Reprinted from *Proceedings of the Pattern Recognition and Information Processing Conference* (pp. 218-223), Dallas, TX (1981).

IMAGE DATA COMPRESSION WITH THE LAPLACIAN PYRAMID*

Edward H. Adelson
Psychology Dept., New York University
New York, New York 10003

Peter J. Burt
Electrical, Computer, and Systems Engineering Dept.
Rensselaer Polytechnic Institute
Troy, New York 12181

Abstract

We describe a new technique for image encoding in which Gaussian-like operators serve as the basis functions. The representation differs from established techniques in that the Gaussian code elements are localized in both space and spatial frequency.

Pixel to pixel correlations are first removed by subtracting a low-pass filtered copy of the image from the image itself. The result is a net data compression since the difference, or error, image has low variance, and the low-pass filtered image may be represented at reduced sample density. Further data compression is achieved by quantizing the difference image and repeating the encoding process for the low-pass filtered image.

The encoding process is equivalent to sampling the image with Laplacian operators of many scales. Thus the code tends to enhance salient image features. A primary advantage of the present code is that it is well suited for many image analysis tasks as well as for data compression. Fast algorithms are described for coding and decoding.

I. Introduction

A common characteristic of images is that neighboring pixels are highly correlated. To represent the image directly in terms of the pixel value is therefore inefficient: most of the encoded information is redundant.

The first task in designing an efficient, compressed code is to find a representation which, in effect, decorrelates the image pixels. This has been achieved through predictive and through transform techniques.

In predictive coding pixels are encoded sequentially in a raster format. However, prior to encoding each pixel, its value is predicted from previously coded pixels in the same and preceding raster lines. The predicted pixel value, which represents redundant information, is subtracted from the actual pixel value, and only the difference, or prediction error, is encoded. Since only previously encoded pixels are used in predicting each pixel's value, this process is said to be causal. Restriction to causal prediction facilitates decoding: to decode a given pixel its predicted value is recomputed from already decoded neighboring pixels, and added to the stored prediction error.

Non-causal prediction, based on a symmetric neighborhood centered at each pixel should yield more accurate prediction, and hence greater data compression. However this approach does not permit simple sequential decoding. Non-causal approaches to image coding typically involve image transforms, or the solution to large sets of simultaneous equations. Rather than encode pixels sequentially, they are encoded all at once, or by blocks .

Both predictive and transform techniques have advantages. The former is relatively simple to implement and readily adapted to local image characteristics. The latter generally provides greater data compression, but at the expense of greater computation.

Here we shall describe a new technique for removing image correlation which is intermediate between the predictive and transform methods. The technique is non-causal, yet computations are relatively simple.

The predicted value for each pixel is computed as a local weighted average, using a symmetric. unimodal, weighting function centered on the pixel itself. The predicted values for all pixels are first obtained by convolving this Gaussian-like weighting function with the image. The result is a low-pass filtered image which is then subtracted from the original.

Let $g_0(ij)$ be the original image, and $g_1(ij)$ be the result of applying an appro-priate low-pass filter to g_0. The prediction error $L_0(ij)$, is then given by

$$L_0(i3) = g_0(ij) - g_1(ij)$$

Rather than encode g_0 we encode L_0 and g_1. This results in a net data compression because (a) L_0 is largely decorrelated, so may be represented

* The support of the National Science Foundation under Grant MC5-79-23422 is gratefully acknowledged. The first author was supported by NIH Postdoctoral Training Grant EY07003.

pixel by pixel with many fewer bits than g_0, and (b) g_1 is low-pass filtered so may be encoded at a reduced sample rate.

Further data compression is achieved by iterating this process. The reduced image, g_1, is itself low-pass filtered to yield g_2 and a second error image is obtained: $L_2(ij) = g_1(ij) - g_2(ij)$. By repeating these steps several times we obtain a sequence of two-dimensional arrays L_0, L_1, L_2..L_N, each of which is smaller than its predecessor (by a factor of l/4). If we now imagine these arrays stacked one above another the result is a tapering pyramid data structure. The value at each node in the pyramid represents the difference between two Gaussian-like functions convolved with the original image. The difference of Gaussian (or DOG) is equivalent to the so-called Laplacian operators commonly used in image enhancement (Rosenfeld and Kak, 1976). Thus we refer to the proposed compressed image representation as the Laplacian-pyramid code.

The coding scheme outlined above will be practical only if required filtering computations can be performed with an efficient algorithm. A suitable fast algorithm has recently been developed (Burt, 1981) and will be described in the next section.

Two additional characteristics of the Laplacian-pyramid code may give it an advantage over other encoding systems. First, the code is very similar to image representation in the human visual system. Thus additional compression may be achieved by quantizing the code elements, with quantization levels directly matched to perceptual character-istics of human observers. Second, and perhaps more important, is the fact that the Laplacian- pyramid code tends to enhance salient image features. Laplace operators are used in computer image analysis to detect simple features such as edges. Image representations based on convolution with Laplacians have been proposed for a wide variety of basic analysis tasks including texture analysis, motion, and stereopsis (Marr and Poggio, 1979; Pietikainen, 1980). Thus the Laplacian-pyramid code provides not only a compressed representation but one which is appropriate for computer image understanding.

II. 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 g_1 is a "reduced" version of g_0 in that both reso-lution and sample density are decreased. In a similar way we form g_3 as a reduced version of g_2, and so on. Filtering is performed by convolution with a Gaussian-like weighting function, so the sequence of images g_0, g_1, .. , g_N is called the Gaussian pyramid.

A fast algorithm for generating the Gaussian pyramid is given in the next sub-section. 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.

A) Gaussian Pyramid Generation

Suppose the image is represented initially by the array go which contains C columns and R rows of pixels. Each pixel represents the light intensity at the corre-sponding 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 l 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 O within a 5 by 5 window. Each value within level 2, representing g_2, is then obtained from values within level l by applying the same pattern of weights. A graphical representation of this process in one dimension is given in Figure l,

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

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

which means

for levels $0 < l \leq N$

and nodes i,j $0 \leq i < C_l$, $O \leq j < R_l$

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

Here N refers to the number of levels in the pyramid, while C_l and R_l are the dimensions of the l^{th} level. Note in Figure l that the density of nodes is reduced by half in one dimension, or by a fourth in two dimen-sions from level to level. The dimensions of the original image are appropriate for pyra-mid 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 $R_l = M_R 2^{N-1} + 1$.

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 cer-tain constraints. For simplicity we make w separable:

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

The one-dimensional, length 5, function w is normalized

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

and symmetric

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

An additional constraint is called equal contribution (Burt l981). Let $\hat{w}(0) = a$, $\hat{w}(-1) = \hat{w}(l) = b$ and $\hat{w}(-2) = \hat{w}(2) = c$. In this case equal contribution requires that $a + 2b = 2c$. These three constraints are satisfied when

$\hat{w}(0) = a$
$\hat{w}(-1) = \hat{w}(1) = (1/4)a$
$\hat{w}(-2) = \hat{w}(2) = 1/4 - a/2$

Equivalent weighting functions. For certain choices of the generating kernel, i.e., for certain a, the process of iterative pyramid generation described above is equi-valent to convolving the image with Gaussian-like weighting functions. That is, the value of any node could have been obtained directly (although at considerably greater compu-tational cost) by convolving the image with a Gaussian-like weighting function centered at the node. The size 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 Figure 2. In this case a = 0.4. The shape of the equivalent function converges rapidly to a characteristic form with successively higher planes 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 three choices of a are shown in Figure 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.

Fast filter. The effect of convolving an image with a Gaussian-like function is to blur, or low-pass filter, the image. The py-ramid algorithm reduces the filter band limit by one octave from level to level. The sample interval is also reduced by this factor, but remains below the Nyquist limit. This is a fast algorithm, requiring fewer computational steps to compute a set of filtered images than are required by the fast Fourier trans-form to compute a single filtered image (Burt, 1981).

B. 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 be-tween 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}. If EXPAND is applied l times, g_l is ex-panded to the size of the original image.

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

$$g_{l,0} = g_l \qquad (2)$$

and

$$g_{l,n} = \text{EXPAND}(g_{l,n-l})$$

By EXPAND we mean

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

$$g_{l,n}(ij) = 4 \sum_{m=-2}^{2} \sum_{n=-2}^{2} w(m,n) g_{l,n-1}((i-m)/2,(j-n/2)).$$

Only terms for which (i - m)/2 and (j - n)/2 are integers are included in this sum.

Gaussian pyramid filtering and interpolation are illustrated in Figure 4. In this example the original image" is a one-dimensional step function of width 129 samples, as shown in Figure 4a. Sample values of g are shown as dots in Figure 4b. The expanded function $g_{4,4}$, shown in 4b as a dark curve, gives a graphic representation of the contents of g_4, clearly showing the effects of low-pass filtering. The function $g_{4,4}$ was obtained by repeated interpolation between sample points, using EXPAND. The same result could have been obtained if the equivalent weighting functions at level 4 were scaled by the sample values and then summed. Appro-priately scaled equivalent weighting func-tions are shown in Figure 4b as light curves centered at the level 4 sample points.

III. 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 representa-tion 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 genera-ted by encoding g_1 in the same way. We now give a formal definition for the Laplacian pyramid, and examine its properties.

A) Laplacian Pyramid Generation

The Laplacian pyramid is a sequence of error images, $L_0, L_1, \ldots, L_N$. Each is the difference between two levels of the Gaussian pyramid. Thus:

$$\text{for } 0 \leq l < N \qquad (3)$$

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

$$= g_l - g_{l+1,1}$$

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 Gaussian-like functions with the original image. Again, this is equivalent 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 band-pass 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 pass band is reduced by an octave. These points are illustrated in Figure 5, which shows the equivalent Laplacian weighting functions at three successive levels of the pyramid and their Fourier transforms.

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 Fig. 2. Then $L_{l,l}$ is the size of the original image. $L_{4,4}$ for the one-dimensional step image of Figure 4a is shown in Figure 4c.

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

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

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 O is reached and g_0 is recovered. This procedure simply reverses the steps in Laplacian pyramid generation. From Eq. 3 we see that:

$$g_N = L_N \tag{4}$$

and for l = N - 1, N- 2, .. , O

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

B) Quantizing the Laplacian Code

Our objective in constructing the Laplacian pyramid has been to obtain image data compression by removing image correlations. Substantial further compression may be obtained by quantizing the pyramid node values. Such quantization inevitably intro-duced distortion in the reconstructed image. However, with judicious design this distortion may not be disturbing to human observers.

It is known, for example, that humans are more sensitive to gray level fluctuations in low-frequency components of an image than to high-frequency components (Carlson and Cohen, 1978; Kretzmer, 1956). Thus we may choose to allocate fewer quantization levels to L_0 than to other pyramid levels. It is also known that humans are less sensitive to a given fluctuation in gray level when it occurs in the neighborhood of a prominent image feature than when it occurs in isolation (Netravali and Prasada, 1977). Advantage may be taken of this perceptual limit-ation by adjusting the quantization levels adaptively over the image.

In the examples we shall show, three quantization levels are used for level zero and five for the other levels. While adaptive techniques can be efficiently incorporated in the pyramid structure, we have not done so in these examples.

Let $C_l(ij)$ be the result of quantizing $L_l(ij)$. We have adopted the following simple (and non-optimal) quantization policy:

$$c_0(ij) = \begin{cases} +A \text{ if } L_0(ij) \geq +T \\ 0 \text{ if } -T < L_0(ij) \leq +T \\ -A \text{ if } L_0(ij) \leq -T. \end{cases} \tag{5}$$

At level l, 0 < l < N

$$C_l(ij) = \begin{cases} 2A & \text{if } 2T \leq L_l(ij) \\ A & \text{if } 2T/3 \leq L_l(ij) < 2T \\ 0 & \text{if } -2T/3 < L_l(ij) < 2T/3 \\ -A & \text{if } -2T < L_l(ij) \leq -2T/3 \\ -2A & \text{if } L_l(ij) \leq 2T \end{cases}$$

Nodes at level N are always positive. For simplicity we say $C_N(ij) = L_N(ij)$.

Summary of the coding and decoding procedures. Figure 6 is a flow diagram for Laplacian pyramid coding. The first step, shown on the far left, is bottom up con-struction of the Gaussian pyramid images g_0, g_1, …, g_N (Eq. 1). The Laplacian pyramid images L_0, L_1, …, L_N are then obtained as the difference between successive Gaussian levels (Eq. 3). These are quantized to yield the compressed code represented by the pyramid of values $C_l(ij)$ (Eq. 5). Finally image reconstruction follows expand and sum procedure (Eq. 4) using C values in the place of L values. Here we designate the reconstructed image by r_0.

IV. Experimental Results and Discussion

Figure 7 illustrates the Laplacian pyramid operation. The original image (lower left) measures 97 by 97 pixels, and is represented by 64 gray levels.

The bottom row of Figure 7 shows the image blurred at successively higher levels of the Gaussian pyramid. Each level has been expanded to 97 by 97 pixels. The top row shows the corresponding contents of the Laplacian pyramid. Again each level of the Laplacian is the difference between levels of the Gaussian. Conversely, each Gaussian level is equal to the sum of all Laplacian images at the same level and above. Thus, reading the bottom row from right to left one can see the images sharpen as finer and finer Laplacian images are added to the sum, until finally the original image is reconstructed. Figure 7 involves no quantization, so the reconstruction is perfect.

The top row of Figure 8 shows the effects of quantization (Eq. 5) on the Laplacian pyramid. The bottom row of Figure 8 shows the result of image reconstruction from the quantized Laplacian pyramid code. A careful comparison of Figures 7 and 8 will reveal some changes in contrast. However no disturbing artifacts have been introduced by quantization which might interfere with human perception.

We may now compute the image data compression obtained through quantization alone. This may be specified as the number of bits of information needed to represent the com-pressed code for each pixel in the original image. Suppose there are P pixels in the image. There will then be P nodes in the zero level array of the C pyramid. Level 1 will have P/4 nodes, level 2 will have P/16, and so on. The total number of nodes excluding level

zero is P/3. For each node in level 0 we need to specify one of three gray levels. Assuming all values are equally likely we require at least $\log_2 3$ bits. Nodes in higher levels require $\log_2 5$ bits, since they may assume any of five values. The total num-ber of bits per pixel is given by $(P \log_2 3 + P/3 \log_2 5)/P$ or 2.36 bits per pixel. This figure does not include substantial further reductions which may be obtained with variable length Huffman coding and adaptive techniques. Nodes in higher levels require $\log_2 5$.

It should be observed that the Laplacian pyramid code is particularly well suited for progressive image transmission. In this type of transmission a coarse rendition of the image is sent first to give the receiver an early impression of image content, then subsequent transmission provides image details of progressively finer resolution (Knowlton, 1980). The observer may terminate transmission of an image as soon as its contents are recognized, or as soon as it becomes evident that the image will not be of interest. To achieve progressive trans-mission, the top-most level of the pyramid code is sent first, and expanded in the receiving pyramid to form an initial very coarse image. The next lower level is then transmitted, expanded, and added to the first. and so on. At the receiving end, the initial image appears very blurry, but then comes steadily into "focus." This progression is illustrated in the lower row of Figure 8, from right to left. Note that while 2.4 bits are required for each pixel of the full transmission (left-most bottom image, Figure 8) less than a third of these, or 0.77 bits, are needed for each pixel of the previous image (second from left, Figure 8), and 0.19 bits for the image previous to that (third from left).

Finally, it should be observed that the Laplacian pyramid encoding scheme requires relatively simple computations. The computations are local and may be performed in parallel, and the same computations are iterated to build each pyramid level from its predecessor. We may envision performing Laplacian coding and decoding in real time using array processors and a pipeline architecture.

An additional benefit stressed in the introduction is that in computing the Laplacian pyramid, one automatically has access to bandpass copies of the image. In this representation image features of various sizes are enhanced and are directly available to image processing and pattern recognition.

References

1. Burt, P.J., Fast Filter Transforms for Image Processing. To appear in CGIP (1981).

2. Carlson, C.R., and R.W. Cohen, Visibility of Displayed Information, Technical Report to the Office of Naval Research, Contract No. N00014-74-C-0184, July 1978.

3. Knowlton, K., Progressive Transmission of Gray-scale and Binary Pictures by Simple, Efficient, and Lossless Encoding Schemes. Proc. IEEE 68, 885-896 (1980).

4. Kretzmer, E.R., Reduced-alphabet Representation of Television Signals, IRE National Conv. Rec. PT4 p. 140 (1956).

5. Marr, D., and T. Poggio, A Computational Theory of Human Stereo Vision, Proc. Roy. Soc. Lond. B. 204 301-328 (1979).

6. Netravali, A.N., and B. Prasada, Adaptive Quantization of Picture Signals Using Spatial Masking. Proc. IEEE 65 53G-548 (1977).

7. Pietikainen, M., On the Use of Hierarchicallv Computed "Mexican Hat" Features for Texture Discrimination. Tech. Rept. - 968, Computer Science Center, University of Maryland (1980).

8. Rosenfeld, A., and Kak, A., Digital Picture Processing, Academic Press, New York, 1976.

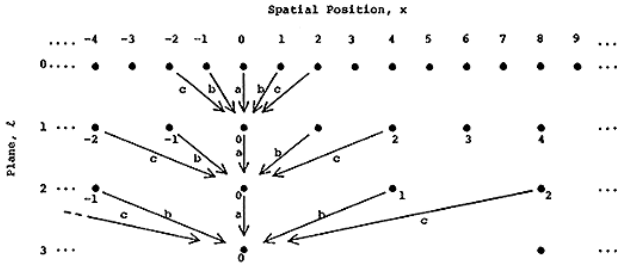

Fig. 1. Graphical representation of Gaussian pyramid construction.

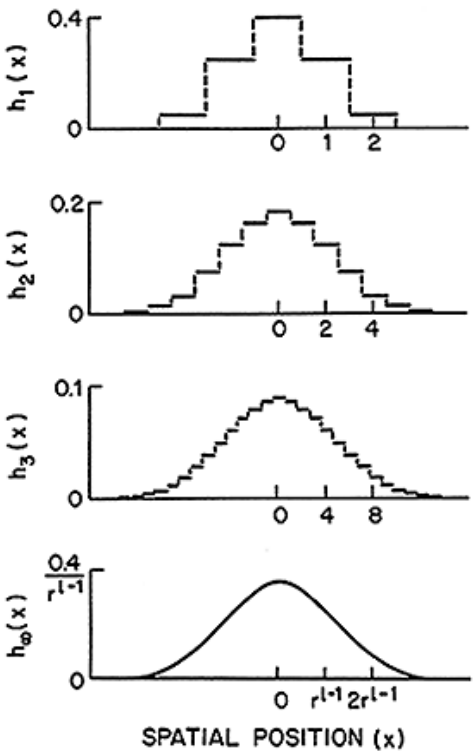

Fig. 2. Equivalent weighting functions at levels 1, 2 and 3. These converge to the function shown below.

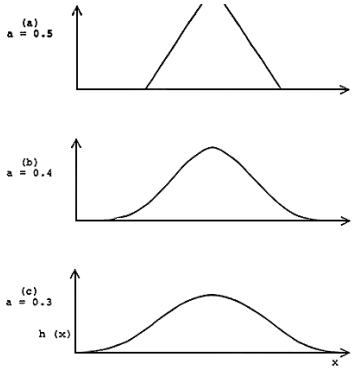

Fig. 3. Equivalent weighting functions for a = 0.5, 0.4 and 0.3.

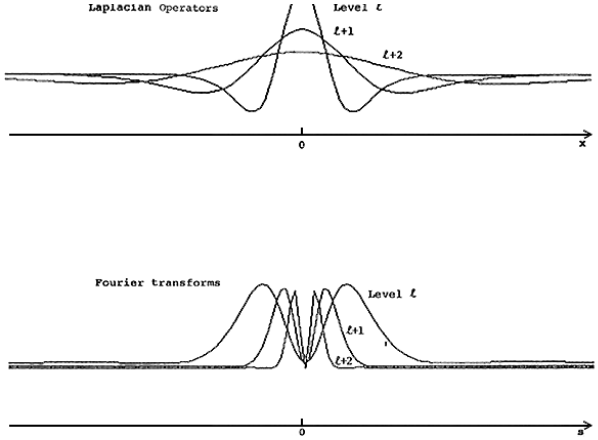

Fig. 4. Laplacian weighting functions and their Fourier transforms.

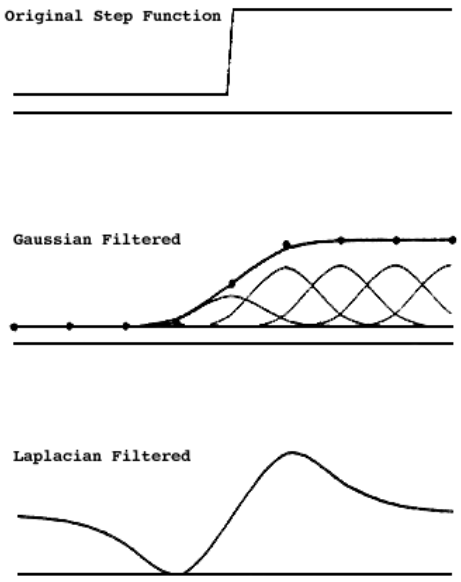

Fig. 5. One-dimensional step function and its level 4 Gaussian and Laplacian representations.

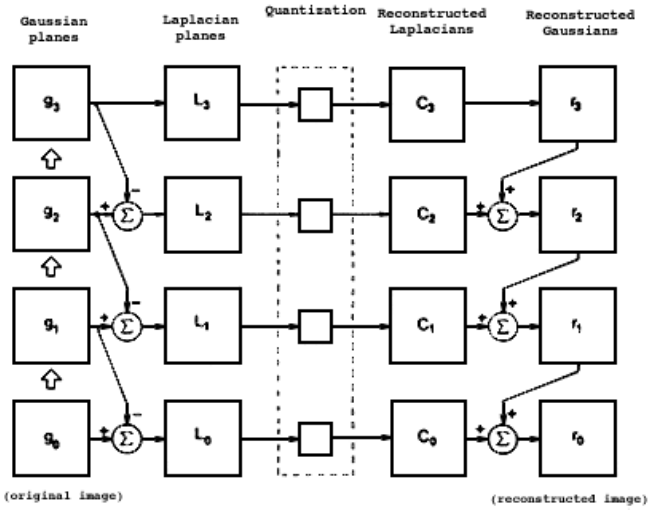

Fig. 6. Flow diagram for Laplacian coding.

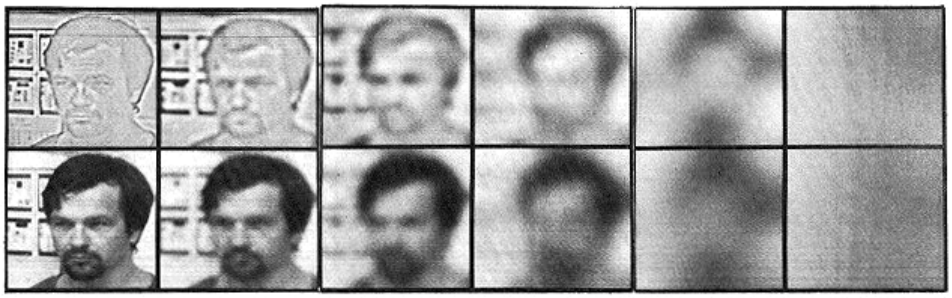

Fig. 7. Laplacian and Gaussian images before quantization

Fig. 8. As in Fig. 7, after quantization.