Chapter 3

Image Pyramids

In chapter XX we motivated translation invariant linear filters as a way of accounting for the fact that objects in images might appear at any location. Therefore, a reasonable way of processing an image is by manipulating pixel neighborhoods in the same way independently on the image location. In addition to translation invariance, scale invariance is another fundamental property of images. Due to perspective projection, objects at different distances will appear with different sizes as shown in figure 3.1. Therefore, if we want to locate all the bird instances in this image, we will have to apply an operator that is invariant in translation and in scale. Image pyramids provide an efficient representation for space-scale invariant processing.

3.1 Image Pyramids and multi-scale image analysis

Image information occurs over many different spatial scales. Image pyramids–multi-resolution representations for images–are a useful data structure for analyzing and manipulating images over a range of spatial scales. Here we'll discuss three different ones, in a progression of complexity. The first is a Gaussian pyramid, which creates versions of the input image at multiple resolutions. This is useful for analysis across different spatial scales, but doesn't separate the image into different frequency bands. The Laplacian pyramid provides that extra level of analysis, breaking the image into different isotropic spatial frequency bands. The Steerable pyramid provides a clean separation of the image into different scales and orientations. There are various other differences between these pyramids, which we'll describe below.

As a motivating example, lets assume we want to detect the birds from figure **??** using the normalized correlation approach described in section XX. If we have a template of a bird, the normalized correlation will be able to detect only the birds that have a similar image size than the template. To introduce scale invariance, one possible solution is to change the size of the template to cover a wide range of possible sizes and apply them to the image. Then, the ensemble of templates will be able to detect birds of different sizes. The disadvantage of this approach is that it will be computationally expensive as detecting large birds will require computing convolutions with big kernels which is very slow. Another alternative is to change the image size as shown in figure 3.1 resulting in a *multiscale image pyramid*. In this example, the original image has a resolution of 848×643 pixels. Each image in the pyramid is obtained by scaling down the image from the previous level by reducing the number of pixels by factor of 25%. This operation is called downsampling and we will study it in detail in this chapter. Now we can use the pyramid to detect birds at different sizes using a single template. The red box in the figure denotes the size of the template used. The figure shows how birds of different sizes become detectable at, at least, one of the levels of the pyramid. This method will be more efficient as the template can be kept small and the convolutions will remain computationally efficient.

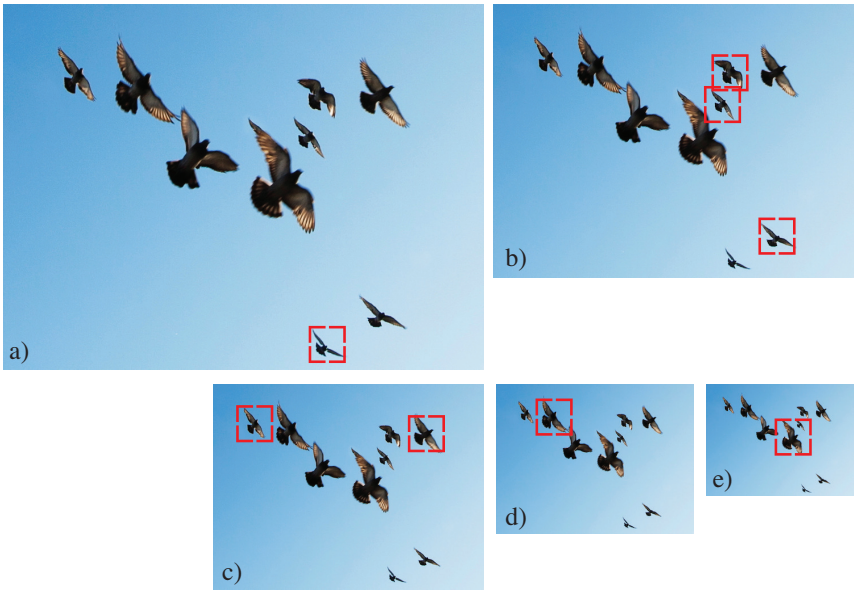


Figure 3.1: Multiscale image pyramid. Each image is 25% smaller than the previous one. The red box indicates the size of a template used for detecting flying birds. As the size of the template is fixed, it will only be able to detect the birds that tightly fit inside the box. Birds that are smaller or larger will not be detected within a single scale. By running the same template across many levels in this pyramid, different birds instances are detected at different scales.

Mutiscale image processing and image pyramids have many applications beyond scale invariant object detection. In this chapter we will describe some important image pyramids and their applications.

3.2 Linear image transforms

Let's first look at some general properties of linear image transforms. For an input image x of N pixels, a linear transform is:

$$r = P^T x \tag{3.1}$$

where r is a vector of dimensionality M, and P is a matrix of size $N \times M$. The columns of $P = [P_0, P_1, ..., P_{M-1}]$ are the projection vectors. The vector r contains the transform coefficients: $r_i = P_i^T x$. The vector r corresponds to a different representation of the image x than the original pixel space.

The transform P is said to be critically sampled when $M = N$. The transform is over-sampled when $M > N$, and under-sampled when $M < N$. We are interested in transforms that are invertible, so that we can recover the input x from the projection coefficients r:

$$x = Br = \sum_{i=0}^{M-1} r_i B_i \tag{3.2}$$

The columns of $B = [B_0, B_1, ..., B_{M-1}]$ are the basis vectors. The input signal x can be reconstructed as a linear combination of the basis vectors B_i weighted by the representation coefficients r_i.

The transform P is complete, encoding all image structure, if it is invertible. If critically sampled (i.e., $M = N$) and the transform is complete, then $B = (P^T)^{-1}$. If it is over-complete (over-sampled and complete), then the inverse can be obtained using the pseudo inverse $B = (PP^T)^{-1}P$.

An important special case is when the transform is self-inverting, then $PP^T = I$. The values of P can be real or complex (like in the Fourier transform). For complex transforms, we should replace the P^T by P^{*T} (complex conjugate transpose).

3.3 Gaussian pyramid

We'd like to make a recursive algorithm for creating a multi-resolution version of an image. A gaussian filter is a natural one to use to blur out an image, since multiple applications of a gaussian filter is equivalent to application of a single, wider gaussian filter.

Here's an elegant, efficient algorithm for making a resolution reduced version of an input image. It involves two steps: convolving the image with a low-pass filter (for example, using the 4-th binomial filter $b_4 = [1, 4, 6, 4, 1]$ / 16, normalized to sum to 1, separably in each dimension), and then subsampling by a factor of 2 the result. Each level is obtained by filtering the previous level with the 4-th binomial filter with a stride of 2 (on each dimension). Applied recursively, this algorithm generates a sequence of images, subsequent ones being smaller, lower resolution versions of the earlier ones in the processing.

One **block** of the Gaussian pyramid computation.

To make the filters more intuitive, it is useful to write the two steps in matrix form. The following matrix shows the recursive construction of level $k+1$ of the Gaussian pyramid for a 1D image:

$$g_{k+1} = D_k B_k g_k = G_k g_k \tag{3.3}$$

where D_k is the downsampling operator, B_k is the convolution with the 4-th binomial filter, and $G_k = D_k B_k$ is the blur-and-downsample operator for level k. We call the sequence of images $g_0, g_1, ..., g_N$ as the Gaussian pyramid. The first level of the Gaussian pyramid is the input image: $g_0 = x$.

It is useful to check a concrete example. If x is a 1D signal of length 8, and if we assume zero boundary conditions, the matrices for computing g_1 are:

$$G_0 = D_0 B_0 = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix} \frac{1}{16} \begin{bmatrix} 6 & 4 & 1 & 0 & 0 & 0 & 0 & 0 \\ 4 & 6 & 4 & 1 & 0 & 0 & 0 & 0 \\ 1 & 4 & 6 & 4 & 1 & 0 & 0 & 0 \\ 0 & 1 & 4 & 6 & 4 & 1 & 0 & 0 \\ 0 & 0 & 1 & 4 & 6 & 4 & 1 & 0 \\ 0 & 0 & 0 & 1 & 4 & 6 & 4 & 1 \\ 0 & 0 & 0 & 0 & 1 & 4 & 6 & 4 \\ 0 & 0 & 0 & 0 & 0 & 1 & 4 & 6 \end{bmatrix} \tag{3.4}$$

Multiplying the two matrices:

$$g_1 = G_0 x = \frac{1}{16} \begin{bmatrix} 6 & 4 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 4 & 6 & 4 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 4 & 6 & 4 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 4 & 6 & 4 \end{bmatrix} x \tag{3.5}$$

the first level of the gaussian pyramid is a signal g_1 with length 4. Applying the recursion we can write the output of each level as a function of the input x: $g_2 = G_1 G_0 x$, $g_3 = G_2 G_1 G_0 x$, and so on.

For 2D images the operations are analogous. Figure 3.3 shows the Gaussian pyramid of an image.

3.4 Laplacian pyramid

In the gaussian pyramid, each level losses some of the fine image details available in the previous level. The Laplacian pyramid is simple: it represents, at each level, what is present in a Gaussian pyramid image of one level, but not present at the level below it. We calculate

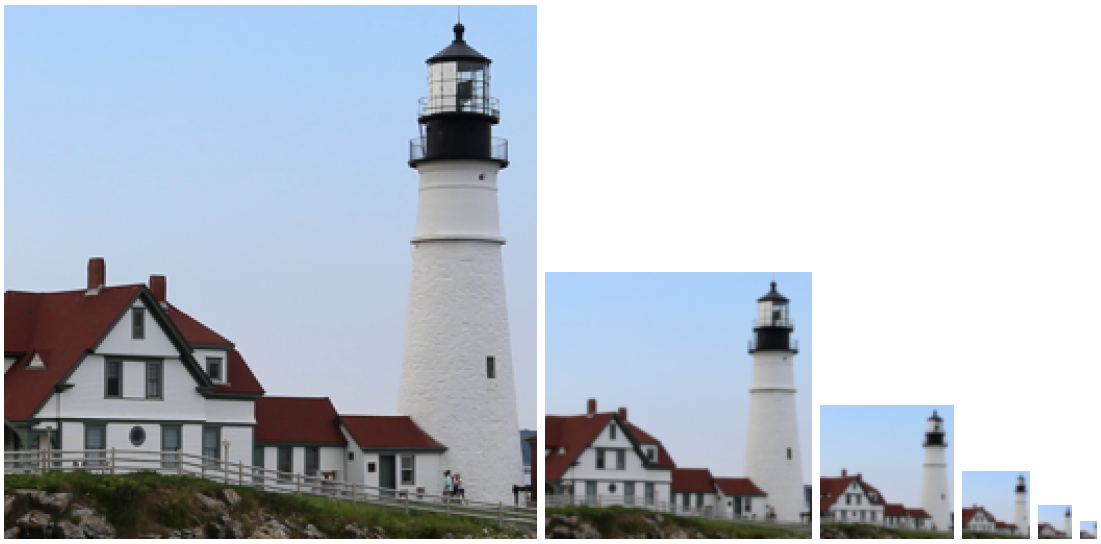
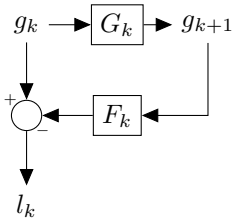


Figure 3.2: Gaussian pyramid. The Gaussian pyramid is built for each color channel independently.

that by expanding the lower-resolution Gaussian pyramid image to the same pixel resolution as the neighboring higher-resolution Gaussian pyramid image, then subtracting the two. This calculation is made in a recursive, telescoping fashion.

One **block** of the Laplacian pyramid computation.



Let's look at the steps for calculating a Laplacian pyramid. What we want is to compute the difference between g_k and g_{k+1}. To do this first we need to upsample the image g_{k+1} so that it has the same size as g_k. Let $F_k = B_k U_k$ be the upsample-and-blur operator for pyramid level k. The operator F_k applies first the upsampling operator U_k, that inserts zeros between samples, followed by blurring by the same filter B_k than the one we used for the Gaussian pyramid. The Laplacian pyramid coefficients, l_k, at pyramid level k, are:

$$l_k = g_k - F_k g_{k+1} = (I_k - F_k G_k) g_k \tag{3.6}$$

For instance, for a 1D input x of length 8, and assuming zero boundary conditions, the operators to compute the first level of the Laplacian pyramid are:

$$F_0 = 2B_0U_0 = \frac{1}{8}\begin{bmatrix} 6 & 4 & 1 & 0 & 0 & 0 & 0 & 0 \\ 4 & 6 & 4 & 1 & 0 & 0 & 0 & 0 \\ 1 & 4 & 6 & 4 & 1 & 0 & 0 & 0 \\ 0 & 1 & 4 & 6 & 4 & 1 & 0 & 0 \\ 0 & 0 & 1 & 4 & 6 & 4 & 1 & 0 \\ 0 & 0 & 0 & 1 & 4 & 6 & 4 & 1 \\ 0 & 0 & 0 & 0 & 1 & 4 & 6 & 4 \\ 0 & 0 & 0 & 0 & 0 & 1 & 4 & 6 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix} \tag{3.7}$$

The factor 2 is necessary because inserting zeros decreases the average value of the signal g_{k+1} by a factor of 2. Multiplying the two matrices:

$$l_0 = g_0 - F_0 g_1 = g_0 - \frac{1}{8}\begin{bmatrix} 6 & 1 & 0 & 0 \\ 4 & 4 & 0 & 0 \\ 1 & 6 & 1 & 0 \\ 0 & 4 & 4 & 0 \\ 0 & 1 & 6 & 1 \\ 0 & 0 & 4 & 4 \\ 0 & 0 & 1 & 6 \\ 0 & 0 & 0 & 4 \end{bmatrix} g_1 \tag{3.8}$$

The Laplacian pyramid is an overcomplete representation (more coefficients than pixels): the dimensionality of the representation is higher than the dimensionality of the input.

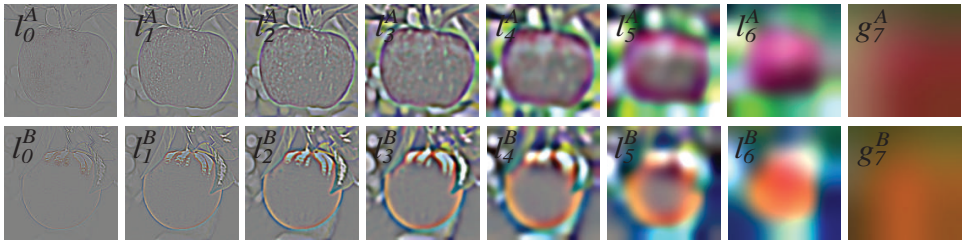
Note that the reconstruction property of the Laplacian pyramid does not depend on the filters used for subsampling and upsampling. Even if we used random filters the reconstruction property would still hold.

3.4.1 Image blending

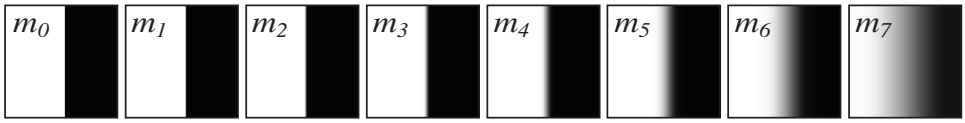
The Laplacian pyramid is used in many image processing or analysis applications. Here we show one fun application: image blending. The goal is to combine two images into one. A mask is used to define how the images will be combined. If we want to blend the following two images using the mask shown in the right:

Making a sharp transition from one image to another gives an artifactually sharp image boundary (see the straight edge of the apple/orange.)

Using the Laplacian pyramid, we can transition from one image to the next over many different spatial scales to make a gradual transition between the two images. First, we build the Laplacian pyramid for the two input images, in this example we use 7 levels and we also keep the last low-pass residual:



and the Gaussian pyramid of the mask as shown below (note that we use 8 levels, one level more than for the Laplacian pyramid):



Now we combine the three pyramids to compute the Laplacian pyramid of the blended image. The Laplacian pyramid of the blended image is obtained as: $l_k = l_k^A * m_k + l_i^B * (1 - m_k)$. The same is done for the low-pass residual.

3.5 Steerable pyramid

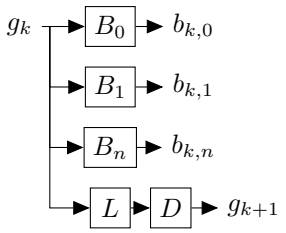
The Laplacian pyramid provides a richer representation than the Gaussian pyramid. But we would like to have an even more expressive image representation. The steerable pyramid adds information about image orientation. Therefore, the Steerable representation is a multi-scale oriented representation that is translation-invariant. It is non-aliased and self-invertible. Ideally, we'd like to have an image transformation that was shiftable–where we could perform

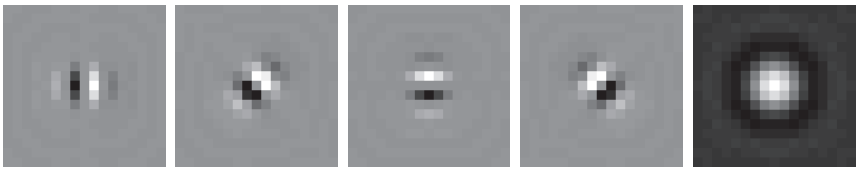
interpolations in position, scale, and orientation using linear combinations of a set of basis coefficients. The steerable pyramid goes part way there.

We analyze in orientation using a steerable filter bank. We form a decomposition in scale by introducing a low-pass filter (designed to work with the selected bandpass filters), and recursively breaking the low-pass filtered component into angular and low-pass frequency components. Pyramid subsampling steps are preceded by sufficient low-pass filtering to remove aliasing.

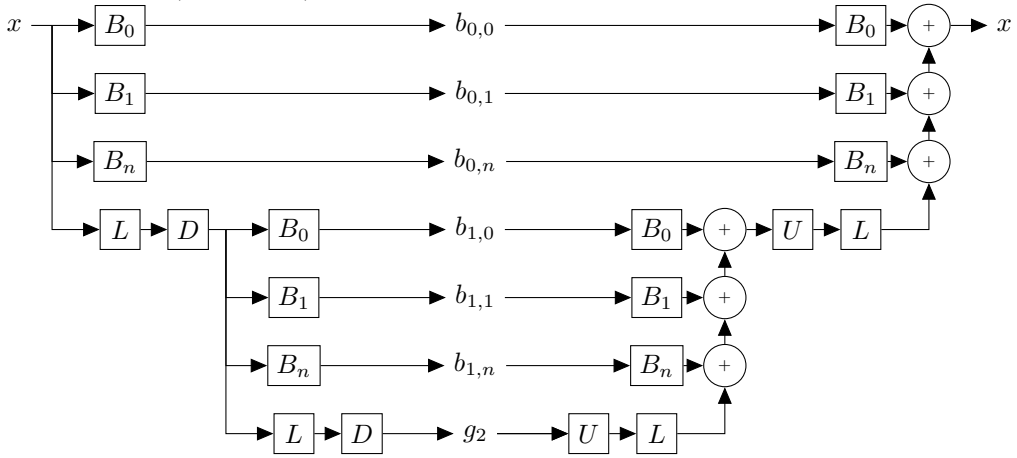
To ensure that the image can be reconstructed from the steerable filter transform coefficients, the filters must be designed so that their sums of squared magnitudes "tile" in the frequency domain. We reconstruct by applying each filter a second time to the steerable filter representation, and we want the final system frequency response to be flat, for perfect reconstruction.

One **block** of the Steerable pyramid computation.





The following block diagram shows the steps to build a 2 level steerable pyramid and the reconstruction of the input. The architecture has two parts: 1) the analysis network (or encoder) that transforms the input image x into a representation composed of $r = [b_{0,0}, ..., b_{0,n}, b_{1,0}, ...b_{1,n}, ..., b_{k-1,0}, ...b_{k-1,n}]$ and the low pass residual g_{k-1}. And 2) the synthesis network (or decoder) that reconstructs the input from the representation r.



The Steerable pyramid is a self-inverting overcomplete representation (more coefficients than pixels).

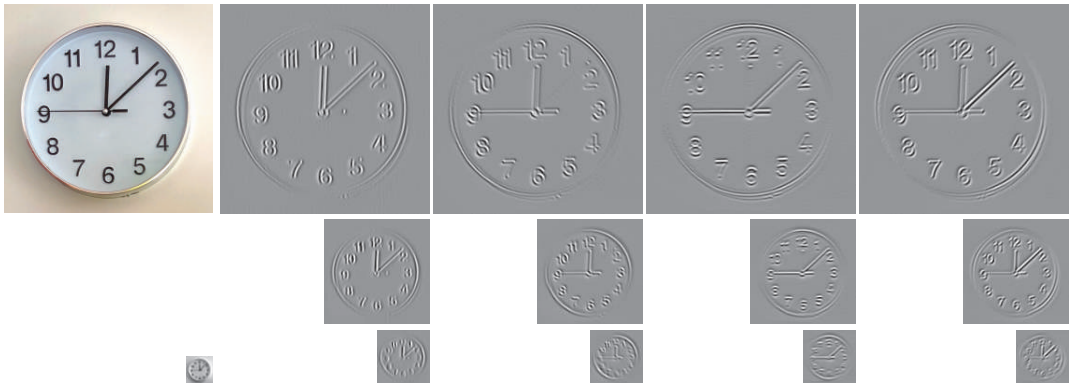


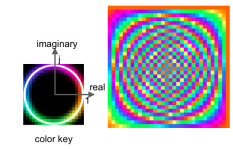
Figure 3.4: Steerable pyramid representation (3 levels and 4 orientations). Why is that each orientation subband seems to indicate a different time?

3.6 Summary

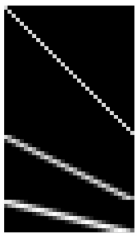
Here is a pictorial summary of the different pyramid representations we've been discussing in this chapter. The figure shows the projection matrices P for each transformation in the 1D case.

To recap briefly: The Fourier transform reveals spatial frequency content of the image wonderfully, but suffers from having no spatial localization. A Gaussian pyramid provides a multi-scale representation of the image, useful for applying a fixed-scale algorithm to an image over a range of spatial scales. But it doesn't break the image into finer components than simply a range of low-pass filtered versions. The representation is over-complete that is, there are more pixels in the Gaussian pyramid representation of an image than there are in the image itself.

The Laplacian pyramid reveals what is captured at one spatial scale of a Gaussian pyramid, and not seen at the lower-resolution level above it. Like the Gaussian pyramid, it is overcomplete. It is useful for various image manipulation tasks, allowing you to treat different spatial frequency bands separately. A steerable pyramid representation can be very overcomplete, depending on the number of orientations represented, but has negligible aliasing artifacts and so can be useful for various image analysis applications.

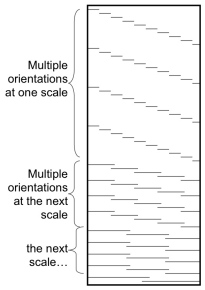


1-d Fourier transform



1-d Gaussian pyramid

1-d Laplacian pyramid



Steerable pyramid

Figure 3.5: Visual comparison of linear transform image representations discussed in this chapter. The Fourier transform gives a complex-valued output (represented in color) and is global–each output coefficient depends, in general, on every input pixel. The Gaussian pyramid is seen to be banded, showing that it is a localized transform where the output values only depend on pixels in a neighborhood. It is an over-complete representation, shown by the transform matrix being taller than it is wide. The Laplacian pyramid is a bandpassed image representation, except for the low-pass residual layer, shown in the bottom rows. For this matrix, zero is plotted as grey. The steerable pyramid is an over-complete, multi-orientation representation. In 1-d, image orientation isn't defined, but we show schematically a rasterization of the 2-d steerable pyramid transform. The multiple orientations, and non-aliased subbands cause the representation to be very overcomplete, much taller than it is wide. All the transforms, except for the Fourier transform, are convolutional, revealed by the diagonal banding in the matrices.