DIP 3

Image enhancement

Image Enhancement

- to improve the interpretability or perception of information in images for human viewers, or to provide `better' input for other automated image processing techniques.

Image enhancement techniques can be divided into two broad categories:

1. 1Spatial domain methods, which operate directly on pixels, and
2. frequency domain methods, which operate on the Fourier transform of an image.

Spatial Domain Methods

The value of a pixel with coordinates (x,y) in the enhanced image $\hat{F}$ is the result of performing some operation on the pixels in the neighbourhood of (x,y) in the input image, F.

Neighbourhoods can be any shape, but usually they are rectangular.

The simplest form of operation is when the operator T only acts on a 1×1 pixel neighbourhood in the input image, that is $\hat{F}(x,y)$ only depends on the value of F at (x,y). This is a grey scale transformation or mapping.

The simplest case is thresholding where the intensity profile is replaced by a step function, active at a chosen threshold value. In this case any pixel with a grey level below the threshold in the input image gets mapped to 0 in the output image. Other pixels are mapped to 255.
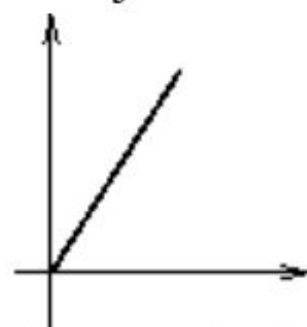
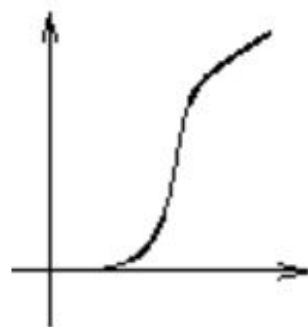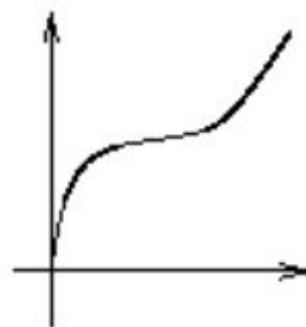**Figure 1:** Tone-scale adjustments.
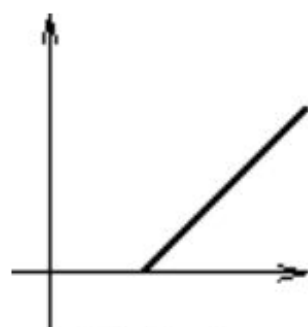
Darkening Lightening Compressed to darks Compressed to lights
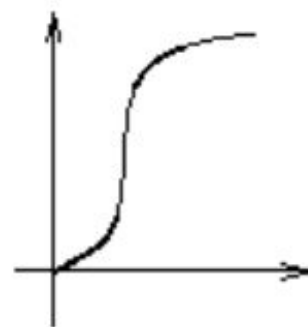
High contrast Low constrast Emphasize shadows Emphasize lights

Histogram Equalizer

Histogram equalization is a common technique for enhancing the appearance of images. Suppose we have an image which is predominantly dark. Then its histogram would be skewed towards the lower end of the grey scale and all the image detail is compressed into the dark end of the histogram. If we could `stretch out' the grey levels at the dark end to produce a more uniformly distributed histogram then the image would become much clearer.

Histogram equalization involves finding a grey scale transformation function that creates an output image with a uniform histogram (or nearly so).

Figure 2: The original image and its histogram, and the equalized versions. Both images are quantized to 64 grey levels.
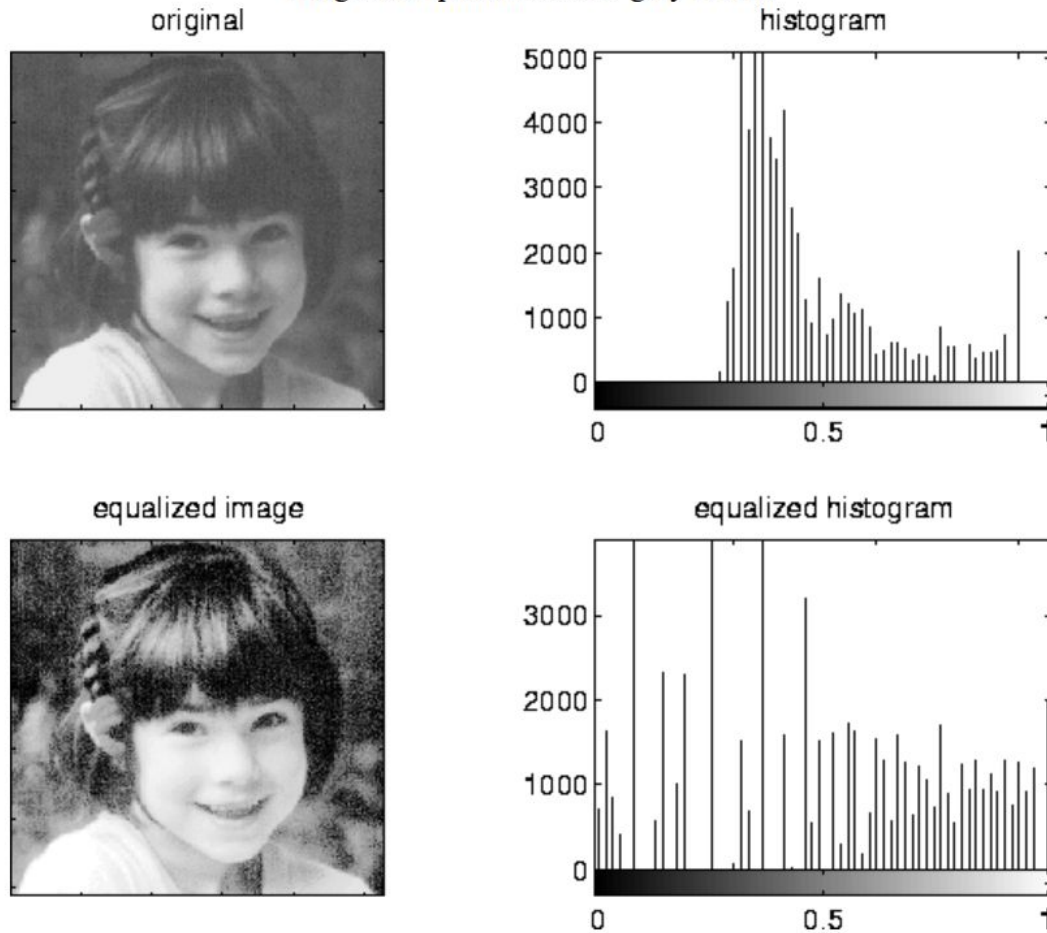
How do we determine this grey scale transformation function? Assume our grey levels are continuous and have been normalized to lie between 0 and 1.

We must find a transformation T that maps grey values r in the input image F to grey values $s = T(r)$ in the transformed image $\hat{F}$.

It is assumed that

- T is single valued and monotonically increasing, and
- $0 \leq T(r) \leq 1$ for $0 \leq r \leq 1$.

The inverse transformation from s to r is given by

$$r = T^{-1}(s).$$

If one takes the histogram for the input image and normalizes it so that the area under the histogram is 1, we have a probability distribution for grey levels in the input image $P_r(r)$.

If we transform the input image to get $s = T(r)$ what is the probability distribution $P_s(s)$?

From probability theory it turns out that

$$P_s(s) = P_r(r)\frac{dr}{ds},$$

where $r = T^{-1}(s)$.

Consider the transformation

$$s = T(r) = \int_0^r P_r(w)dw.$$

This is the cumulative distribution function of r. Using this definition of T we see that the derivative of s with respect to r is

$$\frac{ds}{dr} = P_r(r).$$

Substituting this back into the expression for P_s, we get

$$P_s(s) = P_r(r)\frac{1}{P_r(r)} = 1$$

for all $s, where 0 \leq s \leq 1$. Thus, $P_s(s)$ is now a uniform distribution function, which is

what we want.

Discrete Formulation

We first need to determine the probability distribution of grey levels in the input image. Now

$$P_r(r) = \frac{n_k}{N}$$

where n_k is the number of pixels having grey level k, and N is the total number of pixels in the image.

The transformation now becomes

$$
s_k = T(r_k) = \sum_{i=0}^{k} \frac{n_i}{N}
$$

$$
= \sum_{i=0}^{k} P_r(r_i).
$$

Note that $0 \le r_k \le 1$, the index $k = 0, 1, 2, \dots, 255$,and $0 \le s_k \le 1$.

The values of s_k will have to be scaled up by 255 and rounded to the nearest integer so that the output values of this transformation will range from 0 to 255. Thus the discretization and rounding of s_k to the nearest integer will mean that the transformed image will not have a perfectly uniform histogram.

Image Smoothening

The aim of image smoothing is to diminish the effects of camera noise, spurious pixel values, missing pixel values etc. There are many different techniques for image smoothing; we will consider neighbourhood averaging and edge-preserving smoothing.

Neighbourhood Averaging

Each point in the smoothed image, $\hat{F}(x, y)$ is obtained from the average pixel value in a neighbourhood of (x,y) in the input image.

For example, if we use a 3×3 neighbourhood around each pixel we would use the mask

$$\begin{array}{ccc} 1/9 & 1/9 & 1/9 \\ 1/9 & 1/9 & 1/9 \\ 1/9 & 1/9 & 1/9 \end{array}$$

Each pixel value is multiplied by $\frac{1}{9}$, summed, and then the result placed in the output image.

This mask is successively moved across the image until every pixel has been covered. That is, the image is *convolved* with this smoothing mask (also known as a spatial filter or kernel).

However, one usually expects the value of a pixel to be more closely related to the values of pixels close to it than to those further away. This is because most points in an image are spatially coherent with their neighbours; indeed it is generally only at edge or feature points where this hypothesis is not valid. Accordingly it is usual to weight the pixels near the centre of the mask more strongly than those at the edge.

Some common weighting functions include the rectangular weighting function above (which just takes the average over the window), a triangular weighting function, or a Gaussian.

In practice one doesn't notice much difference between different weighting functions, although Gaussian smoothing is the most commonly used. Gaussian smoothing has the attribute that the frequency components of the image are modified in a smooth manner.

Smoothing reduces or attenuates the higher frequencies in the image. Mask shapes other than the Gaussian can do odd things to the frequency spectrum, but as far as the appearance of the image is concerned we usually don't notice much.

Edge Preserving Smoothing

Neighbourhood averaging or Gaussian smoothing will tend to blur edges because the high frequencies in the image are attenuated. An alternative approach is to use median filtering. Here we set the grey level to be the median of the pixel values in the neighbourhood of that pixel.

The median m of a set of values is such that half the values in the set are less than m and half are greater. For example, suppose the pixel values in a 3×3 neighbourhood are (10, 20, 20, 15, 20, 20, 20, 25, 100). If we sort the values we get (10, 15, 20, 20, |20|, 20, 20, 25, 100) and the median here is 20.

The outcome of median filtering is that pixels with outlying values are forced to become more like their neighbours, but at the same time edges are preserved. Of course, median filters are non-linear.

Median filtering is in fact a morphological operation. When we erode an image, pixel values are replaced with the smallest value in the neighbourhood. Dilating an image corresponds to replacing pixel values with the largest value in the neighbourhood. Median filtering replaces pixels with the median value in the neighbourhood. It is the rank of the value of the pixel used in the neighbourhood that determines the type of morphological operation.

Figure 3: Image of Genevieve; with salt and pepper noise; the result of averaging; and the result of median filtering.

original

added noise

average

median

Image Sharpening

The main aim in image sharpening is to highlight fine detail in the image, or to enhance detail that has been blurred (perhaps due to noise or other effects, such as motion). With image sharpening, we want to enhance the high-frequency components; this implies a spatial filter shape that has a high positive component at the centre (see figure 4 in the next slide).

Figure 4: Frequency domain filters (top) and their corresponding spatial domain counterparts (bottom).
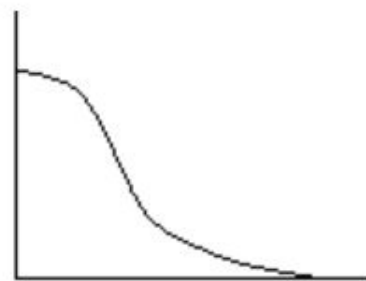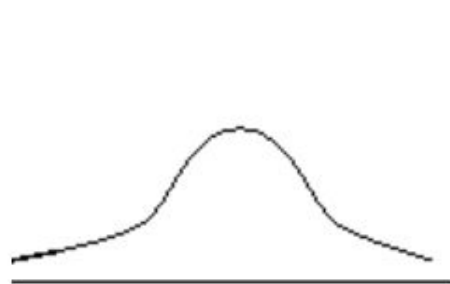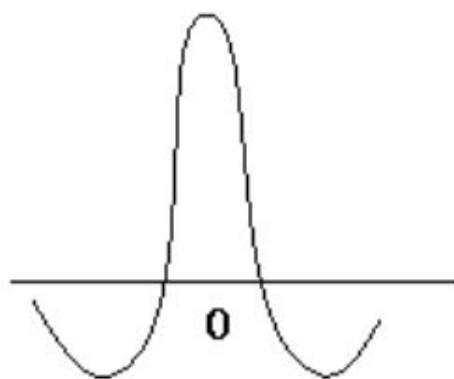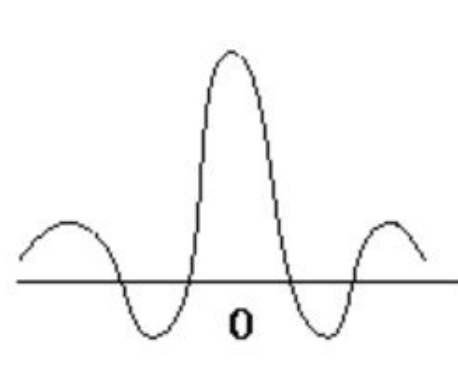
A simple spatial filter that achieves image sharpening is given by

$$-1/9 \quad -1/9 \quad -1/9$$
$$-1/9 \quad 8/9 \quad -1/9$$
$$-1/9 \quad -1/9 \quad -1/9$$

Since the sum of all the weights is zero, the resulting signal will have a zero DC value (that is, the average signal value, or the coefficient of the zero frequency term in the Fourier expansion). For display purposes, we might want to add an offset to keep the result in the $0\ldots255$ range.

High boost filtering

We can think of high pass filtering in terms of subtracting a low pass image from the original image, that is,

$$\text{High pass} = \text{Original} - \text{Low pass}.$$

However, in many cases where a high pass image is required, we also want to retain some of the low frequency components to aid in the interpretation of the image. Thus, if we multiply the original image by an amplification factor A before subtracting the low pass image, we will get a *high boost* or *high frequency emphasis* filter. Thus,

$$
\begin{aligned}
\textbf{High boost} \ &= \ A.\text{Original} - \textbf{Low pass} \\
&= \ (A-1).(\text{Original}) + \text{Original} - \textbf{Low pass} \\
&= \ (A-1).\text{Original} + \textbf{High pass}.
\end{aligned}
$$

Now, if $A = 1$ we have a simple high pass filter. When $A > 1$ part of the original image is retained in the output.

A simple filter for high boost filtering is given by

$$
\begin{array}{ccc}
-1/9 & -1/9 & -1/9 \\
-1/9 & \omega/9 & -1/9 \\
-1/9 & -1/9 & -1/9
\end{array}
$$

where $\omega = 9A - 1$.

Frequency Domain Method

Image enhancement in the frequency domain is straightforward. We simply compute the Fourier transform of the image to be enhanced, multiply the result by a filter (rather than convolve in the spatial domain), and take the inverse transform to produce the enhanced image.

The idea of blurring an image by reducing its high frequency components, or sharpening an image by increasing the magnitude of its high frequency components is intuitively easy to understand. However, computationally, it is often more efficient to implement these operations as convolutions by small spatial filters in the spatial domain. Understanding frequency domain concepts is important, and leads to enhancement techniques that might not have been thought of by restricting attention to the spatial domain.

Filtering

Low pass filtering involves the elimination of the high frequency components in the image. It results in blurring of the image (and thus a reduction in sharp transitions associated with noise). An ideal low pass filter would retain all the low frequency components, and eliminate all the high frequency components. However, ideal filters suffer from two problems: *blurring* and *ringing*. These problems are caused by the shape of the associated spatial domain filter, which has a large number of undulations. Smoother transitions in the frequency domain filter, such as the Butterworth filter, achieve much better results.

Figure 5: Transfer function for an ideal low pass filter.

.deal frequency low pass filter   spatial domain counterpart

Homomorphic Filtering

Images normally consist of light reflected from objects. The basic nature of the image F(x,y) may be characterized by two components: (1) the amount of source light incident on the scene being viewed, and (2) the amount of light reflected by the objects in the scene. These portions of light are called the illumination and reflectance components, and are denoted i(x,y) and r(x,y) respectively. The functions i and r combine multiplicatively to give the image function F:

F(x,y) = i(x,y)r(x,y),

where $0 < i(x,y) < \infty$ and $0 < r(x,y) < 1$.   We cannot easily use the above product to operate separately on the frequency components of illumination and reflection because the Fourier transform of the product of two functions is not separable; that is

$$\mathcal{F}(F(x,y)) \neq \mathcal{F}(i(x,y))\mathcal{F}(r(x,y)).$$

Suppose, however, that we define

$$
\begin{aligned}
z(x,y) &= \ln F(x,y) \\
&= \ln i(x,y) + \ln r(x,y).
\end{aligned}
$$

Then

$$
\begin{aligned}
\mathcal{F}(z(x,y)) &= \mathcal{F}(\ln F(x,y)) \\
&= \mathcal{F}(\ln i(x,y)) + \mathcal{F}(\ln r(x,y))
\end{aligned}
$$

or

$$Z(\omega, \nu) = I(\omega, \nu) + R(\omega, \nu),$$

where Z, I and R are the Fourier transforms of z, $\ln i$ and $\ln r$ respectively. The function Z

represents the Fourier transform of the *sum* of two images: a low frequency illumination image and a high frequency reflectance image.

Figure 6: Transfer function for homomorphic filtering.
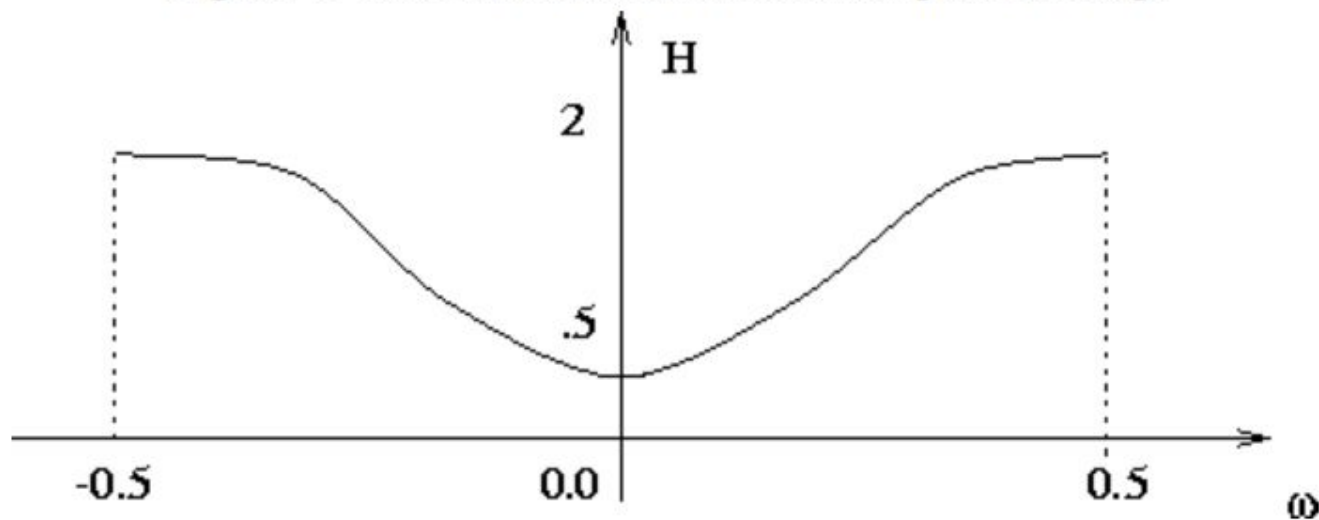
If we now apply a filter with a transfer function that suppresses low frequency components and enhances high frequency components, then we can suppress the illumination component and enhance the reflectance component. Thus

$$S(\omega, \nu) = H(\omega, \nu)Z(\omega, \nu)$$
$$= H(\omega, \nu)I(\omega, \nu) + H(\omega, \nu)R(\omega, \nu),$$

where S is the Fourier transform of the result. In the spatial domain

$$s(x, y) = \mathcal{F}^{-1}(S(\omega, \nu))$$
$$= \mathcal{F}^{-1}(H(\omega, \nu)I(\omega, \nu)) + \mathcal{F}^{-1}(H(\omega, \nu)R(\omega, \nu)).$$

By letting

$$i'(x, y) = \mathcal{F}^{-1}(H(\omega, \nu)I(\omega, \nu))$$

and

$$r'(x, y) = \mathcal{F}^{-1}(H(\omega, \nu)R(\omega, \nu))$$

$$s(x,y) = i'(x,y) + r'(x,y).$$

Finally, as z was obtained by taking the logarithm of the original image F, the inverse yields the desired enhanced image $\hat{F}$: that is

$$
\begin{aligned}
\hat{F}(x, y) &= exp[s(x, y)] \\
&= exp[i'(x, y)]exp[r'(x, y)] \\
&= i_0(x, y)r_0(x, y).
\end{aligned}
$$

Thus, the process of homomorphic filtering can be summarized by the following diagram:

Figure 7: The process of homomorphic filtering.

$$f(x,y) \dashrightarrow \boxed{\ln} \dashrightarrow \boxed{FFT} \dashrightarrow \boxed{H} \dashrightarrow \boxed{FFT^{-1}} \dashrightarrow \boxed{exp} \dashrightarrow g(x,y)$$
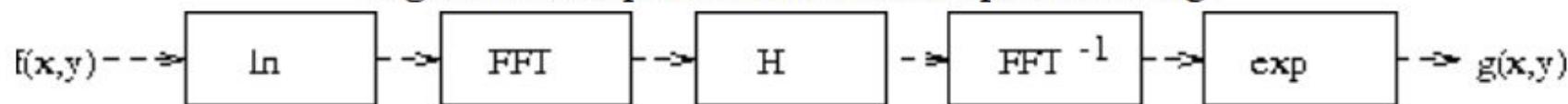
Geometric Transformations

In this section we consider image transformations such as rotation, scaling and distortion (or undistortion!) of images. Such transformations are frequently used as pre-processing steps in applications such as document understanding, where the scanned image may be mis-aligned.

There are two basic steps in geometric transformations:

1. A spatial transformation of the physical rearrangement of pixels in the image, and

2. a grey level interpolation, which assigns grey levels to the transformed image

Spatial transformation

Pixel coordinates (x,y) undergo geometric distortion to produce an image with coordinates (x',y'):

$$x' = r(x, y)$$
$$y' = s(x, y),$$

where r and s are functions depending on x and y.

Examples:

1.

Suppose $r(x, y) = \frac{x}{2}$, $s(x, y) = \frac{y}{2}$. This halves the size of the image. This transformation can be represented using a matrix equation

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \frac{1}{2} & 0 \\ 0 & \frac{1}{2} \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

2.

Rotation about the origin by an angle θ is given by

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

Remember the origin of the image is usually the top left hand corner. To rotate about the centre one needs to do a transformation of the origin to the centre of the image first.

Tie points

Often the spatial transformation needed to correct an image is determined through tie points. These are points in the distorted image for which we know their corrected positions in the final image. Such tie points are often known for satellite images and aerial photos. We will illustrate this concept with the example of correcting a distorted quadrilateral region in an image.

We model such a distortion using a pair of bilinear equations:

$$x' = c_1 x + c_2 y + c_3 xy + c_4$$

$$y' = c_5 x + c_6 y + c_7 xy + c_8.$$

We have 4 pairs of tie point coordinates. This enables us to solve for the 8 coefficients $c_1 \ldots c_8$.

We can set up the matrix equation using the coordinates of the 4 tie points:

$$
\begin{bmatrix}
x'_1 \\
y'_1 \\
x'_2 \\
y'_2 \\
x'_3 \\
y'_3 \\
x'_4 \\
y'_4
\end{bmatrix}
=
\begin{bmatrix}
x_1 & y_1 & x_1y_1 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & x_1 & y_1 & x_1y_1 & 1 \\
x_2 & y_2 & x_2y_2 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & x_2 & y_2 & x_2y_2 & 1 \\
x_3 & y_3 & x_3y_3 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & x_3 & y_3 & x_3y_3 & 1 \\
x_4 & y_4 & x_4y_4 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & x_4 & y_4 & x_4y_4 & 1
\end{bmatrix}
\begin{bmatrix}
c_1 \\
c_2 \\
c_3 \\
c_4 \\
c_5 \\
c_6 \\
c_7 \\
c_8
\end{bmatrix}
$$

In shorthand we can write this equation as

$$[X'Y'] = [M][C],$$

which implies

$$[C] = [M]^{-1} [X'Y'].$$

Having solved for the coefficients $c_1, \ldots, c_8$ we can use them in our original bilinear equations above to obtain the corrected pixel coordinates (x',y') for all pixels (x,y) in the original image within (or near to) the quadrilateral being considered.

To correct for more complex forms of distortion, for example lens distortion, one can use higher order polynomials plus more tie points to generate distortion correction coefficients $c_1, \ldots, c_n$.

Grey Level Interpolation

The problem we have to consider here is that, in general, the distortion correction equations will produce values x' and y' that are not integers. We end up with a set of grey levels for non integer positions in the image. We want to determine what grey levels should be assigned to the integer pixel locations in the output image.

The simplest approach is to assign the grey value for $F(x,y)$ to the pixel having closest integer coordinates to $\hat{F}(x', y')$. The problem with this is that some pixels may be assigned two grey

values, and some may not be assigned a grey level at all - depending on how the integer rounding turns out.

The way to solve this is to look at the problem the other way round. Consider integer pixel locations in the output image and calculate where they must have come from in the input image. That is, work out the inverse image transformation.

The way to solve this is to look at the problem the other way round. Consider integer pixel locations in the output image and calculate where they must have come from in the input image. That is, work out the inverse image transformation.

These locations in the input image will not (in general) have integer coordinates. However, we do know the grey levels of the 4 surrounding integer pixel positions. All we have to do is interpolate across these known intensities to determine the correct grey level of the position when the output pixel came from.

Various interpolation schemes can be used. A common one is bilinear interpolation, given by

$$v(x,y) = c_1 x + c_2 y + c_3 xy + c_4,$$

where $v(x,y)$ is the grey value at position (x,y).

Thus we have four coefficients to solve for. We use the known grey values of the 4 pixels surrounding the `come from' location to solve for the coefficients.

We need to solve the equation

$$
\begin{bmatrix} v_1 \\ v_2 \\ v_3 \\ v_4 \end{bmatrix} = \begin{bmatrix} x_1 & y_1 & x_1y_1 & 1 \\ x_2 & y_2 & x_2y_2 & 1 \\ x_3 & y_3 & x_3y_3 & 1 \\ x_4 & y_4 & x_4y_4 & 1 \end{bmatrix} \begin{bmatrix} c_1 \\ c_2 \\ c_3 \\ c_4 \end{bmatrix}
$$

or, in short,

$$[V] = [M][C],$$

which implies

$$[C] = [M]^{-1}[V].$$

This has to be done for every pixel location in the output image and is thus a lot of computation! Alternatively one could simply use the integer pixel position closest to the `come from location'. This is adequate for most cases.

References

1

Rafael C. Gonzalez and Richard E. Woods. *Digital Image Processing*. Addison-Wesley Publishing Company, 1992, chapter 4.

2

William J. Mitchell. *The Reconfigured Eye: Visual Truth in the Post-Photographic Era*. MIT Press 1992.