$ be the result of expanding L_l n times using (2). Then, $L_{l,l}$ is the size of the original image.

The expanded Laplacian pyramid levels for the “Lady” image of Fig. 4 are shown in the bottom row of Fig. 5. Note that image features such as edges and bars appear enhanced in the Laplacian pyramid. Enhanced features are segregated by size: fine details are prominent in $L_{0,0}$, while progressively coarser features are prominent in the higher level images.

Decoding

It can be shown that the original image can be recovered exactly by expanding, then summing all the levels of the Laplacian pyramid:

$$g_0 = \sum_{l=0}^{N} L_{l,l}. \tag{4}$$

A more efficient procedure is to expand L_N once and add it to L_{N-1}, then expand this image once and add it to L_{N-2}, and so on until level 0 is reached and g_0 is recovered. This procedure simply reverses the steps in Laplacian pyramid generation. From (3) we see that

$$g_N = L_N$$

and for $l = N-1, N-2, \ldots, 0$,

$$g_l = L_l + \text{EXPAND}(g_{l+1}).$$

Entropy

If we assume that the pixel values of an image representation are statistically independent, then the minimum number of bits per pixel required to exactly encode the image is given by the entropy of the pixel value distribution. This optimum may be approached in practice through techniques such as variable length coding.

The histogram of pixel values for the "Lady" image is shown in Fig. 6(a). if we let the observed frequency of occurrence $f(i)$ of each gray level i be an estimate of its probability of occurrence in this and other similar images, then the entropy