Chapter 6

II - Image decomposition

We have already seen that images need not be decomposed as pixels with their individual intensities. They can also be interpreted as a linear combination of sine patterns, as has been demonstrated by the Fourier transform, introduced in chapter 5. This chapter shows that the Fourier transform is just one alternative way of describing images as a combinatiuon of different patterns.

6.1 II - Scale space / image pyramids

Suppose you are flying in an airplane and you look down over a large forrest. At high altitude, this forrest is like a large green see, a homogeneously textured region. Flying lower, you will start seeing the individual trees. Still lower you would discern the individual leaves. Scenes can be looked at and processed at different scales, i.e. at different spatial resolutions. Depending on the resolution, other structures will be the dominant regions of interest. At a very large scale, we have the more or less homogeneously looking forrest. At finer resolution, the individual trees become objects of interest, etc.. The practical relevance of these ideas mainly lies in the fact that smaller images can be processed faster.

Thus, by building an *image pyramid* where higher levels correspond to lower resolution versions of the same image and then starting the processing at the pyramid's top, we hope to increase efficiency. We could first detect the large green region corresponding to the forrest. On a low resolution scale this region will be described by relatively few pixels in a similarly small overall image. Once we have spotted this region at the pyramid top, we can move down to find the individual trees, etc.. As we know, simply subsampling images in order to obtain the smaller images at the pyramid top is dangerous (Nyquist!). Therefore, prior to any reduction in resolution, we will smooth the previous image first. Thus we obtain a series of images, usually of dimensions $2^n \times 2^n$, with n ranging between two values, corresponding to the minimal and maximal resolutions, where subsequent images are obtained by smoothing and then subsampling their predecessor.

Not only is it necessary to detect and describe changes in an image at different scales, but in addition, much useful information can be obtained by combining descriptions across

scales. For instance, spatial coincidence at all scales of and edge found by some edge detection method may signal a physical edge distinct from surface markings and shadows, as was suggested by the late David Marr. Psychophysical research has shown in biological vision systems images are analyzed at different resolutions. Visual information is processed in parallel (although some time delays have been suggested to differ) by a number of spatial-frequency-tuned channels such that spatial filters of different sizes operate at the same location. Furthermore, the spatial grain of analysis continuously changes from foveal to peripheral locations. Average receptive field sizes of both retinal and cortical neurons increase monotonically with eccentricity (due to the increasing lack of smaller receptive fields).

Suppose we would now want to build such a pyramid, or, as has become more fashionable a name, *scale space* [1]. Which filter would we use to smooth the images in order to go to the next level? For one thing, we would like the newer level to contain less detail. We can interpret this as the new level containing less edges for instance. We would like the edges to simply stay (wander about), disappear, or melt together when going to lower resolutions. What we wouldn't like to see is that edges would appear out of the blue, i.e. that they result from smoothing without one being able to retrace them to edges in the original image. Such edges would bear no relation with the original data and would therefore be a nuisance. Fig. 6.1 shows a multi-resolution pyramid. The images on the left are the smoothed images. After each smoothing step (using an approximation to a Gaussian filter) the image is reduced in its dimensions: only half the width and height is taken. This reduction is justified if smoothing is sufficiently effective in reducing the higher frequencies, as we know from the Nyquist theorem (cf. chapter 5). The images on the right are the differences between each of the pictures on the left and its smoothed version (before size reduction). The difference between two images smoothed (i.e. low-pass filtered) to a different degree yields a band-passed image, as can easily be understood from the spatial frequency domain. As can be seen the edges stand out. Returning to the requirement that it should always be possible to retrace edges to the original (unsmoothed) image, one can prove that the only smoothing filter guaranteeing such behaviour is the Gaussian if the signal is one-dimensional. The difference between two Gaussian filters comes close to the Laplacian of a Gaussian, and following chapter ?? the zero-crossings of such filter are useful for the detection of edges as can be anticipated from the right side of the pyramid.

For the Gaussian zero-crossings of any linear operator applied to the different images in the scale space will always find their origins in zero-crossings at lower (less smoothed) levels. The zero-crossings of the Laplacian are an important example. One should be very careful in applying these ideas in practice. Confronted with the problem of implementation, one has to resort to discrete approximations of the images and the levels in scale space. This has important consequences. One will have to build discrete approximations of the Gaussian filter profile. The matter is not quite as simple as that, though. If one would

[1] The name scale space is in fact used more generally, for cases with continuous or discrete scale parameter, i.e. for a continuum of increasingly smooth images or for a stack of images taken for but a few, discrete scale parameter values. Also is there no assumption on the possible use of subsampling made, whereas the use of the name pyramid subsumes such subsampling to take place.

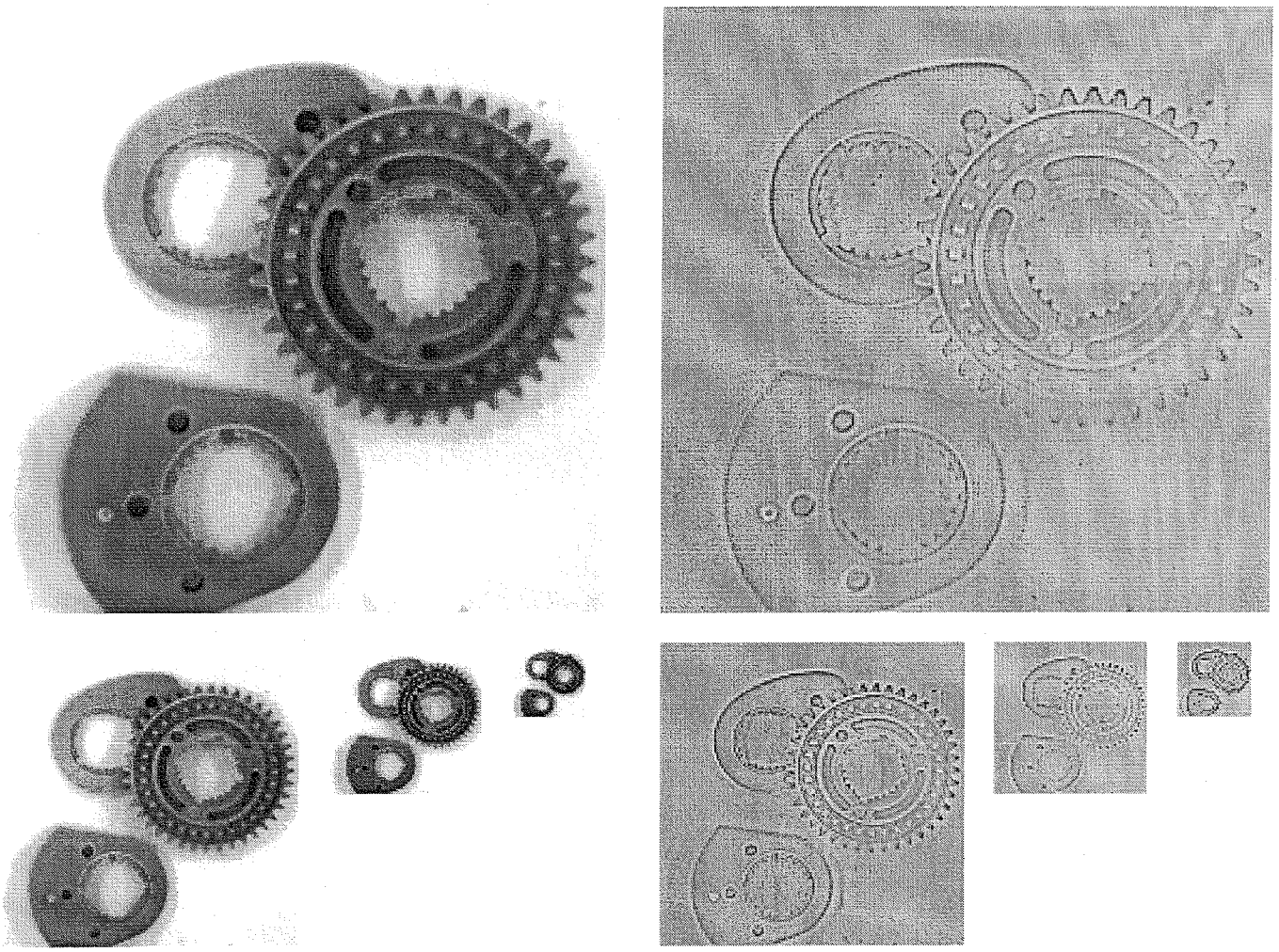

Figure 6.1: *Gaussian pyramid (left), corresponding Laplacian pyramid with bright values negative and dark values positive (right).*

e.g. decide to apply a small smoothing filter with positive coefficients c_{-1}, c_0, c_1, then one should make sure that $c_0^2 \geq 4c_{-1}c_1$ in order not to create structure that was absent at lower levels. Thus, the binomial filter [1,2,1] is a valid scale space filter, whereas [1,1,1] is not.

6.2 II - Unitary transforms

6.2.1 Image independent unitary transforms for compression

In chapter 5 the Fourier transform for images has been described. The transform can also be considered a decomposition of the image into cosine form basis images. This decomposition is not the only one possible, neither is it necessarily the best for a given application. One such application is image compression. Suppose we can write an image as a linear combination of a series of basis images (as we have alreadhy seen the Fourier transform). Now, if we could truncutate this decomposition after a small number of basis images while still having a good approximation of the image, and only keep the weights for the remaining basis images, this could constitute a good compression.

This section discusses unitary image transforms for image compression purposes. The unitary aspect of the transforms is of particular importance, as will become clear. Unitary is for complex functions what orthogonality is for real functions. In chapter 5 an image was written as a linear combination of weighted Dirac impulses. We could interpret this as a trivial decomposition into basis images where each basis image contains a single Dirac impulse at a different pixel each. All such basis images are orthogonal, i.e. their inner product (correlation) is zero. The inner product of two $N \times N$ basis images i_1 and i_2 is defined as

$$\sum_{x=0}^{N-1} \sum_{y=0}^{N-1} i_1(x,y)\ i_2^*(x,y)$$

with * the complex conjugate. As can be seen, this is a generalisation of the orthogonality concept for vectors. One can also say that the basis images are decorrelated and thus contain little redundancy with respect to eachother. In fact, one can interpret an image as a large vector of pixel intensities. Hence, basis images are such vectors and the sum expresses that in case of unitary transforms they are orthogonal.

Now, going from the original Dirac-based decomposition (one weight, namely the intensity value, per pixel) to a linear combination of unitary, basis images amounts to describing the image in terms of one set of orthogonal basis vectors to another such basis. This means that the unitary transforms apply a rotation to the data. The hope is that after this rotation the projection of the image (a vector) onto many of the new axes vanish. In other words, we hope we can truncate the linear decomposition after a few terms. For the Fourier transform, we have e.g. seen that most energy is concentrated in the lower frequencies and therefore could drop the higher frequency basis images in the decomposition and still get a good approximation of the image.

In the sequel, the concepts of orthogonality and its relation to the selection of useful basis images for the efficient decomposition of images will be clarified further. Then it

will be shown how arbitrary unitary transforms can be interpeted as such decompositions. Moreover, it will be shown that the decomposition coefficients are optimal in a least-square sense.

Orthogonality of functions

Two (possibly complex) square integrable functions $\phi_i(x)$ and $\phi_j(x)$ are called *orthogonal* on the interval $[a, b]$ if

$$\int_a^b \phi_i(x)\phi_j^*(x)dx = 0$$

with * indicating the complex conjugate. The trivial function $\phi(x) = 0$ clearly is orthogonal to all other functions, but we are only interested in orthogonal functions that also satisfy

$$\int_a^b |\phi_i(x)|^2 dx = a_i,$$

with $a_i \neq 0$ some positive, real number. In particular, we are interested in sets of *orthonormal* functions ϕ_i such that

$$\int_a^b \phi_i(x)\phi_j^*(x)dx = \delta_{ij}$$

with δ_{ij} the Kronecker delta, i.e. $\delta_{ij} = 1$ if $i = j$ and $\delta_{ij} = 0$ otherwise. It is clear that sets of orthogonal functions can be modified to sets of orthonormal functions via a simple scaling with a factor $a_i^{-\frac{1}{2}}$.

Before proceeding, it is useful to give examples of such sets of orthogonal functions. Consider the interval $[0, P]$ with $P = 2\pi/\omega$ which corresponds to a period for the functions $\cos m\omega x$ and $\sin m\omega x$ with $m = 1, 2, \ldots$. One can show that

$$\begin{aligned}
&\textstyle\int_0^P \cos m\omega x \ \cos n\omega x \ dx = \delta_{mn}\frac{P}{2} \ , \\
&\textstyle\int_0^P \cos m\omega x \ \sin n\omega x \ dx = 0 \ , \\
&\textstyle\int_0^P \sin m\omega x \ \sin n\omega x \ dx = \delta_{mn}\frac{P}{2} \ .
\end{aligned}$$

Considering all positive values of m, a countable set of orthogonal functions is generated. The reader will probably have made the connection to vector calculus already by himself. Indeed, in vector calculus sets of orthonormal vectors are chosen as a basis. The above is a generalization towards infinite dimensions. The parallel can be drawn further. Other vectors can be characterized on the basis of their projections on the basis vectors. In the same vein, arbitrary, square integrable functions can be characterized by their correlations with the basis set of orthonormal functions. In the continuous case, such characterization needs not be unique though. This is an example where the generalization from finite to infinite dimensions is not trivial. There are orthogonal basis sets of functions that do not allow unambiguous descriptions of the original function via such projection mechanism. Therefore, the orthonormal functions sets should also observe the *completeness condition*, i.e. one should make sure they allow such unambiguous description. As an example, the

set of orthogonal functions $\cos m\omega x$ is orthogonal but not complete. It is clear that the linear combination of such even functions can never produce odd functions.

In view of this complication, this is a good point to switch to discrete situations, since, after all, we only consider sampled images and sampled functions. In the discrete case we are dealing with finite-dimensional problems. Suppose the functions are represented as $(N \times 1)$ sample vectors. Completeness is no issue since any combination of N orthogonal basis functions will be a complete basis. The problem is now one of getting at sufficient orthogonal basis patterns. Consider an example with 16 samples (i.e. read-outs at 16 discrete positions). Hence, 16 orthogonal basis functions are required. The functions $\cos \frac{2\pi}{16} ux$ with $x = 0, 2 \ldots 15$ and $u = 0, 1, 2 \ldots$ are still an orthogonal set of functions. However, only up to $u = 8$ are these sampled functions really different. Beyond $u = 8$ the aliasing effect reflects all frequencies back onto frequencies within the $k = 0 \ldots 8$ range. One gets stuck with a mere set of 9 orthogonal basis functions. This stumbling block should not come as a surprise, since it is clear that odd, sampled functions can still not be represented as a linear combination of even, sampled ones. In order to construct a sufficiently large set of 16 orthogonal basis functions, another type of basis functions is called for. As an example, the Fourier basis functions

$$\frac{1}{N} e^{-2\pi i \frac{ux}{N}}$$

form a complete set of orthogonal basis functions. In these complex trigonometric functions lie "hidden" 9 basis functions in their real components (cos terms with $u = 0 \ldots 8$) and 7 basis functions in their imaginary part (sin terms with $u = 1 \ldots 7$, since $u = 0$ and $u = 8$ yield functions that vanish on all the sample values). This totals up the desired 16 orthogonal basis functions which are then combined in a way as to yield 16 orthogonal and independent complex trigonometric functions.

Many other systems of orthogonal functions can be found, a few of which will be described in later sections.

Basis images

So far the discussion has been confined to the one-dimensional case. Orthogonal function sets of higher dimensions can be easily obtained from there, however. Suppose we are dealing with $M \times N$ images and we have just found two 1-D sets, e.g. a set of M orthogonal functions $\phi_i(x)$ and a set of N orthogonal functions $\psi_j(y)$. Then we can construct MN orthogonal 2-D basis functions B_{ij} as follows:

$$B_{ij}(x, y) = \phi_i(x)\ \psi_j(y).$$

As a matter of notation, matrix and vector components wil be meant if there is mentioning of the row and/or column indices (as e.g. in the previous expression). If the matrix or vector itself is referred at, the indices are dropped. Vectors are column vectors. Hence, the above equation could equally well be written at the matrix rather than the matrix component level:

$$B_{ij} = \phi_i\ \psi_j^T$$

with ^{T} always meaning the matrix transpose.

These basis functions can then be used to decompose the images. Within this context the basis functions are usually referred at as *basis images*, as already mentioned. It is important to note that there are plenty of basis image sets that cannot be decomposed into products of 1-D basis functions. When this is possible the basis images are said to be *separable*. Since practical implementations of vision are almost exclusively concerned with the separable case, the entire further discussion will be based on that assumption.

Let us for a moment return to the observation that the original image has already been described as such a decomposition using a linear combination of Dirac impulses. It is clear that basis images $B_{ij}(k,l) = \delta_{ik}\delta_{jl}$ form a set of orthonormal basis images. In that sense, going to other decompositions can be seen as transitions from one orthonormal vector basis to another one. Such transitions are known be correspond to rotations and/or reflections, characterized by unitary transform matrices.

Decomposition of images

Now suppose we have a sampled function $f(x,y)$, i.c. an image, that we want to decompose in terms of the basis functions (images) $B_{uv}(x,y)$. The problem is to find the weights w_{uv} such that

$$f(x,y) = \sum_{u=0}^{M-1}\sum_{v=0}^{N-1} w_{uv}B_{uv}(x,y) \ . \tag{6.1}$$

In fact, this is an easy task, due to the orthonormality of the basis functions. Suppose we want to calculate a particular weight $w_{u'v'}$. If we multiply both sides of 6.1 with $B^*_{u'v'}(x,y)$ and take a sum over the whole domain, then we obtain

$$\begin{aligned}
\textstyle\sum_{x=0}^{M-1}\sum_{y=0}^{N-1} f(x,y)B^*_{u'v'}(x,y) &= \textstyle\sum_{x=0}^{M-1}\sum_{y=0}^{N-1}\sum_{u=0}^{M-1}\sum_{v=0}^{N-1} w_{uv}\ B_{uv}(x,y)B^*_{u'v'}(x,y) \ , \\
&= \textstyle\sum_{u=0}^{M-1}\sum_{v=0}^{N-1} w_{uv}\left(\sum_{x=0}^{M-1}\sum_{y=0}^{N-1} B_{uv}(x,y)B^*_{u'v'}(x,y)\right) \ , \\
&= w_{u'v'} \ .
\end{aligned}$$

This operation resembles the projection of a vector onto basis vectors, i.e. on the calculation of the coordinates of a vector w.r.t. an orthonormal basis. We can also interpret this operation as a correlation of the given image f with the basis images, making it clear that the weight factors $w_{u'v'}$ will be large for basis images that strongly resemble the given images and smaller otherwise.

We define the transformed image F, also of dimensions $M \times N$ as

$$F(u,v) = w_{uv} \ .$$

The *forward transform* is then given by

$$F(u,v) = \sum_{x=0}^{M-1}\sum_{y=0}^{N-1} f(x,y)\ B^*_{uv}(x,y) \ .$$

The *backward transform*, *backtransform*, or *inverse transform* we alreay met and is given by

$$f(x,y) = \sum_{u=0}^{M-1} \sum_{v=0}^{N-1} F(u,v)\ B_{uv}(x,y)\ .$$

Considering basis images that have been composed of two sets ϕ_u and ψ_v of 1-D orthonormal functions such that

$$B_{uv} = \phi_u \psi_v^T\ ,$$

the following relation can easily be shown to hold:

$$w_{uv} = \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} f(x,y)\ B_{uv}^*(x,y) = \phi_u^{*t}\ f\ \psi_v^*\ .$$

Hence, the calculation scheme for separable basis images entails separate 1-D operations on the rows and the columns of the image, which is preferable from the viewpoint of computational efficiency. Defining matrices Φ and Ψ as

$$\Phi = (\phi_1 \ldots \phi_M) \qquad \text{and} \qquad \Psi = (\psi_1 \ldots \psi_N)\ .$$

We have,

$$F = \Phi^{*T} f \Psi^*$$

and using the inverse matrices

$$f = \Phi F \Psi^T$$

where the unitary nature of the matrices was used ($\Phi^{-1} = \Phi^{*T}$ and $\Psi^{-1} = \Psi^{*T}$). Now, Φf can be seen to be a $M \times N$ matrix, say f_c, that results from column operations on f:

$$f_c(i,j) = \sum_{u=0}^{M-1} F(u,j)\ \phi_u(i)\ .$$

The final result is then obtained by applying row operations on f_c:

$$\begin{aligned} f(i,j) &= \textstyle\sum_{v=0}^{N-1} f_c(i,v)\ \psi_v(j)\ , \\ &= \textstyle\sum_{v=0}^{N-1} \left(\sum_{u=0}^{N-1} F(u,v)\ \phi_u(i) \right)\ \psi_v(j)\ , \end{aligned}$$

or, in terms of the whole image,

$$f = \sum_{u=0}^{M-1} \sum_{v=0}^{N-1} F(u,v) \phi_u \psi_v^T\ .$$

This result tallies with the original point of departure.

If the basis functions are such that the matrices are also Hermitian (or real symmetric) then also $\Phi^{-1} = \Phi$ and $\Psi^{-1} = \Psi$, since for a Hermitian matrix A one has $A^{*t} = A$. In that case the forward and backward transforms are identical:

$$\begin{aligned} F &= \Phi f \Psi^T\ , \\ f &= \Phi F \Psi^T\ . \end{aligned}$$

This is e.g. *not* the case for the Fourier transform. The columns of the matrix Φ would contain values like $e^{-i\frac{2\pi}{M}ux}$ with u fixed to the number of the column and x the row number. Inverting the matrix, an operation equivalent to taking the complex conjugate of the transpose, simplifies to taking the complex conjugate. This is because taking the transpose boils down to interchanging the values for u and x, which clearly makes no difference. The same holds for Ψ. Hence, in case of the Fourier transform the inverse transform uses Φ^* and Ψ^* instead of Φ and Ψ.

The situation is also simplified if $M = N$. In that case $\Phi = \Psi$ if the same sets of 1-D basis functions are used. This is the most common situation. Square images with dimensions that are powers of 2 are amenable to very fast treatment, due to fast versions that exist for most of the transforms in that case (e.g. the FFT).

Truncation property of unitary image transforms

The forward transform yields the weights w_{uv} to be used for an exact decomposition. If data compression is the goal, one has not gained much, since there still are MN such weights. This section deals with the question which weights to use if not all are retained.

Suppose we would like to find the truncated decomposition

$$\hat{f}(x,y) = \sum_{u=0}^{M'-1}\sum_{v=0}^{N'-1} c_{uv}B_{uv}(x,y)$$

with $M' < M$ and $N' < N$, that minimizes the sum of the errors squared. We thus look for the least-square approximation $\hat{f}(x,y)$ for $f(x,y)$ such that

$$e_{M'N'} = \sum_{x=0}^{M-1}\sum_{y=0}^{N-1} |f(x,y) - \hat{f}(x,y)|^2$$

is minimized. The answer is that we should just stick to the weights w_{uv} as is shown next.

theorem

The weights w_{uv} that minimize $e_{M'N'}$ are given by

$$w_{uv} = \sum_{x=0}^{M-1}\sum_{y=0}^{N-1} f(x,y)B^*_{uv}(x,y) \ .$$

Proof. We show that any other choice for the weights, say coefficients c_{uv}, would yield a larger $e_{M'N'}$. For such alternative choice of weights the error would be given by

$$\begin{aligned}
e_{M'N'} &= \textstyle\sum_{x=0}^{M-1}\sum_{y=0}^{N-1}\left|f(x,y) - \sum_{u=0}^{M'-1}\sum_{v=0}^{N'-1} c_{uv}B_{uv}(x,y)\right|^2 \ , \\
&= \textstyle\sum_{x=0}^{M-1}\sum_{y=0}^{N-1}\left|f(x,y) - \sum_{u=0}^{M'-1}\sum_{v=0}^{N'-1} w_{uv}B_{uv}(x,y) + \right. \\
&\qquad\qquad \left. \textstyle\sum_{u=0}^{M'-1}\sum_{v=0}^{N'-1}(w_{uv} - c_{uv})B_{uv}(x,y)\right|^2 \ , \\
&= \textstyle\sum_{x=0}^{M-1}\sum_{y=0}^{N-1}\left|\sum_{u=M'}^{M-1}\sum_{v=N'}^{N-1} w_{uv}B_{uv}(x,y) + \sum_{u=0}^{M'-1}\sum_{v=0}^{N'-1}(w_{uv} - c_{uv})B_{uv}(x,y)\right|^2 \ , \\
&= \textstyle\sum_{x=0}^{M-1}\sum_{y=0}^{N-1}\left|\sum_{u=M'}^{M-1}\sum_{v=N'}^{N-1} w_{uv}B_{uv}(x,y)\right|^2 + \sum_{u=0}^{M'-1}\sum_{v=0}^{N'-1}|w_{uv} - c_{uv}|^2 \ .
\end{aligned}$$

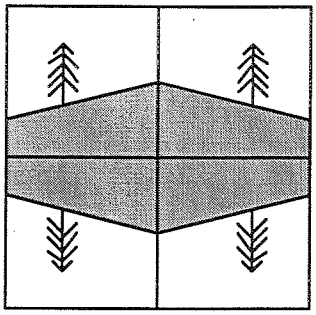

Figure 6.2: *Images can be made even by appending copies.*

The first term is independent of the choice of c_{uv} and the last term is always positive. Clearly, this sum is minimized (zero) for the choice $c_{uv} = w_{uv}$. Hence, the theorem is proved.

Proceeding from the last expression in the proof we see that for $c_{uv} = w_{uv}$:

$$\begin{aligned} e_{M'N'} &= \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} \left| \sum_{u=M'}^{M-1} \sum_{v=N'}^{N-1} w_{uv} B_{uv}(x,y) \right|^2 , \\ &= \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} \left(\sum_{u=M'}^{M-1} \sum_{v=N'}^{N-1} |w_{uv}|^2 \right) . \end{aligned}$$

It follows that $e_{M'N'}$ will never increase upon taking more terms in the decomposition.

6.2.2 Examples of unitary transforms

The Cosine transform

The previous sections have dealt with the issue of choosing weights for given sets of basis images. There remains the important problem of selecting basis images that yield strong compaction of variance. This selection process has to a large extent been driven by practical experience and concerns about efficiency. A particularly important choice yields the *discrete cosine transform.*

As we have already seen in chapter 5, the Fourier transform of an even, real function consists of cosine terms only. On the other hand, we have introduced the construction of fig. 6.2, leading to even images. As we have noted there, this construction is useful for the elimination of false high frequencies from the DFT. The extended image is even around the point $(-\frac{1}{2}, -\frac{1}{2})$. The DFT F_e of this extended, even image f_e is given by

$$F_e(u,v) = \frac{1}{4N^2} \sum_{x=-N}^{N-1} \sum_{y=-N}^{N-1} f_e(x,y)\, e^{-2\pi i \left(\frac{u(x+1/2)}{2N} + \frac{v(y+1/2)}{2N} \right)}$$

for $u = -N, \ldots, N-1$ and $v = -N, \ldots, N-1$. Because $f_e(x,y)$ is even, this expression can be simplified to

$$F_e(u,v) = \frac{1}{N^2} \sum_{x=-N}^{N-1} \sum_{y=-N}^{N-1} f_e(x,y) \cos\left(\frac{\pi}{N} u(x+1/2) \right) \cos\left(\frac{\pi}{N} v(y+1/2) \right) .$$

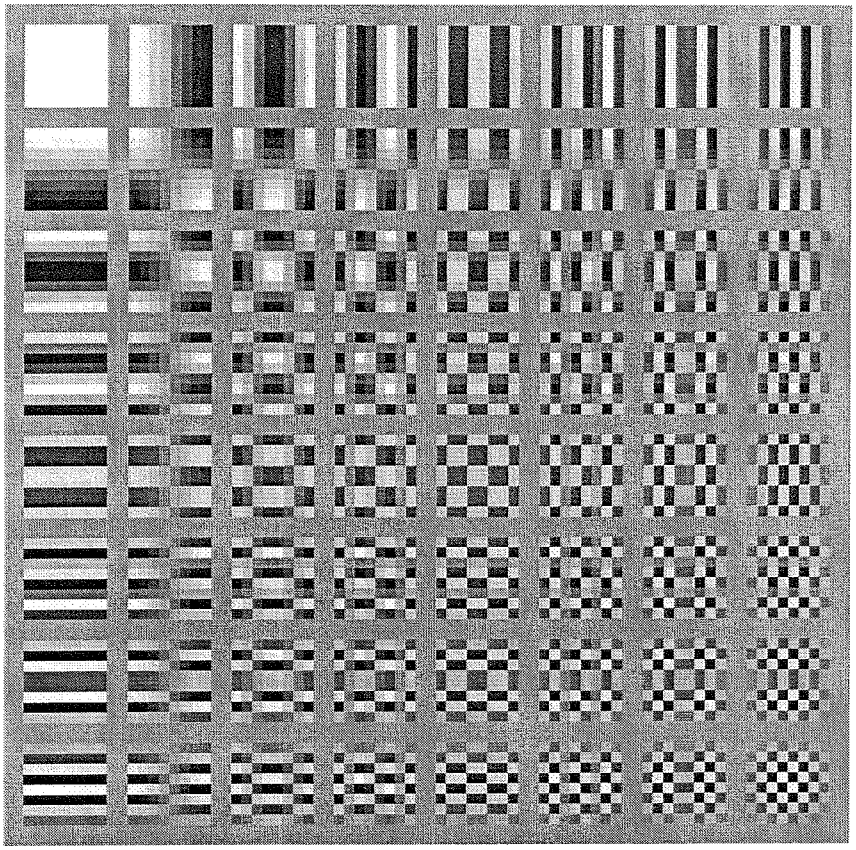

Figure 6.3: *Basis image for the cosine transform. Bright values are positive, dark values negative and zero corresponds to some grey level in between.*

The forward cosine transform is defined as

$$F(u,v) = \frac{4C(u)C(v)}{N^2} \sum_{x=0}^{N-1} \sum_{y=0}^{N-1} f(x,y) \cos\left(\frac{\pi}{N} u(x+1/2)\right) \cos\left(\frac{\pi}{N} v(y+1/2)\right).$$

where

$$C(w) = \begin{cases} \frac{1}{\sqrt{2}} & \text{for } w = 0, \\ 1 & \text{for } w = 1, 2, \ldots, N-1. \end{cases}$$

The resulting basis images for a 8×8 image are shown in fig. 6.3. The backward DCT is defined as

$$f(x,y) = \sum_{u=0}^{N-1} \sum_{v=0}^{N-1} C(u)C(v)\ F(u,v) \cos\left(\frac{\pi}{N} u(x+1/2)\right) \cos\left(\frac{\pi}{N} v(y+1/2)\right).$$

In the previous section we have seen that taking only a limited number of terms in the expansions leads to minimal truncation errors. The DCT transform is the most popular transform for such compression applications. Just as with the FFT, the DCT has a



Figure 6.4: *Girl image to be compressed.*

$\mathcal{O}(n \log n)$ fast implementation. Straightforward implementations would have complexity $\mathcal{O}(n^2)$. The DCT eliminates the boundary discontinuities introduced by the DFT. Whereas the DFT will always contain high frequencies from the boundary discontinuities, such high frequencies do not occur in the DCT. As a result, energy in the DCT will be more strongly concentrated at low frequencies and the resulting compression efficiency is higher. Its different components are, on the average, more decorrelated than those obtained with other unitary, image independent transforms. Its energy compaction behaviour is often quite comparable to that of the KLT transform (and for certain types of image statistics the KLT is identical to the DCT). Moreover, the DCT only requires real computations. DCT chips are available.

As an illustration of the compression capability of the DCT, consider fig. 6.4. Fig. 6.5 shows the Fourier power spectrum and the cosine transform of this image. Notice that *dc* is found at the center of the Fourier spectrum and at the upper left corner of the cosine transform image (the Cosine Transform being a quadrant of the Fourier transform of a larger image containing four reflected copy images). The presence of higher frequencies is clearly more salient with the Fourier transform. A vertical stripe indicates components up to high frequencies along that direction, originating from the discontinuity between the top and bottom rows in fig. 6.4. Fig. 6.6 shows a zonal truncation for both transforms: the same number of low frequency components are retained in both cases (remember that the Fourier transform has a real and a complex component and that for real input such as the example image, the real component will be even and the imaginary component will be odd). Finally, backtransforming the zonal truncations, the compressed images of fig. 6.7 are found. Since the edges contain higher frequencies primarily, artefacts can be seen at these places. Upon comparing both results, the horizontal ripple at the top and bottom of the DFT based image will be noted. These result from the spurious high frequencies introduced by the periodic repetition of the pattern. The DCT based image doesn't suffer from this shortcoming.

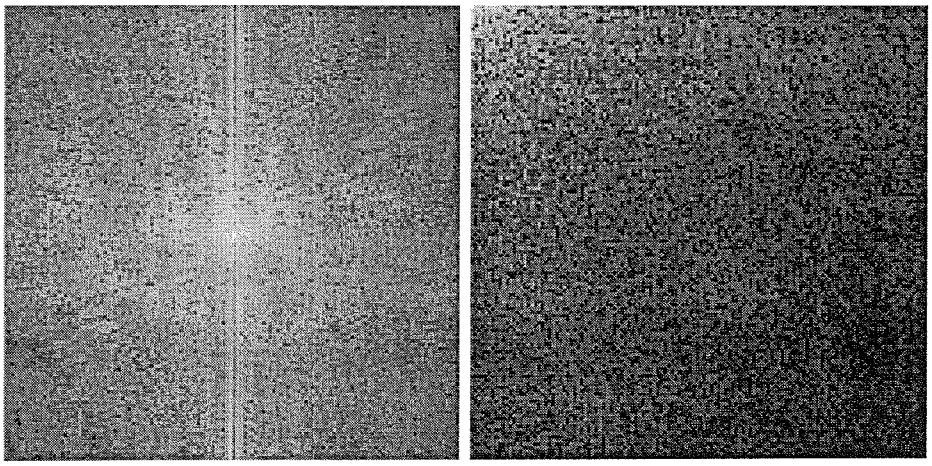

Figure 6.5: *Left: Fourier power spectrum of fig. 6.4, Right: Cosine transform of the same image.*

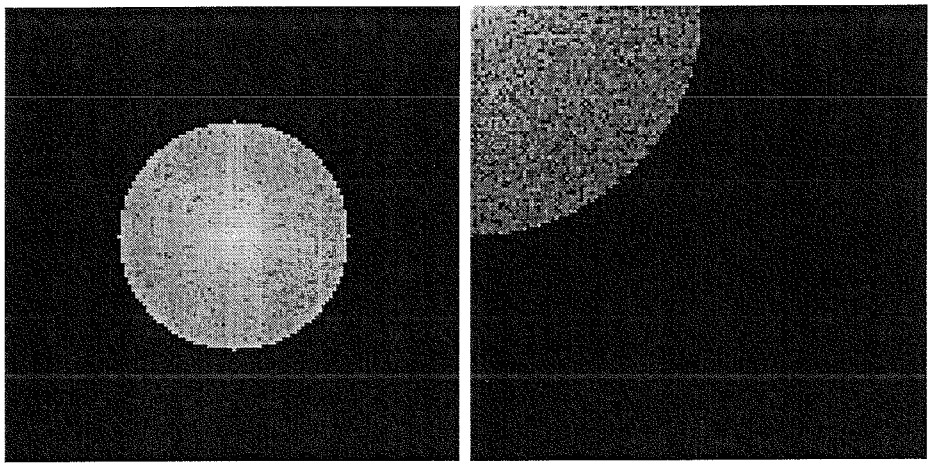

Figure 6.6: *Left: zonal truncation of the Fourier spectrum around DC, Right: zonal truncation of the Cosine Transform around DC.*



Figure 6.7: *Top: DFT compressed image, bottom: DCT compressed image.*

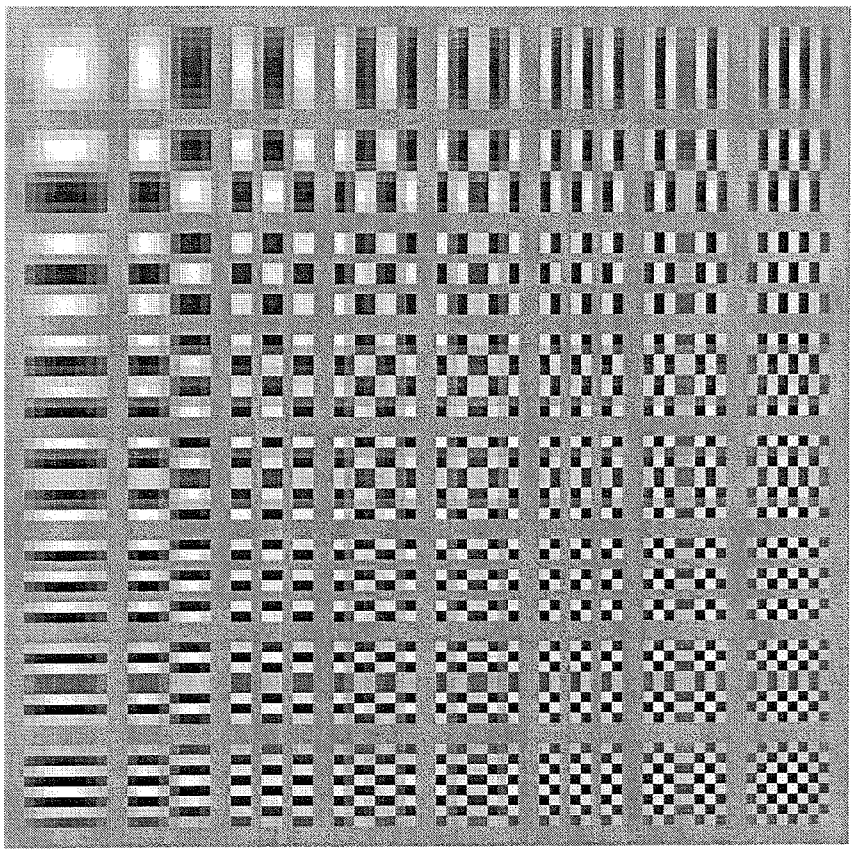

Figure 6.8: *Basis image for the sine transform*

Other unitary transforms

Besides the DCT, apart from the DFT the most important image independent unitary transform, several other such transforms have been introduced. We now give a number of examples of such transforms. They are not often used and will therefore be treated very superficially. Images will be assumed square.

The two-dimemsional *sine transform* is defined as

$$F(u,v) = \frac{2}{(N+1)} \sum_{x=0}^{N-1} \sum_{y=0}^{N-1} f(x,y) \sin\left(\frac{\pi}{N+1}(x+1)(u+1)\right) \sin\left(\frac{\pi}{N+1}(y+1)(v+1)\right)$$

Basis images for the sine transform as shown in fig. 6.8 in the case of a 8×8 image.

The *Hadamard transform* can be implemented with higher speed, because it only contains 1s and -1s. It thus requires no multiplications. It therefore was one of the first transforms to be implemented in hardware, mainly for compression applications. With hardware ever becoming more powerful it has increasingly been replaced by the DCT. The Hadamard transform is built around a class of matrices with orthogonal (Walsh) rows

and columns. Starting from the matrix

$$H_2 = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}.$$

A recursive technique then allows the construction of such matrices with an even number of rows and columns:

$$H_{2(n+1)} = \frac{1}{\sqrt{2}} \begin{pmatrix} H_n & H_n \\ H_n & -H_n \end{pmatrix}$$

These Hadamard matrices are all symmetric. For $n = 8$ this yields

$$H_8 = \frac{1}{2\sqrt{2}} \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 & 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 \\ 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 \\ 1 & -1 & 1 & -1 & -1 & 1 & -1 & 1 \\ 1 & 1 & -1 & -1 & -1 & -1 & 1 & 1 \\ 1 & -1 & -1 & 1 & -1 & 1 & 1 & -1 \end{pmatrix}$$

The rows (viz. columns) of these matrices can be used to build the basis images. Each row can be seen as defining a filter with the row elements as the filter's coefficients. Applying one such filter to the rows and another to the columns, we obtain the basis images. The resulting 64 Hadamard basis images for the 8×8 case are shown in fig. 6.9. Rather than decomposing a function into sine components, the Hadamard transform decomposes into block functions. This transform comes in different versions, depending on the order in which the basis images are structured.
The binary nature of the patterns has resulted in some useful applications of the Hadamard transform in document processing, e.g. barcode reading.

Similarly, the *Haar transform* is defined on the basis of Haar matrices. Without going into the details, the matrix for the 8×8 case is

$$\frac{1}{2\sqrt{2}} \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 \\ \sqrt{2} & \sqrt{2} & -\sqrt{2} & -\sqrt{2} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \sqrt{2} & \sqrt{2} & -\sqrt{2} & -\sqrt{2} \\ 2 & -2 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 2 & -2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 2 & -2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 2 & -2 \end{pmatrix}.$$

The corresponding basis images are shown in fig. 6.10. Slant matrices, which we won't specify, form the basis of the *Slant transform*. The basis images for the 8×8 case are illustrated in fig. 6.11.

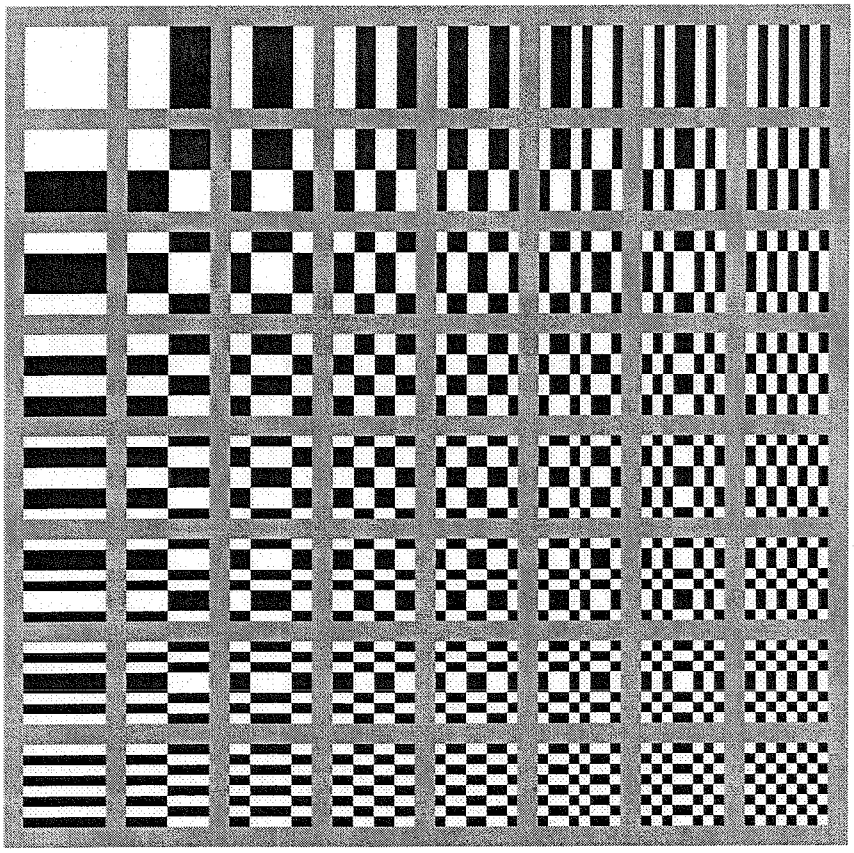

Figure 6.9: *Basis images for the Hadamard transform*

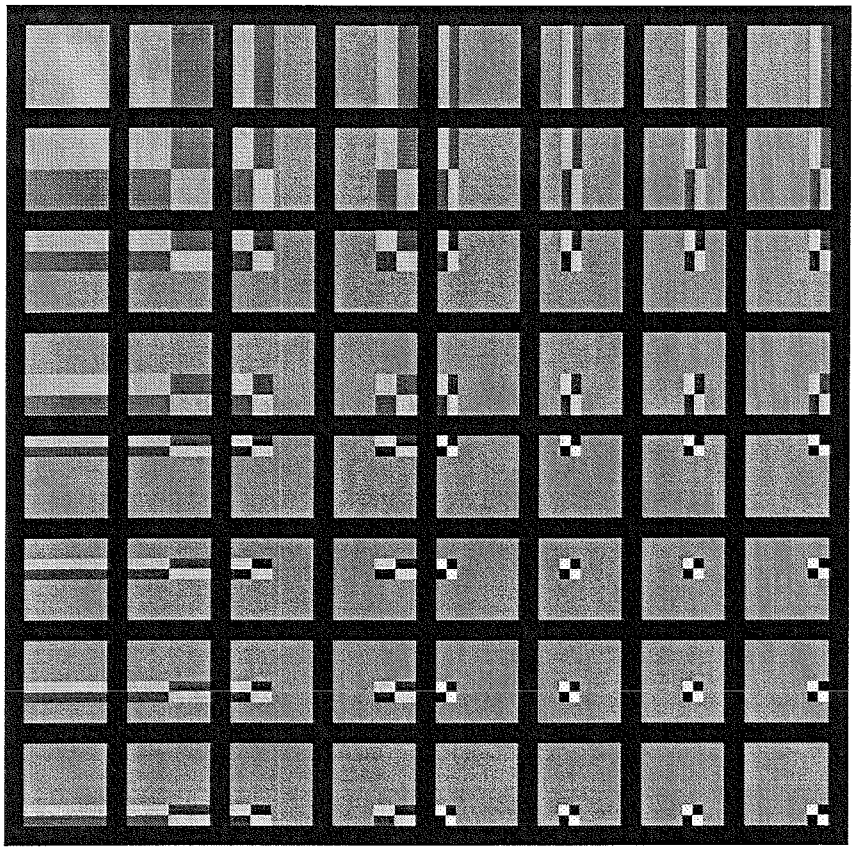

Figure 6.10: *Basis images of the Haar transform*

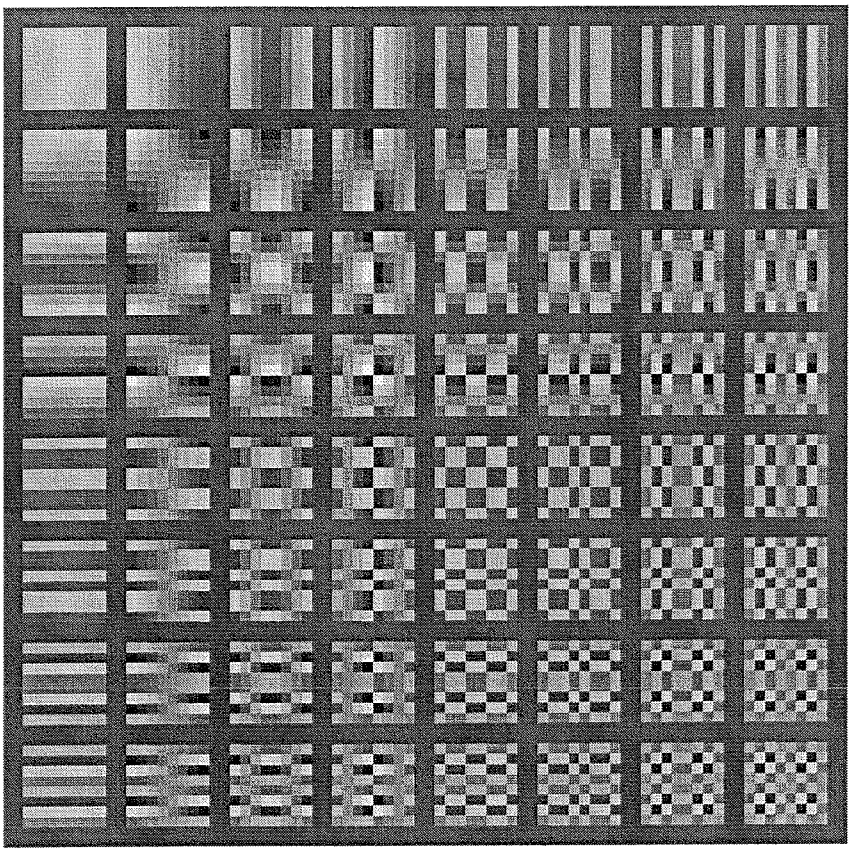

Figure 6.11: *Basis images of the Slant transform*

In these examples we have shown how the rows and columns of the basis images can be chosen from several sets of orthogonal vectors. In each case, we have selected the same orthogonal set for the rows and columns. In principle this need not be so. One could e.g. decompose the columns of the image into Hadamard waves and the rows in Haar waves, leading to many possible kinds of mixed transforms.

6.3 II - Wavelets

6.4 II - Principal Component Analysis

6.4.1 Highlighting the essence via decorrelation

The unitary transformations of section ?? decompose images into a set of fixed basis images. In other words, the rotation in the space of intensity vectors is always the same. One could wonder whether it is not possible to make this rotation dependent on the image content. Of course, using basis images that are specialised for a single image doesn't make sense. This would be a basis consisting of the image itself and some orthogonal images which all get weight zero in the decomposition. But one can imagine that one is looking at images with a lot of characterisitics in common. Examples could be head-and-shoulder images are they are used in video-conferencing (i.e. images of a person whose head and shoulders are about the only thing visible in the image) or even images of faces only, that are kept in a database of criminal records. Wouldn't it be possible to come up with a set of basis images that describe and therefore compress such images more effectively?

The Karhunen-Loève transform (KLT) is a method that does exactly that. Given a set of images, it extracts the necessary statistics from them to generate a set of orthogonal basis images, the weights of which are uncorrelated. This means that every basis image brings in information that is uncorrelated with the previous. Therefore, redundancy in the information carried by the different basis images is reduced by at least avoiding linear dependencies. This will be all the more effective as the set of images that one considers look more alike. If this set is too diverse, one can prove that under certain assumptions of the statistics of general images, the KLT will start to resemble the DCT. This tallies with the success that the DCT has had for the compression of general images.

Note that image dependent decompositions such as the KLT have the disadvantage that the receiver end needs to get the basis images as well as the weights before the images can be reconstructed. However, if it is clear what kind of images are dealt with, the corresponding KLT basis images only have to be sent over once and can then be used whenever relevant.

The Karhunen-Loève transform is actually nothing but Principal Component Analysis (PCA) applied to images. Therefore, henceforth we will call it that way. PCA has many other applications in computer vision than compression. It is very useful for the selection of features for classification or inspection, for instance. We will meet PCA again in chapters 13 and 15.

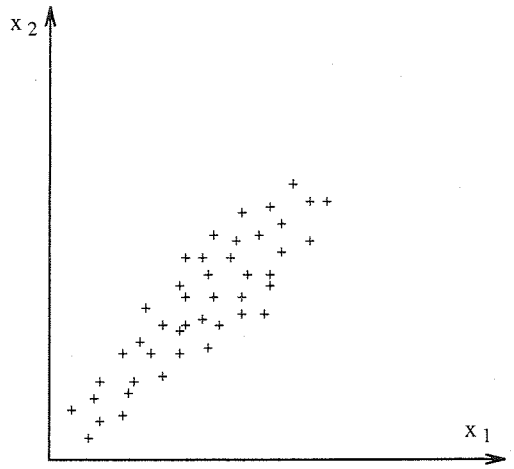

Figure 6.12: *Plot of observations for two correlated variables* x_1, x_2.

First, we look at a two-dimensional example to see what PCA (KLT) is about. Figure 6.12 shows a plot of observations on two highly correlated variables x_1, x_2. The reader could e.g. think of x_1 as the grey value in a particular pixel and x_2 as the grey value in its right neighbouring pixel. Each point in the figure represents such pair of intensities for one image, drawn from the set of images. For most images, the two intensities will be very close. Indeed, neighboring pixels tend to have similar intensities. which transpires through the diagonal nature of the point cloud in the figure. For the applications that are discussed further on, another example may be useful. Suppose we want to determine whether a person is a child or an adult from some simple measurements taken with simple sensors. Suppose x_1 and x_2 stand for the length and weight of the person. Since length and weight are correlated, again a linearly elongated point cloud will result.

Although there is considerable variation in both variables, knowing one yields pretty precise information on the value of the other. So instead of keeping track of two variables, measuring one would already yield much of the information. We thus could lower the storage cost by considering 1 instead of 2 variables. However, just keeping the value of one of the original variables x_1 or x_2 is suboptimal. We can take advantage of the correlation by a simple rotation of coordinate axes, as shown in fig. 6.13. The first, re-oriented component (z_1) has more variation than either of the original variables, whereas the second component z_2 has less variation, i.e. there is less uncertainty about this component. Dropping z_2 and only keeping z_1 causes a smaller loss of information of our compression. If one would fit an ellipse to the data, the z_1 axis would coincide with the long axis and the z_2 axis with the short axis. In that sense the axes are oriented to make the longest and shortest cuts through the ellipse, respectively. These axis are rotated with respect to the original axes in a way to minimize the uncertainty in the second component. Thus the information deleted by not considering the second component resulting from this specific rotation is minimal. This is the simple observation underlying PCA.

Typically, PCA will be applied to higher-dimensional cases, in which cases it fits an

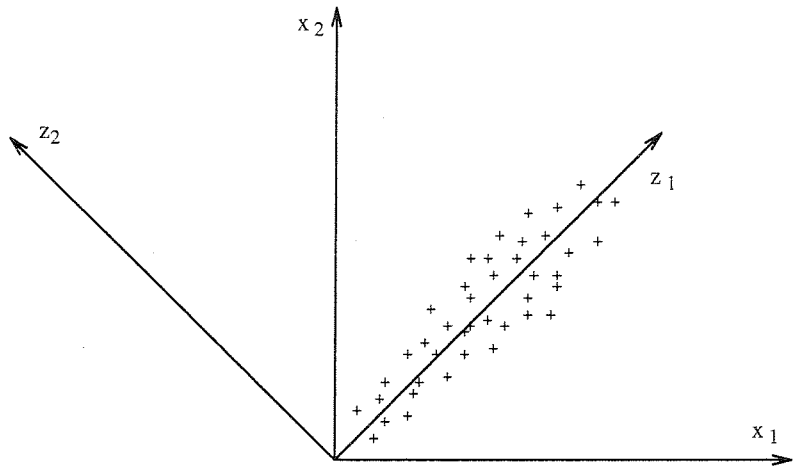

Figure 6.13: *Plot of the observations of fig. 6.12 with respect to rotated components z_1, z_2 and the relationship between these and the original variables.*

hyperellipsoid to the data, with the first component corresponding to the longest axis, the second component corresponding to the second longest axis, etc. In the case of the KLT, the dimension of the space equals the number of pixels in the images. We now set out to describe this process in mathematical terms.

6.4.2 The PCA algorithm

Generally, in a set of p-tuples, the p components have substantial correlations among them. Following the same strategy as in the two-dimensional case, a p-dimensional rotation can be employed to capture most of the variance in the data in a few coordinates. Such transformations are called *unitary* (or orthogonal when working in a real rather than a complex space). In fact, rotation preserves the overall variance in the data, i.e. for original coordinates x_i and rotated coordinates z_j we have

$$\sum_{i=1}^{p} \sigma_i^2 = \sum_{j=1}^{p} \tilde{\sigma_j}^2$$

with σ_i the variance in x_i and $\tilde{\sigma_j}$ the variance in z_j [2]. This relation can be understood by considering the fact that rotations map centers of gravity (i.e. the mean values to be used for calculating the variance) onto centers of gravity. The sum of the variances is then the average squared distance to the center of gravity. Since distances remain invariant under rotation also, this average squared distance will remain the same. Basically, the rotation redistributes the variance over the different coordinates. The aim is to concentrate as much of the variance in as few coordinates possible. In the next section a procedure will be outlined to achieve this redistribution. It will allow a rank-ordering of new coordinates

[2]This relation is usually referred at as the *Parseval equality*, which in turn is a special case of the *Bessel inequality*, which is important for the case of infinite dimensional spaces (Hilbert spaces).

z_j such that the first captures a maximum of variance, the second one the next highest variance from what is left, etc. The corresponding coordinates z_j, which can also be interpreted as linear combinations of the original variables, are called the *first*, the *second*, and so on *principal components* (PCs). For strongly correlated data, the first few PCs will account for most of the variance in the original variables. Conversely, the last few PCs identify directions in which there is very little variation, i.e. they identify near-constant linear relationships between the original variables.

Principal Component Analysis identifies the optimal coordinate rotation to obtain complete decorrelation. Although the example might suggest that the optimal rotation is clear upon visual inspection of the data, PCA is normally meant to deal with situations with many more variables, where visual interpretation would be extremely difficult. By supplying the complete decorrelation, PCA often allows a drastic reduction in the dimensionality of the problem, while retaining most of the variation, i.e. information, in the data. It suffices to retain the first principal components (henceforth denoted as PCs).

Some (moderate) knowledge of the data statistics should be available if one wants to apply PCA, i.e. we need the variances and covariances of the variables. Obtaining these might be rather straightforward, as with inspection on simple features such as weight or length, but also less trivial, e.g. when the image is considered to be a large vector. In the latter case one should consider the variances of pixel values and covariances for pixel pairs over the family of images to be considered, which usually is a rather fuzzy concept. Let us assume that the necessary statistics can be retrieved.

Next we derive a strategy to arrive at uncorrelated PCs. Suppose that x is a column vector of p random variables. The first step is to look for a linear combination $\mathbf{c}_i^T x$ of the components of x which has maximum variance, where $\mathbf{c}_i$ is a column vector of p constant coefficients. Then, look for a linear combination $\mathbf{c}_2^T x$, uncorrelated with $\mathbf{c}_1^T x$, which has maximum variance, and so on, so that at the kth stage a linear combination $\mathbf{c}_k^T x$ is found which has maximum variance and is uncorrelated with $\mathbf{c}_1^T \mathbf{x}, \mathbf{c}_2^T \mathbf{x}, \cdots, \mathbf{c}_{k-1}^T \mathbf{x}$. The kth derived combination $\mathbf{c}_k^T x$ is the kth PC. Up to p PCs could be found, but it is hoped, in general, that most of the variation in $\mathbf{x}$ will be accounted for by much fewer than p PCs.

Having clarified the concept of PCs, we need to know how to find them. Consider the case where the vector of variables $\mathbf{x}$ has a known covariance matrix C. This is the matrix whose (i,j)th element is the covariance between the ith and jth component of $\mathbf{x}$ when $i \neq j$, and the variance of the jth component of $\mathbf{x}$ when $i = j$. To derive the form of the PCs we first consider $\mathbf{c}_1^T \mathbf{x}$, with $var[\mathbf{c}_1^T \mathbf{x}] = \mathbf{c}_1^T C \mathbf{c}_1$. As it stands, the maximum will not be achieved for finite $\mathbf{c}_1$, so a normalization constraint must be imposed. The most convenient constraint here is $\mathbf{c}_1^T \mathbf{c}_1 = 1$. Using the technique of Lagrange multipliers we maximize

$$\mathbf{c}_1^T C \mathbf{c}_1 - \lambda(\mathbf{c}_1^T \mathbf{c}_1 - 1) \ ,$$

where λ is a Lagrange multiplier. Differentiation with respect to the components of $\mathbf{c}_1$ gives the system of equations

$$C\mathbf{c}_1 - \lambda \mathbf{c}_1 = 0 \ ,$$

or,

$$(C - \lambda I_p)\mathbf{c}_1 = 0 \ ,$$

where I_p is the $(p \times p)$ identity matrix. Thus, λ is an eigenvalue of C and $\mathbf{c}_1$ is the corresponding eigenvector. To decide which of the p eigenvectors is the maximizing value of $\mathbf{c}_1$, note that the quantity to be maximized is

$$\mathbf{c}_1^T C \mathbf{c}_1 = \mathbf{c}_1^T \lambda \mathbf{c}_1 = \lambda \mathbf{c}_1^T \mathbf{c}_1 = \lambda,$$

so λ must be as large as possible. Thus, $\mathbf{c}_1$ is the eigenvector corresponding to the largest eigenvalue of C, and $var(\mathbf{c}_1^T \mathbf{x}) = \mathbf{c}_1^T C \mathbf{c}_1 = \lambda_1$, the largest eigenvalue. In general, the kth PC $\mathbf{c}_k^T \mathbf{x}$ has $var(\mathbf{c}_k^T \mathbf{x}) = \lambda_k$, where λ_k is the kth largest eigenvalue of C, and $\mathbf{c}_k$ is the corresponding eigenvector. This will now be proved for $k = 2$; the proof for $k \geq 3$ is slightly more complicated, but very similar. The second PC, $\mathbf{c}_2^T x$, maximizes $\mathbf{c}_2^T C \mathbf{c}_2$ subject to being uncorrelated with $\mathbf{c}_1^T \mathbf{x}$ (i.e. subject to $cov[\mathbf{c}_1^T \mathbf{x}, \mathbf{c}_2^T \mathbf{x}] = 0$, where $cov[a, b]$ denotes the covariance between the random variables a and b). But

$$cov[\mathbf{c}_1^T \mathbf{x}, \mathbf{c}_2^T \mathbf{x}] = \mathbf{c}_1^T C \mathbf{c}_2 = \mathbf{c}_2^T C \mathbf{c}_1 = \mathbf{c}_2^T \lambda_1 \mathbf{c}_1 = \lambda_1 \mathbf{c}_2^T \mathbf{c}_1 = \lambda_1 \mathbf{c}_1^T \mathbf{c}_2.$$

Thus any of the equations

$$\mathbf{c}_1^T C \mathbf{c}_2 = 0, \ \mathbf{c}_2^T C \mathbf{c}_1 = 0, \ \mathbf{c}_1^T \mathbf{c}_2 = 0, \ \mathbf{c}_2^T \mathbf{c}_1 = 0,$$

could be used to specify no correlation between $\mathbf{c}_1^T \mathbf{x}$ and $\mathbf{c}_2^T \mathbf{x}$. Choosing the last of these (an arbitrary choice), and noting that, once again, a normalization constraint is necessary, the quantity to be maximized is

$$\mathbf{c}_2^T C \mathbf{c}_2 - \lambda(\mathbf{c}_2^T \mathbf{c}_2 - 1) - \phi \mathbf{c}_2^T \mathbf{c}_1,$$

where λ, ϕ are Lagrange multipliers. Differentiation with respect to $\mathbf{c}_2$ gives

$$C\mathbf{c}_2 - \lambda \mathbf{c}_2 - \phi \mathbf{c}_1 = 0$$

and multiplication of this equation on the left by $\mathbf{c}_1^T$ gives

$$\mathbf{c}_1^T C \mathbf{c}_2 - \lambda \mathbf{c}_1^T \mathbf{c}_2 - \phi \mathbf{c}_1^T \mathbf{c}_1 = 0,$$

which, since the first two terms are zero and $\mathbf{c}_1^T \mathbf{c}_1 = 1$, reduces to $\phi = 0$. Therefore, $C\mathbf{c}_2 - \lambda \mathbf{c}_2 = 0$, i.e. $(C - \lambda I_p)\mathbf{c}_2 = 0$, and once again λ is an eigenvalue of C and $\mathbf{c}_2$ the corresponding eigenvector. Again, $\lambda = \mathbf{c}_2^T C \mathbf{c}_2$, so λ is to be as large as possible. Thus $\lambda = \lambda_2$, the second largest eigenvalue and $\mathbf{c}_2$ is the corresponding eigenvector. Similarly, the other PCs can be shown to be eigenvectors of C corresponding to the subsequently next largest eigenvalues. Because C is a real, symmetric matrix, we know all its eigenvectors will be orthogonal with respect to each other. This further clarifies the fact that we can interpret the whole operation as a coordinate rotation.

6.4.3 KLT: PCA-based image compression

It has already been mentioned that the KLT is simply applying PCA to complete images. The KLT will remove the redundancy in simple pixel based description that derives from the strong correlations that exist between pixel intensities.

The procedure would look as follows: consider the image as a large vector of pixel intensities, calculate the covariance statistics over the set of images considered, and look for the eigenvalues and eigenvectors of the covariance matrix. The eigenvalues can be interpreted as images, so-called *eigenimages* or in more general terms *basis images*. Combining only the eigenimages with the largest eigenvalues – with weights found by correlation as will be explained later – will in general yield an acceptable approximation to the original image. However, there are at least three important problems with this approach:

Computational problems Applying PCA on $N \times N$ images by interpreting them as $N^2 \times 1$ pixel intensity vectors leads to the calculation and analysis of $N^2 \times N^2$ covariance matrices! These are by all means huge matrices, even for a modern computer. As an example, 65.536×65.536 covariance matrices have to be dealt with for 256×256 images, with such image dimensions still lower than what is usual.

Specifying image statistics The availability of the necessary statistics implies that these can be calculated over a set of representative samples (i.e. a family of images). In many applications image content can take all sorts of forms, making the extraction of representative statistics a difficult if not impossible task. Therefore the method primarily makes sense when dealing with images that all look pretty similar, e.g. a series of head-shoulder portrait photographs in a database for person identification.

Image dependence The fact that the statistics are extracted from a particular set of images also implies that the resulting transform will only be finetuned towards use for this set. This data dependence means that the compression advantage is largely lost if several image families have to be considered since the eigenvectors (images!) for the retained PCs will have to be stored or transmitted.

It follows from the characteristics of PCA that the KLT is the linear transform with the best energy compaction, i.e. it redistributes variance (energy) over a few components most efficiently and thereby yields the minimal least-square error for truncated approximations (cfr. section 6.2.1).

As already mentioned, one can rightfully question the applicability of this technique, since it entails the calculation and analysis of huge covariance matrices. This objection can be remedied, however. A first important consideration is that the number of images used to extract the statistics will typically be much smaller than $p = N^2$, the dimension of "image space". For the sake of argument, consider e.g. working in a three-dimensional space, i.e. $p = 3$. If our training set contains but two samples, then we can draw a plane through the corresponding points in the three-dimensional space and the origin. We can carry out the analysis in a two-dimensional rather than a three-dimensional space, since the sample set clearly has no variance in a direction perpendicular to that plane. Similarly,

extracting statistics from a set of n images, with n typically much lower than $p = N^2$, all samples lie in a n-dimensional subspace (hyperplane) of the p-dimensional image space. Thus, all eigenvectors capturing any variance on the data must remain within the subspace. As known from algebra, the eigenvectors can be used as columns in a linear transformation matrix that diagonalizes the original matrix, i.e. the covariance matrix. The eigenvalues will figure as the elements on the diagonal. Since the eigenvalues represent the variance in the corresponding directions, all directions orthogonal to the n-dimensional hyperplane will have eigenvalue zero and will certainly not produce the eigenvectors with the largest variance. Since there will be many such directions, the determinant of the covariance matrix, found as the product of the eigenvalues, vanishes. We conclude that as soon as $n < p$ the complete $p \times p$ covariance matrix is singular and cannot have rank higher than n.
Fortunately, this means that we can simplify the problem, as there exists a shortcut in the derivation of the first n principal components. Consider the $(p \times n)$ data matrix $\mathbf{X}$ that contains the n sample vectors. We again assume that these samples have been normalised towards zero-mean components. The $(p \times p)$ sample covariance matrix is given by $C = \frac{1}{n}\mathbf{X}\mathbf{X}^T$, Then, if we look for the eigenvectors $\mathbf{c}_i$ of the much smaller $(n \times n)$ matrix $S = \mathbf{X}^T\mathbf{X}$ instead:

$$\mathbf{X}^T\mathbf{X}\mathbf{c}_i = \lambda_i \mathbf{c}_i \ .$$

Multiplying both sides from the left with $\mathbf{X}$ yields

$$\mathbf{X}\mathbf{X}^T\mathbf{X}\mathbf{c}_i = \mathbf{X}\lambda_i\mathbf{c}_i$$

and hence

$$\left(\frac{1}{n}\mathbf{X}\mathbf{X}^T\right)(\mathbf{X}\mathbf{c}_i) = \frac{\lambda_i}{n}(\mathbf{X}\mathbf{c}_i) \ ,$$

implying that $\mathbf{X}\mathbf{c}_i$ is an eigenvector with eigenvalue λ_i/n of C if $\mathbf{c}_i$ is an eigenvector with eigenvalue λ_i of S. Hence, we can find the meaningful, first n eigenvectors of C as well as their eigenvalues without actually having to deal with its huge dimensions.

6.5 II - Independent Component Analysis

6.5.1 Rationale

Principal Component Analysis was introduced as a means of reducing the dependency between features or images. When taking images as an example, PCA generates a set of basis images (eigenimages) that are uncorrelated, i.e. linearly independent. There are applications where one wants to decompose images, not because one wants to compress them, but because the observed images are compositions of different images, in this context usually called source images, that one would like to know. We will give examples later, but think of an image of a shop window in which one has the superposition of a reflection in the window and what's behind. One would like to get an image of what's behind, without the reflection. Such decomposition is the goal of Independent Component Analysis (ICA).

Unfortunately, the decorrelation of random variables via PCA is often not sufficient. To give a simple example, let us consider a random vector $\mathbf{x}$ consisting of the variables $x_1 = \sin(t)$ and $x_2 = \cos(t)$. The random variable $t \in [0 \ldots 2\pi)$ shall be equally distributed. Since $\mathbf{x}$ describes points on the unit circle, every direction can serve as a principal direction and the components are obviously uncorrelated:

$$\mathrm{Cov}\{\mathbf{x}\} = E\{\mathbf{x}\mathbf{x}^T\} = E\{\begin{pmatrix} \sin(t)^2 & \sin(t)\cos(t) \\ \sin(t)\cos(t) & \cos(t)^2 \end{pmatrix}\} = \begin{pmatrix} \frac{1}{2} & 0 \\ 0 & \frac{1}{2} \end{pmatrix} \tag{6.2}$$

Nevertheless, as this example shows, the random variables $\mathbf{x}_1$ and $\mathbf{x}_2$ are definitely not independent.

Since correlation is a second order moment, decorrelation represents only independence up to the second order. Realising statistical independence can be considered as a generalisation of decorrelation that removes only linear dependencies. For several applications, it would be very desirable to find a transformation that removes as much statistical dependence of a random vector's components as possible. An example is blind source separation. Suppose you are at a trade fair. Different exhibitors try to lure you towards their booth by playing music, through exhalted talk about the great features of the product, etc. Suppose now you would tape the noise simultaneously at different positions. ICA is a means to use the different recordings to separate them into the noise made by each of the exhibitors separately. The same can be done for images: if a number of images are different mixes of underlying patterns, these separate patterns can be extracted using ICA. Practical applications are discussed in section 6.6. The different, underlying signals can be expected to be statistically independent in many cases. As mentioned earlier this is the assumption that ICA exploits.

First, let us recall what statistically independent means. Let $\mathbf{c}$ be a random vector with values in $\mathbb{R}^n$ and with the probability density function $p(\mathbf{c})$. The vector $\mathbf{c}$ has mutually independent components c_i, if and only if

$$p(\mathbf{c}) = \Pi_i p_i(c_i) \tag{6.3}$$

with $p_i()$ the probability density function of the ith component of $\mathbf{c}$ taken separately. A natural way of measuring the dependence between a random vector's components is therefore to measure the "distance" between its joint density $p(\mathbf{c})$ and the independent density $\Pi_i p_i(c_i)$. One possibility to do so is given by the Kullback-divergence:

$$\delta(p_\mathbf{x}, p_\mathbf{y}) = \int p_\mathbf{x}(\mathbf{c}) \log(\frac{p_\mathbf{x}(\mathbf{c})}{p_\mathbf{y}(\mathbf{c})})\, d\mathbf{c} \tag{6.4}$$

If we employ the Kullback-divergence to measure the distance between the left and right side of Eq. (6.3), we end up with an expression that is very familiar in information theory, the mutual information I:

$$I(\mathbf{c}) = \int p(\mathbf{c}) \log(\frac{p(\mathbf{c})}{\Pi_i p_i(c_i)})\, d\mathbf{c} \tag{6.5}$$

It is well known that the mutual information I vanishes, if and only if the variables c_i are mutually independent, and is strictly positive otherwise. Hence the mutual information seems to be an appropriate measure for our purposes. If we take into account that the differential entropy H is given by

$$H(\mathbf{c}) = -\int p(\mathbf{c}) \log(p(\mathbf{c}))\, d\mathbf{c} \,, \tag{6.6}$$

and

$$H(c_i) = -\int p_i(c_i) \log(p_i(c_i))\, dc_i \,, \tag{6.7}$$

we can reformulate the mutual information in terms of differential entropy, as is common in information theory:

$$I(\mathbf{c}) = \sum_i H(c_i) \; - H(\mathbf{c}) \tag{6.8}$$

6.5.2 Outline of the algorithm

Suppose one has a series of images $i_1(x,y)$, ..., $i_m(x,y)$ that are linear combinations of underlying patterns $u_1(x,y)$, ..., $u_n(x,y)$. Hence, there is a $m \times n$ matrix A such that

$$\mathbf{i} = A\mathbf{u} \,,$$

where $\mathbf{i}$ is the m-dimensional vector $(i_1, ..., i_m)^T$ and $\mathbf{u}$ the n-dimensional vector $(u_1, ..., u_n)^T$. Given only the images $\mathbf{i}$ the task is to retrieve the underlying patterns $\mathbf{u}$. In this paper we will assume that $m = n$.

As indicated, ICA solves this task by assuming that the original patterns are independent, i.e. it finds linear combinations

$$\mathbf{c} = M\mathbf{i} \tag{6.9}$$

of the images with minimal mutual information:

$$I(c_1, ..., c_n) = \sum_{i=1}^{n} H(c_i) - H(\mathbf{c}) \tag{6.10}$$

where $H()$ again stands for the differential entropy

$$H(\mathbf{c}) = -\int p(\mathbf{c}) \log p(\mathbf{c}) d\mathbf{u} \,,$$

with $p()$ the density distribution. Given eq. (6.9) the mutual information (6.10) can be rewritten as

$$I(c_1, ..., c_n) = \sum_{i=1}^{n} H(c_i) - H(\mathbf{i}) - \log \mathrm{abs}(|M|) \,,$$

with $|M|$ the determinant of the matrix M. Mutual information is zero when $p(\mathbf{c}) = \Pi_i p(c_i)$. The second term does not depend on M. This equation illustrates that minimizing

mutual information is related to finding patterns that have minimal differential entropies. As Gaussian distributions have maximum entropy (for a given variance), this also implies that one can view this as a search for maximally non-Gaussian patterns. This becomes clearer when mutual information is expressed in terms of negentropy. Negentropy J is defined as

$$J(\mathbf{c}) = H(\mathbf{c}_G) - H(\mathbf{c})$$

where $\mathbf{c}_G$ is a Gaussian random variable of the same covariance matrix as $\mathbf{c}$. It quantifies the deviation from a Gaussian density distribution. Mutual information can be rewritten as

$$I(c_1, ..., c_n) = J(\mathbf{c}) - \sum_{i=1}^{n} J(c_i) + \frac{1}{2} \log \frac{\Pi C(i,i)}{\det C}$$

where C is the covariance matrix of $\mathbf{c}$. Note that if the c_i are uncorrelated, then the last term vanishes. This is a very reasonable constraint to impose on patterns that are supposed to be independent. Moreover, as negentropy is invariant under invertible linear transformations, $J(\mathbf{c})$ is the same for any choice of linear combinations. Consequently, minimizing mutual information amounts to finding linear combinations with maximum negentropy or, equivalently, that are maximally non-Gaussian.

The algorithm can be summarized as follows, following the approach proposed by Hyvärinen:

Step 1 The intensities of the images $i_1, ..., i_m$ are shifted as to obtain zero average intensity images $\bar{i_1}, ..., \bar{i_m}$.

Step 2 A principal component analysis (PCA) is performed on the images $\bar{i_1}, ..., \bar{i_m}$. Tbe principal components yield decorrelated data and establish the dimension n to be used (only dominant components are taken into account). This step amounts to a rotation.

Step 3 The data are sphered, i.e. the principal components are rescaled to all have unit variance. The inverse of the square roots of the eigenvalues that result from PCA yield the necessary scaling factors. This is an anisotropic scaling.

Step 4 Once the data are sphered, further rotations keep the components uncorrelated. This step will reorient the components as to maximize their negentropy (i.e. minimize the mutual information). This again is a rotation.

The combination rotation - anisotropic scaling - rotation amounts to a general linear transformation. In contrast to PCA, ICA takes linear combinations that no longer are orthogonal. Next, step 4 is described in more detail. In the sequel $\bar{i_1}, ..., \bar{i_m}$ will simply be written as $i_1, ..., i_m$, assuming the images have zero mean.

For methods that allow one the actually compute the independent components, we refer to the specialised literature. Finding ICs is more difficult than finding PCs, and the problems already start with the efficient calculation of differential entropy and negentropy. Several implementations will not actually maximise negentropy, but a function that is

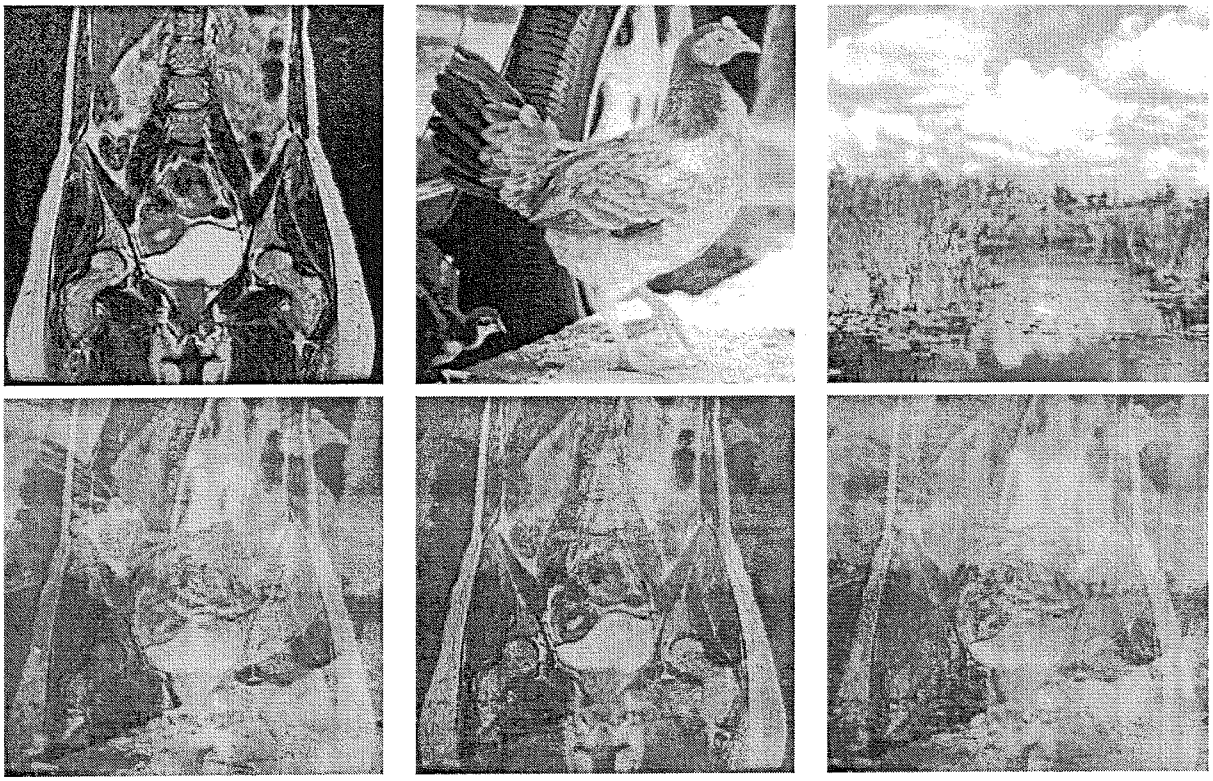

Figure 6.14: *First row: original images. Second row: three different linear combinations formed with the images of the first row.*

simpler to evaluate and the value of which increases with that of negentropy. We mainly wanted to introduce the concept behind ICA. In order to clarify that further, the next section shows an example.

6.5.3 Example

Fig. 6.14 and fig. 6.15 illustrate the algorithm for a synthetic example. Of the three images in the top row of fig. 6.14, three linear combinations were formed that are shown in the second row. Even for a human observer it is not obvious to recognize the underlying patterns in these images. Fig. 6.15 shows the results of the independent component analysis. The first row of this figure shows the principal components that were obtained from these linear combinations (step 2). The second row shows the independent components, obtained after sphering and further rotations to align the linear combinations with directions of maximal negentropy. The independent components correspond almost perfectly to the input images for this idealised example (idealised in the sense that the input images are three perfect linear combinations).

To illustrate that ICA goes after linear combinations of the input that have maximally non-Gaussian distributions, it is interesting to look at the histograms. Fig. 6.16 shows in its top row the histograms of the three input images (the linear combinations) and in

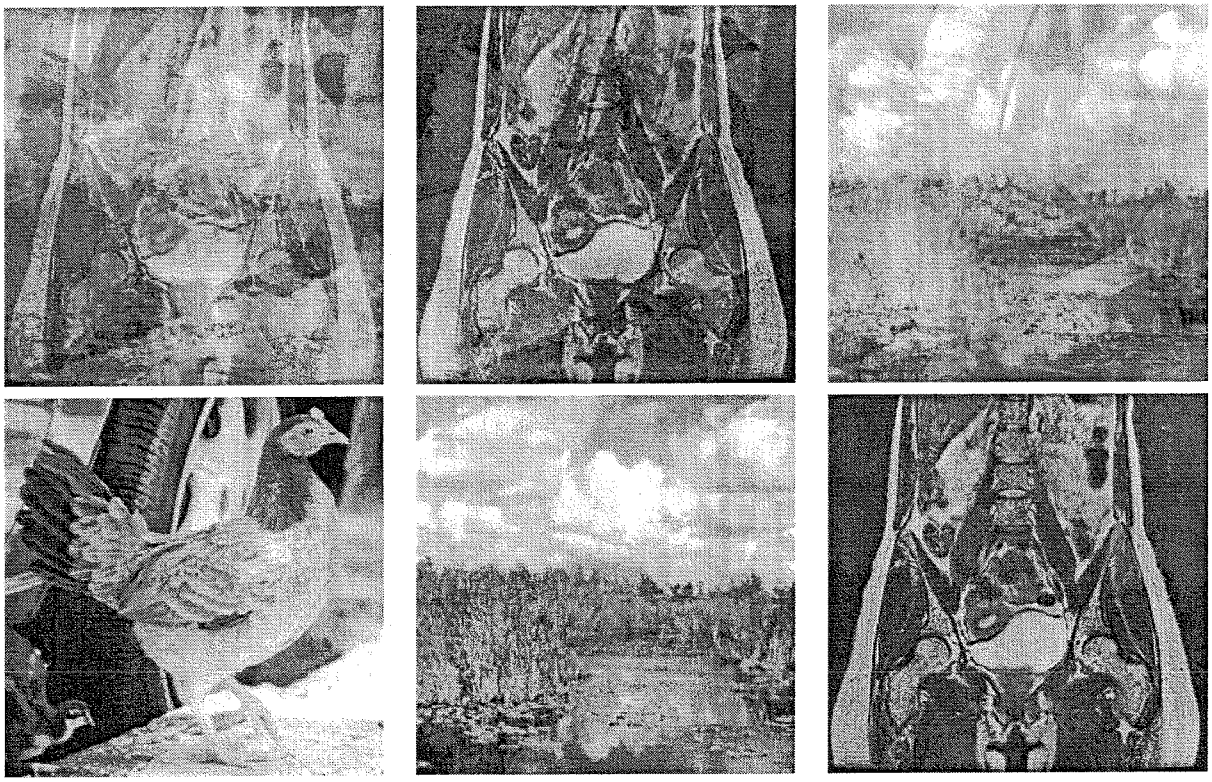

Figure 6.15: *First row: decorrelated images obrained as principal components. Second row: independent components found by the algorithm.*

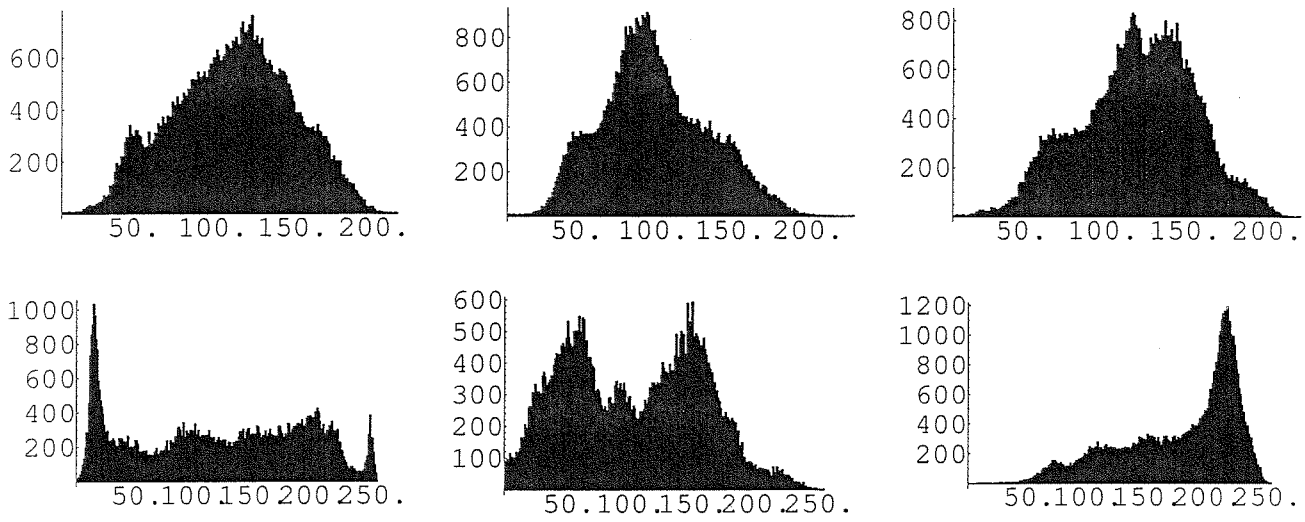

Figure 6.16: *First row: histograms of the input images in fig. 6.14. Second row: histograms for the independent component images.*

its bottom row the histograms of the independent components (the retrieved underlying patterns). Their distributions are less Gaussian than those of the input images.

6.6 Some applications of ICA

Example 1: subtracting window reflections

Events of interest may happen behind glass in which there is a disturbing reflection from the scene outside. It may for instance be important to know until what time an item was still present in a shop window. Alternatively, one might be interested in the reflection, e.g. of a suspicious car parked on the street. In such cases one would like to disentangle the images of the scene behind the glass and the reflected image. In order to achieve this with independent component analysis, one needs different, linear mixtures of the two.

Fortunately, throughout day and night the relative amounts of background and reflection tend to change. Light sources may be turned on or off, cars can pass adding illumination to the scene at night, the sun changes its position, etc. Hence, images taken with the same (surveillance) camera at different times can be used as input to ICA.

Fig. 6.17 shows two images of a scene with a text on the background and the reflection of a mannequin. The two images were taken with one light source less for the second. Image quality has been kept low, as is usually the case for surveillance video data. Fig. 6.18 shows the resulting independent components. Background and reflection have been separated quite well. Only the mannequin face is reflected, all the rest belongs to the scene behind the window. Independent component analysis has correctly separated the two.

Nevertheless, the separation is not as good as for the synthetic example. The compo-



Figure 6.17: *Input images. The face of a mannequin reflects in a glass window, behind which there is a text.*



Figure 6.18: *The independent component images calculated from the pair of images in fig. 6.17.*



Figure 6.19: *The input on the left shows a 5 pound banknote, with a fingerprint. The figure on the right shows a clean banknote, after registration with the image on the left.*

sition of the two patterns is not purely linear. In general, one can expect problems with non-linear sensitivities of the camera. CCD images can be expected to yield better results than those taken with vacuum tube cameras. But even then, problems can be expected with darker or brighter parts of the scene, where saturation occurs.

One of the nice features of ICA is that this is done completely automatically, without the need for adjusting parameter settings. This means that the method is 'objective' and that it can be used by police staff without a thorough training in the underlying mathematics.

Example 2: extracting prints from complicated backgrounds

A problem that often occurs in forensic image processing is to extract patterns from a complex, printed background. A good example are fingerpints on documents such as banknotes. In the past, it was in fact only possible to remove periodic and simple backgrounds. Many of these techniques were based on filtering in Fourier space. First, one has to align the background patterns of the exhibit with the mark and a clean specimen. After that step ICA can be used to separate the patterns.

Fig. 6.19 shows two 5-pound banknotes, one with a fingerprint and a clean specimen. The two have already been aligned with the method of Capel *et al.* from the University of Oxford. This method deforms one pattern with respect to the other on the basis of local projectivities, calculated from the positions of corresponding landmarks in the patterns. ICA yields the components shown in fig. 6.20. Finally, after a simple contrast enhancement, we obtain the result of fig. 6.21. Note that also a second, less salient fingerprint is visible in this result. The presence of mainly the printed text in both components has to do with



Figure 6.20: *The independent components found for the input images of fig. 6.19.*



Figure 6.21: *The independent component with the fingerprint after contrast enhancement.*

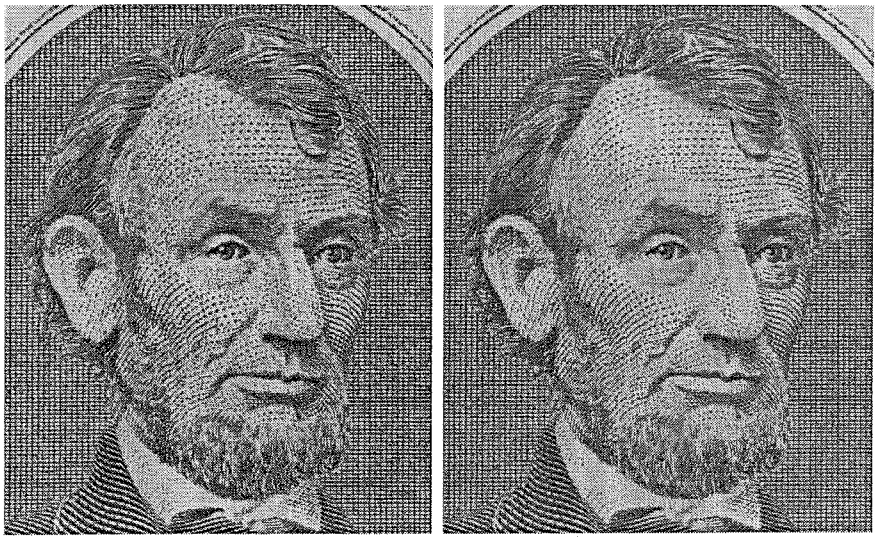

Figure 6.22: *The input on the left shows part of a US dollar banknote, with a fingerprint. The figure on the right shows a clean banknote, after registration with the image on the left.*

the fact that this is printed separately on banknotes and hence cannot be brought into perfect registration together with the rest of the banknote background.

Fig. 6.22 shows two dollar banknotes, again one with a fingerprint (left) and a clean bill (right). The two have already been aligned, again with the method of Capel *et al.*. ICA yields the components shown in fig. 6.23. Finally, after a simple contrast enhancement, we obtain the result of fig. 6.24.

6.7 Further reading

not available yet

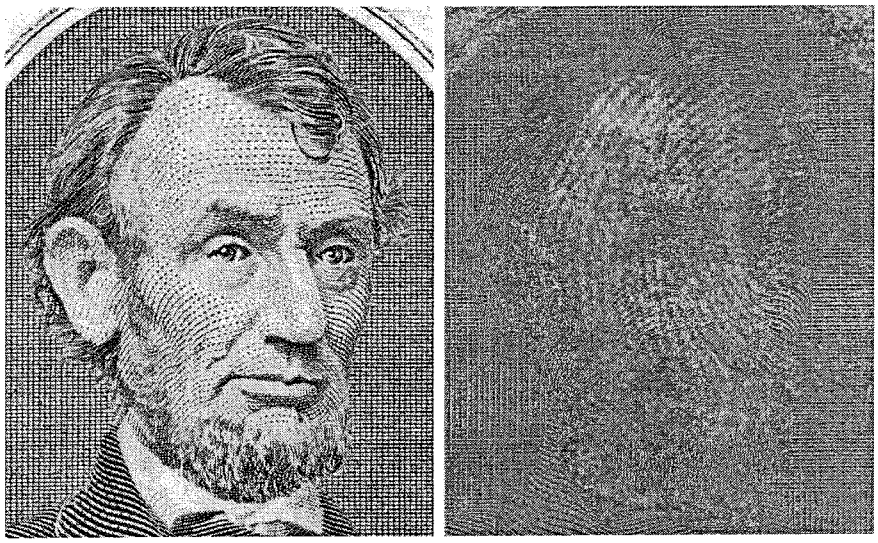

Figure 6.23: *The independent components found for the input images of fig. 6.22.*



Figure 6.24: *The independent component with the fingerprint, after contrast enhancement.*