UNIVERSITEIT•STELLENBOSCH•UNIVERSITY
jou kennisvennoot • your knowledge partner

COMPUTER VISION

Willie Brink
Applied Mathematics, Stellenbosch University

with contributions made by Ben Herbst, Karin Hunter and McElory Hoffman

July 2013

About these lecture notes

These notes serve as a summary of most of the material covered in the Computer Vision module. Much of these notes (Chapters 5, and 7 to 12) are based on the excellent prescribed textbook *Multiple View Geometry in Computer Vision* by Richard Hartley and Andrew Zisserman (2nd edition, Cambridge University Press, 2003).

Contents

Chapter 1

Introduction

Computer Vision is concerned with the development of mathematical techniques and models for interpreting and understanding visual data. The development of this field was perhaps sparked by the visual perception of the human brain, and ongoing efforts aim to mimic its remarkable capabilities.

Data is usually provided in the form of a digital image, or a sequence of images, and a typical list of tasks for a computer vision system may be:

- to segment an object from a cluttered background;
- to reconstruct certain properties of the object (such as its 3D shape, surface reflectance parameters, colour distribution, etc.) by also considering and modelling the process of image acquisition;
- and to recognize or classify the object.

Modelling the visual world with all of its rich complexities can be extremely difficult. The observed colour of a pixel in the image can be ascribed to many factors such as the viewing angle, the reflectance properties of the material, the presence of unknown light sources, atmospheric conditions, camera effects such as electronic noise and lens distortion, etc., and extracting meaningful information from the image is often challenging. There is also the issue of ambiguity (an image is produced essentially by projecting a three-dimensional scene onto a two-dimensional plane, so that large and distant objects appear similar in size as small objects close to the camera). A further complication is often the sheer size of the data — a single 640×480 colour image, for example, is comprised of about a million single-byte integer values.

Amidst these challenges Computer Vision continues to be an active research field. Cameras and other imaging devices are popular choices for sensors due to their versatility, their passiveness (a scene can be measured without inference), the large amount of data one gets from a single sample and the high sampling rate, and the fact that cameras are relatively cheap and readily available. Moreover,

application areas that benefit from computer vision systems are numerous. Some are listed below.

- **medical applications:** extracting information from X-rays, CAT, MRI, microscopy images, etc.
- **security:** intent recognition for crime prevention; biometrics such as face or fingerprint recognition for access control
- **surveillance:** monitoring activities in sensitive or unmanned areas
- **traffic management:** monitoring and responding to the flow of traffic
- **production industry:** automatic inspection and quality control; measurement; optical sorting of items on a conveyor belt
- **robotics:** obstacle detection, navigation, mapping, path planning, etc.
- **military:** missile guidance systems; battlefield awareness
- **quality of life:** driver assistance; intent recognition
- **photography:** automatic face and smile detection; panoramic stitching; high dynamic range imaging
- **sports:** decision support systems such as the Hawk-Eye in cricket; real-time graphics augmentation in broadcasts
- **web applications:** large scale content-based image searching; SafeSearch; automatic image tagging
- **support for visual effects:** in The Matrix, Avatar, The Hobbit, etc.
- **gaming:** Nintendo Wii; Sony EyeToy; Xbox Kinect

This module will not attempt to cover the entire scope of Computer Vision. We will rather focus on what is often called "two-view geometry", ultimately to enable the reconstruction of 3D coordinates from two spatially separate views of the same scene or object (much like the brain's ability to estimate depth from data captured by the two eyes).

(a) a stereo pair of images (b) a 3D reconstruction

Figure 1.1: An example of 3D reconstruction from two spatially separate views of the same scene. The reconstruction is shown here from two different viewpoints.

We start off in Chapter 2 with a quick and basic overview of some fundamentals of digital image processing. We then describe in Chapter 3 the concept of image features, and how corresponding points in different images of the same scene can be identified.

Then in Chapter 4 we delve into some theoretical aspects of the famous singular value decomposition (SVD) of a matrix. This tool will become instrumental in solving a wide variety of problems encountered in later chapters.

Continuing with theoretical development, we introduce in Chapter 5 the projective plane $\mathbb{P}^2$. Using what we refer to as homogeneous representations of points and lines enables an elegant formulation of some previously clumsy concepts, such as the intersection of parallel lines. We build a hierarchy of linear transformations, starting with the well-known Euclidean transformation (rotation and translation) and ending with the most general projective transformation. We will use a special case of the latter, called plane-to-plane homographies, to remove perspective distortion from a planar object in an image.

In Chapter 6 we deviate slightly from the theory of projective geometry and discuss a remarkably useful technique called RANSAC for estimating model parameters from data potentially corrupted by outliers. This technique is used often in many facets of Computer Vision.

We derive the basic pinhole camera model in Chapter 7, and this model will be used throughout the rest of the notes. We will find that by using homogeneous coordinates we are able to express the projective behaviour of cameras as a simple linear operation. In Chapter 8 we discuss a technique for calibrating a real camera to find all of its parameters.

In Chapter 9 we commence our study of two views, and specifically how two cameras are related geometrically. We then consider in Chapter 10 ways of calibrating a stereo rig (a setup consisting of two cameras) from image feature correspondences. Chapter 11 deals with image rectification — a useful transformation that we can apply to a stereo image pair once we have calibration information. Finally, in Chapter 12, we look at a simple way of performing triangulation in order to determine the 3D locations of features seen by two cameras.

Chapter 2

Fundamentals of digital image processing

In some way or another, images typically form the input of a computer vision system. This chapter briefly outlines some basics of digital image representation and processing.

2.1 Image representation

A digital image consists of a finite set of values called picture elements or *pixels* for short. These pixels are arranged in a regular grid (or raster) of rows and columns, and so it can be useful to think of an image as a matrix.

Every pixel in a *greyscale* image (also called an intensity image) is an 8-bit unsigned integer, meaning that it can have an integer value between 0 and 255. A value of 0 corresponds to pitch black, a value of 255 to pure white, and values between these extremes produce various grey levels between black and white. Figure 2.1 gives an example.

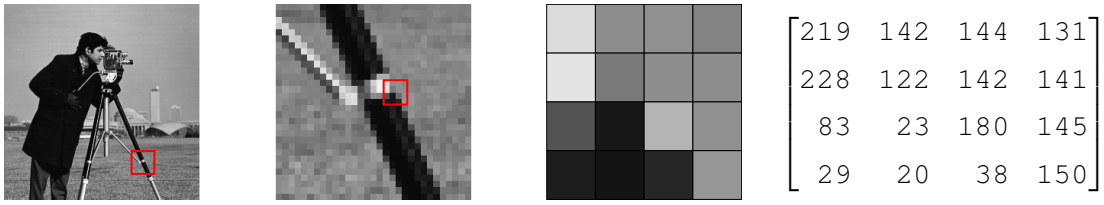


Figure 2.1: The greyscale image on the left consists of a regular grid of pixels, as is evident in the close-ups shown, and each pixel is represented by an integer value between 0 (indicating black) and 255 (indicating white).

A *colour* image is also stored as a raster of pixels. Every pixel is now represented by three integer values between 0 and 255: one for red, one for green and one for blue. These three primary intensities are added to reproduce a certain colour on the screen, and this commonly used way of representing colour is called the RGB colour scheme. Other schemes include HSV, YMCK and YCbCr. Some example colours and their 3-element representations in the RGB scheme are given below.

R: 000 G: 000 B: 000	R: 036 G: 036 B: 036	R: 073 G: 073 B: 073	R: 109 G: 109 B: 109	R: 146 G: 146 B: 146	R: 182 G: 182 B: 182	R: 219 G: 219 B: 219	R: 255 G: 255 B: 255
R: 255 G: 000 B: 000	R: 255 G: 192 B: 000	R: 255 G: 255 B: 000	R: 146 G: 208 B: 080	R: 000 G: 176 B: 080	R: 000 G: 176 B: 240	R: 000 G: 112 B: 192	R: 112 G: 048 B: 160

A colour image is converted to a greyscale image by somehow averaging the three colour channels to arrive at a single intensity value for every pixel. A standard and widely used conversion formula calculates a *weighted* average by adding roughly 30% of the red value, 59% of the green value and 11% of the blue value. We humans are more sensitive to green, so the green channel is weighted most heavily. Figure 2.2 illustrates the effectiveness of this approach.

(a) colour image (b) simple average (c) weighted average

Figure 2.2: In order to convert the colour image in (a) to greyscale, we may simply take the average of the red, green and blue channels, the result of which is shown in (b). Alternatively, a weighted average that places more emphasis on the green channel yields the more pleasing image in (c).

2.2 Pixel transformations

Next we consider some simple operations that can be performed to alter or improve the appearance of an image. Here we focus specifically on greyscale im-

ages, but most of these operations can be utilized on colour images by simply handling every channel separately.

Let $A(r, c)$ denote the pixel value of a greyscale image A at row r and column c. We shall use the convention that rows are numbered from 1 to m and columns from 1 to n (by an alternative convention rows would be numbered from 0 to $m-1$ and columns from 0 to $n-1$).

An important issue worth remembering is that an image is typically stored as a two-dimensional array of unsigned 8-bit integers. When one intends to apply transformations on these pixel values it is a good idea to cast the image data to floating point numbers, perform the required alterations, and then cast the result back to unsigned integers for display or storage purposes. This trick is useful for avoiding unwanted rounding and truncation of the pixel values.

The transformations we consider in this section are all of the form

$$B(r, c) = T(A(r, c)), \tag{2.1}$$

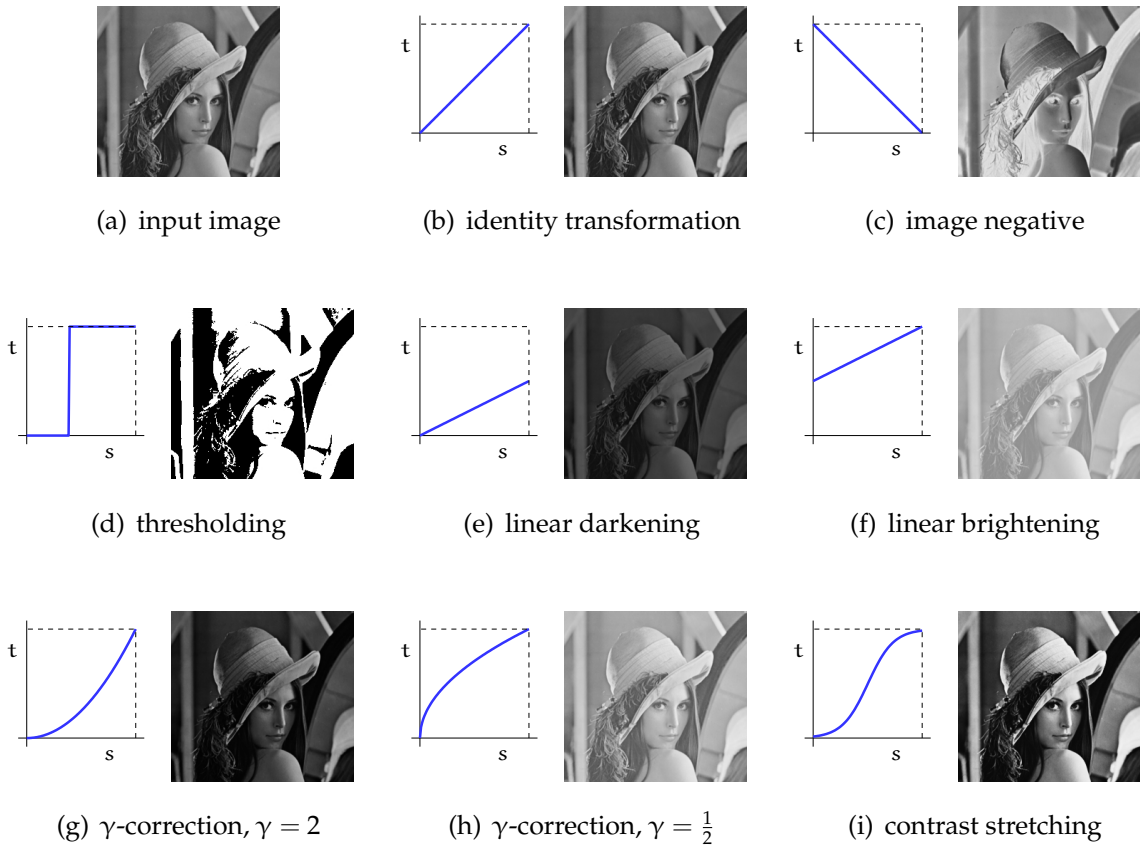
where A is the input image, B is the output image and $T : [0, 255] \mapsto [0, 255]$ is some transformation function (or mapping). Note that the output intensity at pixel coordinates (r, c) is dependent only on the input intensity at those same coordinates, and we may therefore write the transformation function as

$$t = T(s), \tag{2.2}$$

where $s \in [0, 255]$ is the input intensity and $t \in [0, 255]$ is the corresponding output intensity. Since the input intensities can only be integers between 0 and 255 this sort of transformation can be executed efficiently by means of a lookup table.

Some transformation functions, and their effect on an input image, are given in Figure 2.3. We note that

- the identity transformation does not change the image ($t = s$);
- the image negative swops light and dark ($t = 255 - s$);
- thresholding takes all values below the threshold to 0 and all others to 255 (for this example a threshold of 100 was chosen);
- darkening squeezes all values closer to 0 (the slope of that line is $\frac{1}{2}$);
- brightening squeezes the values towards 255 (the previous function is simply shifted upwards by a constant so that 255 now maps to 255);
- gamma correction with $\gamma > 1$ darkens the darker parts of the image (gamma correction is essentially $t = cs^{\gamma}$ where the constant c ensures that 255 maps to 255);
- gamma correction with $\gamma < 1$ brightens the brighter parts;
- the hyperbolic tangent function shown in (i) performs contrast stretching (it is a transformation of the form $t = c_1 \tanh(c_2 s - c_3) + c_4$ where the constants are used to stretch and shift the function appropriately).



(a) input image (b) identity transformation (c) image negative

(d) thresholding (e) linear darkening (f) linear brightening

(g) γ-correction, $\gamma = 2$ (h) γ-correction, $\gamma = \frac{1}{2}$ (i) contrast stretching

Figure 2.3: Illustrating the effects of some pixel transformation functions on the input image in (a).

2.3 Histogram processing

The *histogram* of an $m \times n$ greyscale image is a discrete function

$$h(i) = n_i, \tag{2.3}$$

where $i \in [0, \ldots, 255]$ is the ith greyscale intensity and n_i is the number of pixels in the image with intensity i.

The *normalized* histogram is defined as

$$p(i) = \frac{h(i)}{mn} \tag{2.4}$$

and can be viewed as the probability of intensity i occurring in the image. What we mean by this is that the probability of a randomly selected pixel (such that each pixel in the image is equally likely to be selected) having intensity i is given by $p(i)$. We also note that

$$p(i) \geqslant 0 \text{ for all } i = 0, \ldots, 255, \quad \text{and} \quad \sum_{i=0}^{255} p(i) = 1. \tag{2.5}$$

2.3.1 Histogram equalization

The aim of histogram equalization is to construct a pixel transformation function for a given image such that the histogram of the output is approximately uniform. The intensities in the output image will be more evenly spread, often resulting in a contrast enhancement as we shall soon see.

For now let us assume continuous intensity values in the interval $[0, 255]$, and suppose the transformation function T maps intensity s in the input to t in the output, i.e.

$$t = T(s). \tag{2.6}$$

Let $p_s(s)$ be the probability density function (PDF, the continuous analogue of the normalized histogram) associated with the input image, and $p_t(t)$ the PDF of the output. We wish to choose $T(s)$ in such a way that $p_t(t)$ will be uniform.

From probability theory we know that

$$p_t(t) = p_s(s)\frac{ds}{dt}, \tag{2.7}$$

since $\int p_t(t)dt = \int p_s(s)ds = 1$. We want $p_t(t)$ to be uniform, hence $p_t(t) = \frac{1}{255}$, so that

$$\frac{dt}{ds} = 255p_s(s) \quad \Longrightarrow \quad t = 255\int_0^s p_s(u)du. \tag{2.8}$$

Therefore, transforming the input image with

$$T(s) = 255\int_0^s p_s(u)du \tag{2.9}$$

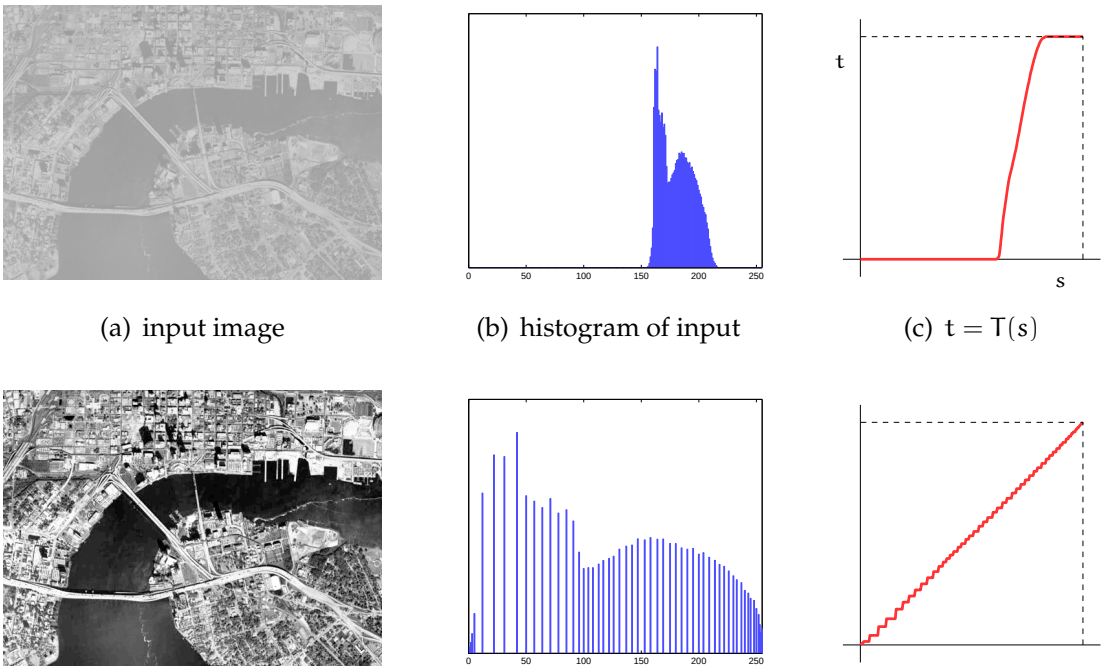
will produce an image with a uniform histogram. Note that the integral part of (2.9) is the cumulative distribution function (CDF) of s.

However, we are not working with continuous intensities. The discrete approximation of the above leads to the following algorithm:

1. let $p(s) = \dfrac{n_s}{mn}$ (the normalized histogram of the input image);
2. define $T(s) = 255\displaystyle\sum_{j=0}^{s} p(j)$, $s = 0, 1, \ldots, 255$;
3. apply this transformation $t = T(s)$ to the input image.

Figure 2.4 gives an example. The input image is low in contrast, which is also clear from the histogram: the intensities of all pixels are in a narrow band between about 150 and 220. Notice that the transformation function shown, as calculated by the procedure above, will have the effect of stretching this band to the entire 0-to-255 interval. Indeed, the output image has much higher contrast.

Incidently, why is the histogram in (e) not perfectly uniform?



(a) input image (b) histogram of input (c) $t = T(s)$

(d) output image (e) histogram of output (f) CDF of (e)

Figure 2.4: Histogram equalization: the histogram of the input image is used to generate a transformation function, shown in (c), that transforms the image into one that has an approximately uniform histogram.

2.3.2 Histogram matching

The aim of histogram matching is to construct an intensity transformation function for a given image, such that the histogram of the output has a pre-specified shape.

Again, let us assume for the moment that intensities are continuous values in $[0, 255]$. We are given the normalized histogram of the input $p_s(s)$, as well as a desired PDF $p_z(z)$ that the output should have. Out task is to find a transformation from s to z. We can do this by first equalizing both $p_s(s)$ and $p_z(z)$.

Let $p_t(t)$ be a uniform PDF. Suppose T is the transformation function that equalizes $p_s(s)$, i.e.

$$t = T(s) = 255 \int_0^s p_s(u)du, \tag{2.10}$$

and G is the transformation function that equalizes $p_z(z)$, i.e.

$$t = G(z) = 255 \int_0^z p_z(v)dv. \tag{2.11}$$

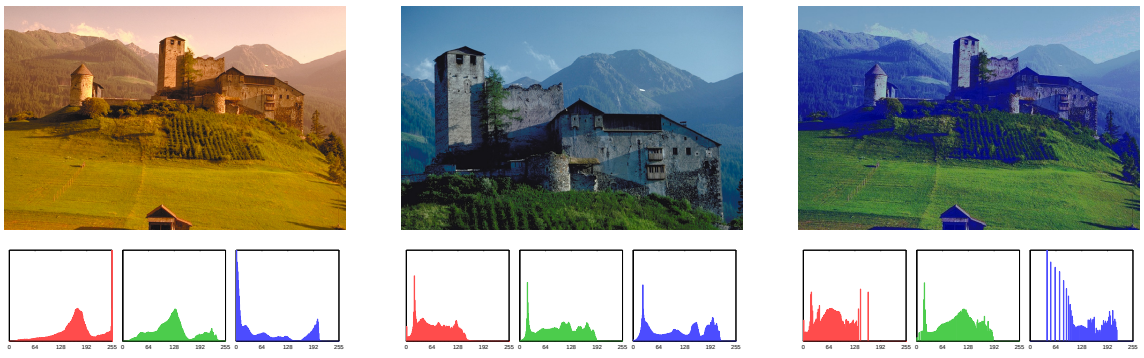
It follows directly from (2.11) that $z = G^{-1}(t)$ and, since $t = T(s)$, we have

$$z = G^{-1}[T(s)]. \tag{2.12}$$

For the discrete case the procedure of histogram matching can be implemented in the following way:

1. compute the normalized histogram $p_s(s)$ of the input image;
2. equalize $p_s(s)$, so let $t = T(s) = 255 \sum_{j=0}^{s} p_s(j)$, $s = 0, \ldots, 255$;
3. equalize the given histogram $p_z(z)$ by calculating $G(z) = 255 \sum_{j=0}^{z} p_z(j)$;
4. for intensity s in the input, find $z^* = \arg\min_z |G(z) - T(s)|$ and map s to z^*.

In Figure 2.5 the first image is transformed so that its histogram approximately matches that of the second image. The result was produced by performing the steps above to each of the three colour channels separately.



(a) first image (b) second image (c) histogram matched

Figure 2.5: Histogram matching: every colour channel of the first image is transformed so that its histogram approximately matches that of the corresponding channel of the second image.

2.4 Spatial filtering

In the previous section we looked at image transformations where every pixel is changed independently according to its own intensity. We now consider spatial image filters where a pixel's new value depends on its old value as well as the intensities of pixels in a small neighbourhood around that pixel. For every such neighbourhood, a pixel in the output image is created with coordinates equal to the coordinates of the centre of the neighbourhood and the value of that new pixel is the result of the filtering operation.

A *linear* spatial filter, meaning that the filter operation is linear, is usually represented by a mask (also known as a kernel, template or window).

2.4.1 Averaging filters

Masks of the 3×3 and 5×5 averaging filters are given by, respectively,

$$\frac{1}{9} \times \begin{array}{|c|c|c|}\hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline \end{array} \quad \text{and} \quad \frac{1}{25} \times \begin{array}{|c|c|c|c|c|}\hline 1 & 1 & 1 & 1 & 1 \\ \hline 1 & 1 & 1 & 1 & 1 \\ \hline 1 & 1 & 1 & 1 & 1 \\ \hline 1 & 1 & 1 & 1 & 1 \\ \hline 1 & 1 & 1 & 1 & 1 \\ \hline \end{array} . \tag{2.13}$$

We apply such a mask to an $m \times n$ greyscale image A to obtain a new image B, by calculating

$$B(r,c) = \sum_{s=-h}^{h} \sum_{t=-h}^{h} M(s+h+1, t+h+1)\, A(r+s, c+t), \tag{2.14}$$

for $r = 1, 2, \ldots, m$ and $c = 1, 2, \ldots, n$. Here h is the so-called half-size of the mask. For the 3×3 mask in (2.13) we have $h = 1$ and for the 5×5 mask $h = 2$. In general, a mask with half-size h is some $(2h+1) \times (2h+1)$ matrix.

The expression in (2.14) amounts to a discrete correlation between the mask M and the image A. It basically centres the mask at pixel coordinates (r, c) on image A, multiplies each mask entry with the corresponding pixel in A, adds the whole lot together, and assigns that value to $B(r, c)$. For masks in (2.13), the value of $B(r, c)$ is thus the average of the pixel $A(r, c)$ and its neighbours, which is why we call them averaging filters.

When moving a filter mask over an image, one has to decide what to do at the boundary pixels where certain neighbours are not available. Some options include padding the image with zeros, duplicating the boundary pixels, or mirroring the image about the boundaries.

Figure 2.6 shows results from applying the averaging filter with increasing mask size. A blurring (or smoothing) effect is apparent.

(a) original (b) 5×5 averaging (c) 9×9 averaging (d) 15×15 averaging

Figure 2.6: Illustrating the effect of filtering an image with an averaging mask of increasing size. The image resolution is 230×230, and boundaries were padded with zeros.

2.4.2 Gaussian blurring

Gaussian blurring is achieved by filtering an image with a Gaussian mask. For some chosen half-size h and standard deviation σ, we calculate

$$\widetilde{g}(i,j) = \exp\left[-\frac{(i-h-1)^2+(j-h-1)^2}{2\sigma^2}\right], \tag{2.15}$$

for $i = 1, 2, \ldots, 2h+1$ and $j = 1, 2, \ldots, 2h+1$. We normalize, so that the coefficients will sum to one, to arrive at our $(2h+1) \times (2h+1)$ Gaussian mask:

$$g(i,j) = \frac{\widetilde{g}(i,j)}{w}, \quad \text{where } w = \sum_{i=1}^{2h+1}\sum_{j=1}^{2h+1} \widetilde{g}(i,j). \tag{2.16}$$

Increasing the half-size h enlarges the mask, so that a pixel's new value will depend on the intensities of a larger neighbourhood around that pixel. The standard deviation σ changes the shape of the Gaussian, in the sense that a smaller σ will place less emphasis on faraway pixels.

2.4.3 The median filter

The median filter is a nonlinear smoothing filter. It operates by ranking the pixels in a neighbourhood centred around $A(r,c)$ according to their intensities, and then taking the median of these values as the new value $B(r,c)$. This filter can be highly successful in removing impulse noise (spurious pixels that have been corrupted to either pitch black or pure white, also appropriately named "salt-and-pepper" noise). An example is given in Figure 2.7.

(a) original (b) corrupted (10%) (c) 3×3 median (d) 5×5 median

Figure 2.7: An image is corrupted by salt-and-pepper noise (here 10% of pixels are changed to either 0 or 255), and a median filter of varying sizes attempts to restore it.

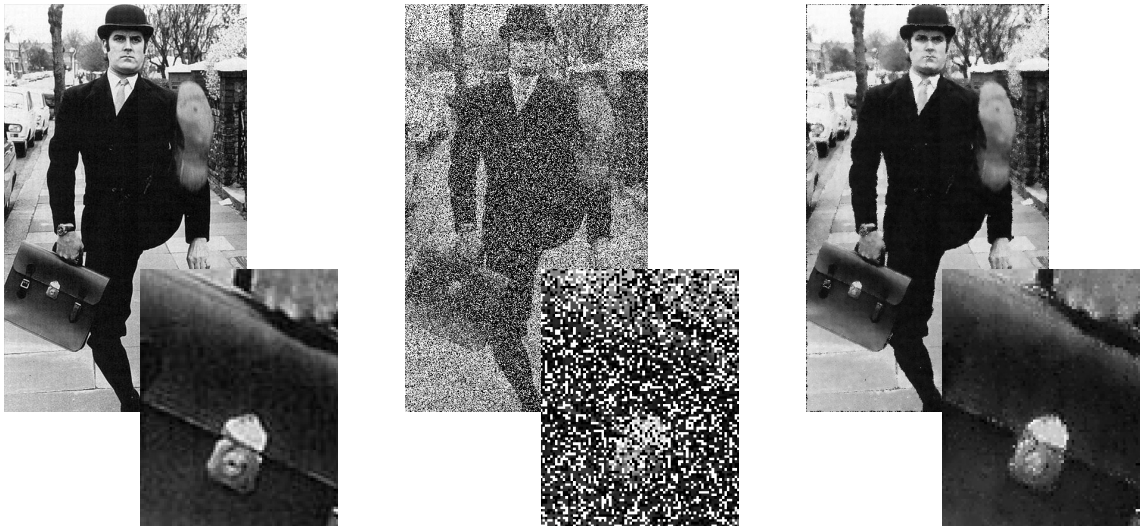
A problem with the standard median filter is that it may also smooth parts of the image that are not corrupted by noise. The *adaptive* median filter attempts to overcome this problem by changing (or adapting) its behaviour based on statistical characteristics of the observed image data in the region of the filter. The algorithm is slightly more complex but usually produces vastly superior results.

Suppose S_{rc} is some local region around pixel (r, c), with intensity $A(r, c)$, in the image A. Furthermore, let z_{min}, z_{max} and z_{med} be the minimum, maximum and median intensity in S_{rc}. The adaptive median filter algorithm is given below. This algorithm is performed for every pixel (r, c) in A, to arrive at a value for pixel (r, c) in the output image B.

while (true)
    if ($z_{med} = z_{min}$) or ($z_{med} = z_{max}$) **then**
        increase window size
        if window size $>$ maximum allowed window size **then**
            return z_{med}
        endif
        re-calculate z_{min}, z_{max} and z_{med}
    else
        break from while-loop
    endif
endwhile
if ($A(r, c) > z_{min}$) and ($A(r, c) < z_{max}$) **then**
    return $A(r, c)$
else
    return z_{med}
endif

In this algorithm we consider z_{min} and z_{max} to be "impulse-like" noise. The purpose of the while-loop is to determine if the current z_{med} is an impulse. If it is, we repeatedly increase the window size around the current pixel until z_{med} is not an impulse or the maximum window size is reached. The purpose of the if-statement after the while-loop is to determine whether or not the intensity $A(r, c)$ of the current pixel is in fact an impulse. If it is not, we return that intensity unchanged. This results in reduced blurring (compared to the standard median filter) because we only change a pixel if we believe it to be an impulse.

We demonstrate the remarkable ability of this adaptive median filter, by corrupting the image shown in Figure 2.8(a) with severe salt-and-pepper noise. This corrupted image, in which about half of the original pixels have been changed to either 0 or 255, is shown in (b). The result of applying the adaptive median filter to this image is given in (c). We observe that the filter is able to remove most of the impulse noise without excessively blurring the image.

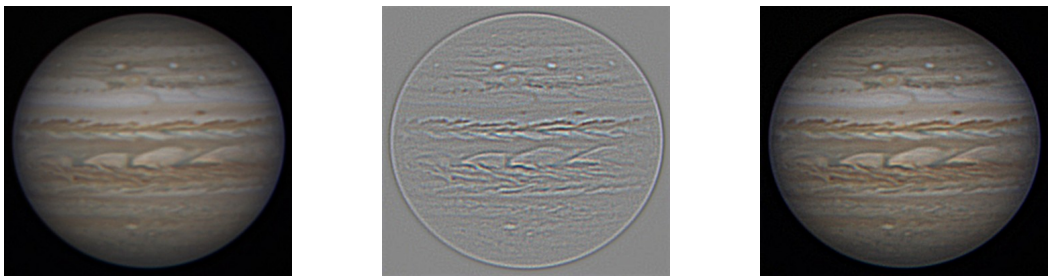


(a) original (b) corrupted (50%) (c) restored with AMF

Figure 2.8: An image is severely corrupted by salt-and-pepper noise (here 50% of pixels are changed to either 0 or 255). The adaptive median filter (AMF) is remarkably successful in restoring this image. Close-ups are also shown.

2.4.4 Image sharpening

The averaging or Gaussian filter smooths an image. We can also apply a filter for the opposite effect, that is to sharpen the image. One technique for this is called *unsharp masking*. The idea is quite intuitive. We smooth the given image (for example with an averaging filter), subtract the smoothed image from the original to get our mask, and then add this mask to the original image. This effectively produces the inverse of smoothing, and the effect can be boosted even more by adding some multiple of the mask to the original (which leads to a technique called "high-boost filtering"). An example is given in Figure 2.9.



(a) original image (b) unsharp mask (c) sharpened image

Figure 2.9: We perform unsharp masking on an image by calculating the difference between the image and a smoothed version of it, and then adding that mask to the original.

2.5 Edge detection

Next we turn our attention to some basic techniques for detecting edges in images. Edges are distinguishable as significant changes in image intensity. If we view the image data as a function in two variables (intensity $= A(r, c)$), we wish to identify points where this function changes quickly or, put differently, where the function's gradient is large in magnitude.

For this reason the *image gradient* is the tool of choice for describing edge strength at every pixel. In addition to edge strength, the gradient provides edge direction as well.

The gradient of a function f in two variables is

$$\nabla f = \begin{bmatrix} g_x \\ g_y \end{bmatrix} = \begin{bmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{bmatrix}. \tag{2.17}$$

This vector points in the direction of greatest change and its length indicates the magnitude of that change. More precisely,

$$M(x, y) = \|\nabla f\| = \sqrt{g_x^2 + g_y^2} \quad \text{and} \quad \alpha(x, y) = \arctan(g_y / g_x) \tag{2.18}$$

give the magnitude and direction of ∇f respectively.

Of course, an image is not a continuous function. Various discrete operators, applied in precisely the same manner as any other linear filter mask, have been proposed as a discrete analogue to the gradient. They all originate from some form of a discrete approximation of the derivative operator, and a commonly used one is the set of Sobel operators:

$$\begin{array}{|c|c|c|} \hline -1 & -2 & -1 \\ \hline 0 & 0 & 0 \\ \hline 1 & 2 & 1 \\ \hline \end{array} \quad \text{and} \quad \begin{array}{|c|c|c|} \hline -1 & 0 & 1 \\ \hline -2 & 0 & 2 \\ \hline -1 & 0 & 1 \\ \hline \end{array}. \tag{2.19}$$

The first of these computes change in the vertical direction (across image rows), so that horizontal edges will be illuminated, and the second is used to compute change in the horizontal direction (across image columns) for illuminating vertical edges. By applying these two masks separately we obtain two edge images, say G_x and G_y, from which magnitude and direction can be determined.

The edge magnitude image M, calculated as

$$M(r, c) = \sqrt{\left[G_x(r, c)\right]^2 + \left[G_y(r, c)\right]^2}, \tag{2.20}$$

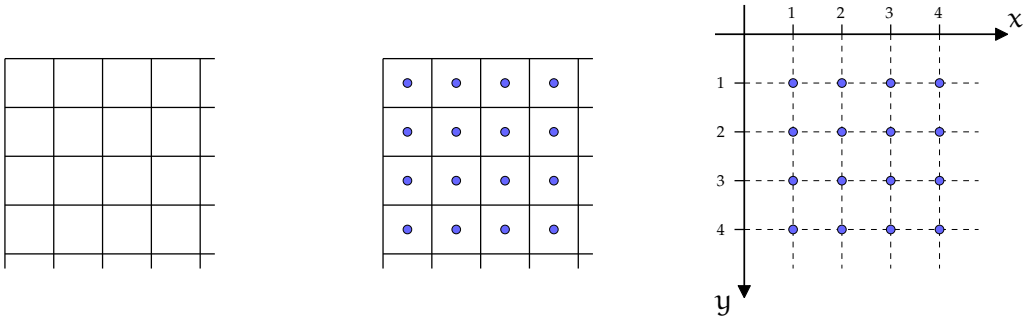
provides an indication of edge strength at every pixel in the original image. If required, these values may be thresholded to produce a binary edge image.

2.6 Image coordinates

Up to now we have described an image as an $m \times n$ matrix, where every entry gives the intensity or colour of the image at a particular pixel. We have also hinted at the fact that an image can be thought of as a function in two discrete variables: the row index and column index.

We will soon start to think of the image as a truncated 2D plane in 3D space, and pixels in the image will be associated with points on this plane. We will sometimes be forced to generalize the integer coordinates of these points to real numbers, and even move beyond the interval $[1, m] \times [1, n]$. In order to accommodate these more general image coordinates, a system of axes can be placed on the image in a simple way.

Figure 2.10 demonstrates this idea, where pixels correspond to integer coordinates in an (x, y) system of axes. Notice that we have chosen the x-axis to run horizontally across the image from left to right, and the y-axis vertically from top to bottom. We could have swopped these axes but our choice eases, for example, image annotation (plotting points or lines on top of an image, as we will soon be doing). In fact, many image processing toolboxes (e.g. in Matlab or Python) follow this convention. It is also important to take note of our choice of origin. In these notes we index pixel rows from 1 to m, and columns from 1 to n, so the top-left pixel has coordinates $(1, 1)$ as opposed to $(0, 0)$. If indices run from 0 to $m - 1$ and 0 to $n - 1$, the top-left pixel will coincide with the origin.



(a) top-left part of an image (b) pixels as discrete points (c) a system of image axes

Figure 2.10: If we think of pixel locations as discrete points in the 2D plane, we can define a system of axes in which the pixels have integer coordinates.

With this system of axes in place, we may now specify some non-integer coordinate pair and it will be clear where this point is located in the image. For example, the centre (x_0, y_0) of an image with m rows and n columns is calculated as

$$x_0 = \tfrac{1}{2}(n + 1), \quad y_0 = \tfrac{1}{2}(m + 1). \tag{2.21}$$

If m or n (or both) is even this point will not coincide precisely with a pixel.

When working with image coordinates, it is crucially important to be aware of the distinction between the matrix $(row, column)$ representation of a pixel and its (x, y) representation. It is clear from Figure 2.10 that the row number corresponds to the y-coordinate, and the column number to the x-coordinate.

2.7 Image interpolation

We end this chapter on image processing with a discussion on image interpolation. It is an issue that can surface in many different contexts and later on, when we warp or distort images, we will require an ability to interpolate the image data.

Suppose we wish to create a new image using some mapping of the form

$$\underline{p}' = f(\underline{p}), \tag{2.22}$$

where $\underline{p} = \begin{bmatrix} x & y \end{bmatrix}^T$ is the image coordinates of a pixel in the original image, and $\underline{p}' = \begin{bmatrix} x' & y' \end{bmatrix}^T$ the image coordinates of its correspondence in the new image. That is to say, we wish to take every pixel (with its colour) in the original image to some new location. Some examples of such mappings are:

- image scaling: $\underline{p}' = s\,\underline{p}$ (if $s < 1$ we downsample, if $s > 1$ we upsample);
- translation: $\underline{p}' = \underline{p} + \underline{t}$ with $\underline{t} = \begin{bmatrix} t_x & t_y \end{bmatrix}^T$;
- rotation about origin: $\underline{p}' = R\,\underline{p}$ with $R = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}$;
- rotation about point $\underline{c}$: $\underline{p}' = R(\underline{p} - \underline{c}) + \underline{c}$.

The problem with implementing a mapping such as one of these directly, i.e. looping through every pixel $\underline{p}$ in the original image, mapping it to new coordinates $\underline{p}'$ and assigning its value to that new pixel, is that we may end up with pixels in the new image that have no values. Imagine, for example, upsampling an image by a factor of 2. The coordinates of every pixel in the original (small) image are multiplied by 2, and the coordinates thus obtained are given the colours of the original image. But what about the pixels in this new (larger) image in-between the multiples of 2?

Therefore, we rather loop through the *new* image (which we have initialized appropriately) and, for every pixel $\underline{p}'$ in this image, we use the inverse mapping

$$\underline{p} = f^{-1}(\underline{p}'), \tag{2.23}$$

to arrive at pixel coordinates $\underline{p}$ in the original image, and then assign the colour at $\underline{p}$ in the original image to the pixel $\underline{p}'$ in the new image. We thereby populate every single pixel in the new image, and avoid the problem of unassigned pixels mentioned above.

There is still one slight complication. When we calculate $\underline{p} = f^{-1}(\underline{p}')$ we may very well get non-integer coordinates. We must decide what colour to assign to pixel $\underline{p}'$, given its corresponding, possibly non-integer coordinates $\underline{p}$ in the original image. Since we only have image data at integer coordinates, *interpolation* becomes necessary.

2.7.1 Nearest neighbour interpolation

The simplest form of image interpolation is called nearest neighbour interpolation. Once $\underline{p}$ is computed, using the inverse mapping in (2.23), we simply round the two coordinates to the nearest integers. If the resulting coordinates fall within the boundaries of the original image, we assign the colour at those coordinates to pixel $\underline{p}'$ of the new image.

Figure 2.11(b) shows the result of applying nearest neighbour interpolation in order to upsample and rotate an image. (What would the mapping f be in this case?) The effect of this scheme often appears a bit "pixelated".

2.7.2 Bilinear interpolation

Bilinear interpolation is an improvement upon nearest neighbour interpolation, in the sense that the result is typically "smoother", at the expense of some additional computation.

Suppose the original image is A, and the new image that we want to create is B. For every pixel $\underline{p}' = \begin{bmatrix} x' & y' \end{bmatrix}^T$ in B we apply (2.23) to get (the possibly non-integer) coordinates $\underline{p} = \begin{bmatrix} x & y \end{bmatrix}^T$. With

$$x_f = \lfloor x \rfloor, \quad x_c = \lceil x \rceil, \quad y_f = \lfloor y \rfloor, \quad y_c = \lceil y \rceil, \tag{2.24}$$

we can combine the colours of the four pixels (x_f, y_f), (x_f, y_c), (x_c, y_f) and (x_c, y_c) in A surrounding (x, y) with the following weighted average, to obtain a colour for the pixel (x', y') in B:

$$\begin{aligned} B(x', y') &= (x_c - x)\Big[(y_c - y)A(x_f, y_f) + (y - y_f)A(x_f, y_c)\Big] \\ &\quad + (x - x_f)\Big[(y_c - y)A(x_c, y_f) + (y - y_f)A(x_c, y_c)\Big]. \end{aligned} \tag{2.25}$$

Note that pixels closer to (x, y) are weighted more heavily. Since this scheme is essentially linear interpolation in both the x- and y-directions, we call it *bilinear* interpolation.

The result of applying bilinear interpolation, rather than nearest neighbour, is shown in Figure 2.11(c). An improvement is quite clear.

(a) original image

(b) nearest neighbour interpolation

(c) bilinear interpolation

Figure 2.11: The task here was to upsample the original image by a factor of 2, and then rotate clockwise through 15°. Results from using nearest neighbour interpolation and bilinear interpolation are shown, with close-ups.

2.7.3 Bicubic interpolation

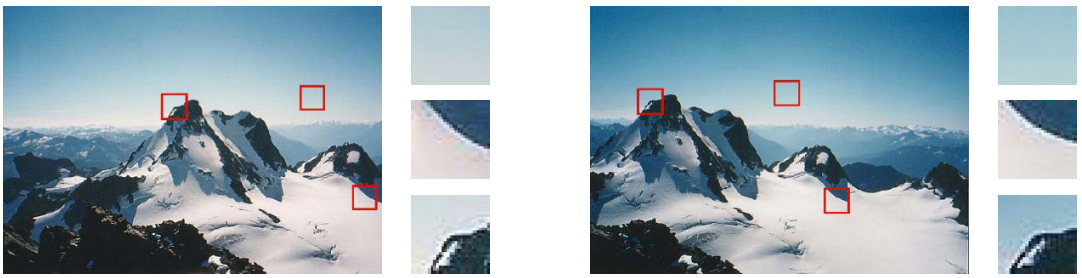
We can take the idea of interpolation one step further, by combining colours in a larger neighbourhood around (x, y). In the previous scheme we used 4 neighbours (in a 2×2 configuration). If we rather use 16 neighbours (in a 4×4 configuration), and apply cubic interpolation in the two directions, we end up with what is called *bicubic* interpolation. This scheme requires even more calculations, but the results are also typically better.

Chapter 3

Feature detection and matching

Many computer vision applications operate on more than one image of the same object or scene. It is often imperative that points corresponding to the same 3D features are matched across those images, in order to relate the images to one another. In this chapter we discuss the general procedure of feature detection and matching, and provide brief details of some common techniques.

Since the aim here is to identify feature points that are distinct and easily matchable across different views (that may also differ in scale and rotation) it makes sense to pick features that exhibit corner-like qualities. Consider for example the two images[1] in Figure 3.1, where local regions around some feature candidates are also shown. The first of these candidates (in the sky) would be a terrible choice for a feature as reliable matching would be near impossible. The second is better, and the third even better, because these regions appear to be much more distinct and therefore more reliable for matching.



(a) image A (b) image B

Figure 3.1: Two images of a scene from different viewpoints. Local regions around some feature candidates are also shown. The top one (in the sky) is a bad choice as it would be near impossible to match reliably. The feature on the bottom is far better, as it appears to be much more distinct and therefore more reliable for matching.

[1] R. Szeliski, *Computer Vision*, Springer-Verlag, 2011.

3.1 Overview

The process of feature detection and matching is usually split into three tasks:

1. *Detection:* Interesting (salient, easily matchable) points are identified in each image.
2. *Description:* The local appearance around each feature point is described in some way that is (ideally) invariant under changes in illumination, translation, scale and in-plane rotation. We typically end up with a descriptor vector for each feature point.
3. *Matching:* Descriptors are compared across the images, to identify similar features. For two images we may get a set of pairs $(x_i, y_i) \leftrightarrow (x'_i, y'_i)$, where (x_i, y_i) is a feature in one image and (x'_i, y'_i) its matching feature in the other image.

As mentioned, feature detection and matching is an essential component of many computer vision applications. Examples include:

- image alignment (e.g. for panoramic stitching);
- video stabilization;
- 3D model reconstruction from two or more views;
- camera motion estimation (e.g. in robotic navigation or augmented reality);
- object tracking;
- object recognition.

The success of these procedures often hinges precariously on the accuracy of the feature matches. It is no wonder that great effort has gone into the development and refinement of many different techniques for detecting and matching image features. Some of these are listed below:

- Lucas-Kanade [Lucas & Kanade 1981]
- Harris [Harris & Stephens 1988]
- Shi-Tomasi [Shi & Tomasi 1994]
- SUSAN (smallest univalue segment assimilating nucleus) [Smith & Brady 1997]
- MSER (maximally stable extremal regions) [Matas et al. 2002]
- SIFT (scale invariant feature transform) [Lowe 2004]
- HOG (histogram of oriented gradients) [Dalal & Triggs 2005]
- FAST (features from accelerated segment test) [Rosten & Drummond 2005]
- SURF (speeded-up robust features) [Bay et al. 2008]

We will look at Harris features because the technique is relatively easy to understand and implement, and discuss very briefly SIFT and SURF because they work well and are among the most commonly used.

3.2 Harris corner detection

The procedure given below is for detecting corner points (i.e. possibly suitable candidates for image feature matching) in a single image, and is loosely based on a paper by Harris and Stephens[2].

1. Suppose A is an 8-bit greyscale image, such that $A(x, y) \in [0, 255]$ gives the intensity of pixel (x, y). We first cast the image to double-precision floating point numbers, then divide by 255 so that every element of A is now a floating point number between 0 and 1.

2. We compute the two components G_x and G_y of the image gradient by filtering A with the Sobel masks (refer to section 2.5).

3. We now blur products of these gradient components with a Gaussian filter (as in section 2.4.2). Suppose $\otimes$ indicates element-wise multiplication, then let

   G_1 be a Gaussian filtered version of $(G_x \otimes G_x)$,
   G_2 a Gaussian filtered version of $(G_y \otimes G_y)$,
   and G_3 a Gaussian filtered version of $((G_x \otimes G_y) \otimes (G_x \otimes G_y))$.

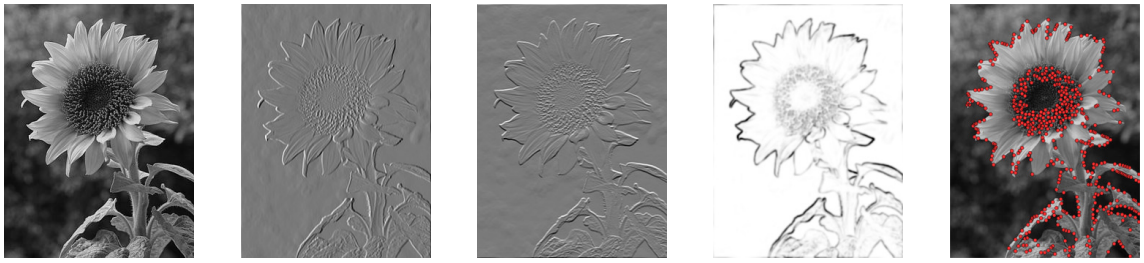
   In generating Gaussian masks for the calculation of G_1, G_2 and G_3 we typically choose $h = 3$ and $\sigma = 1$.

4. A corner response is now calculated for every pixel:
   $$R(x, y) = G_1(x, y) + G_2(x, y) - k\Big[\big(G_1(x, y)\big)\big(G_2(x, y)\big) - \big(G_3(x, y)\big)^2\Big]$$
   where the constant k is fixed empirically at 0.04. These response values give an indication of corner strength at the pixels.

5. Every pixel is visited and if its R-value is maximum in a 3×3 neighbourhood around that pixel we mark it. Finally, the Harris corners of A are the strongest N marked pixels based on their R-values (or, alternatively, all the marked pixels with R-values greater than some threshold).

An example is shown in Figure 3.2. The two components G_x and G_y of the gradient of image A are shown, as well as the corner response image R (for clarity in displaying this image, black indicates a high response). The 500 strongest Harris corner points are shown in (e).

[2] C. Harris & M. Stephens, *A combined corner and edge detector*, Alvey Vision Conference, pp. 147–152, 1988.



(a) image A (b) G_x (c) G_y (d) R (e) corners

Figure 3.2: The 500 strongest Harris corner points, as determined by the procedure described in section 3.2, for the input image in (a).

3.3 Window correlation

The technique given in the previous section returns a list of pixel coordinates which we can think of as features. After we have applied such a feature detector on two images, our next task would be to determine which feature in image A matches with (or corresponds to) which feature in image B.

Window correlation is a simple, albeit not very good, technique for doing just that. Suppose the features in image A have coordinates (x_1, y_1), (x_2, y_2), ..., (x_m, y_m) and those in B have coordinates (s_1, t_1), (s_2, t_2), ..., (s_n, t_n). Window correlation requires two parameters: a maximum disparity d_{max} and a window half-size h, and may be implemented as follows.

for every feature i in A **do**
  $\text{match}(i) = -1$
  $c_{min} = \infty$
  for every feature j in B that is within d_{max} of feature i **do**
    $c = \sqrt{\sum_{u=-h}^{h} \sum_{v=-h}^{h} \left[A(x_i + u, y_i + v) - B(s_j + u, t_j + v)\right]^2}$
    if $c < c_{min}$ **then**
      $c_{min} = c$
      $\text{match}(i) = j$
    endif
  endfor
endfor

Note that this method essentially compares a local region around a feature in A to one around a feature in B. Once completed, feature i in image A matches with feature $j = \text{match}(i)$ in image B as long as $\text{match}(i) \neq -1$. If $\text{match}(i) = -1$ feature i has no matching feature in image B.

This method of matching features is not very good, in the sense that it typically produces many incorrect matches. It is based on an assumption that a local $(2h+1) \times (2h+1)$ window centred around a particular feature remains unchanged from one image to the next (i.e. as the camera moves). This assumption would be approximately true only if the feature and its surrounding pixels are on a plane perpendicular to the camera's principal viewing direction, and the camera moves parallel with this plane without rotating. Any other movement might change the appearance of local patches quite drastically.

Slightly better performance can be achieved by incorporation a *consistency check*. The idea is quite simple: repeat the procedure of window correlation, but now find for every feature in B its closest feature in A. We then keep only those matches which were found consistently, i.e. those (i, j) pairs for which j is the closest feature in B to feature i in A, *and* i is the closest feature in A to feature j in B. This consistency check roughly doubles the execution time, but should remove a number of incorrect matches.

3.4 Visualizing feature matches

The process of feature matching may return a set of correspondences $(x_i, y_i) \leftrightarrow (x'_i, y'_i)$, such that point (x_i, y_i) in image A is believed to match with point (x'_i, y'_i) in image B. It is often useful to visualize these correspondences, in order to get some qualitative idea of the accuracy or correctness.

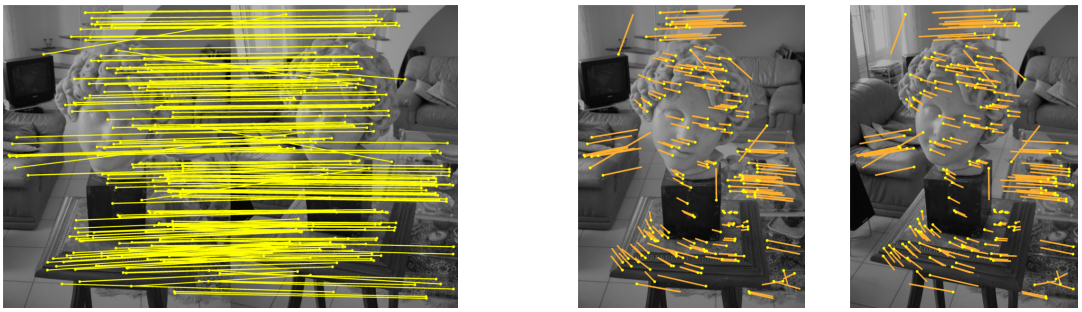
There are various ways in which to visualize feature correspondences. We typically display the image(s), and plot features as dots over them. It might be a good idea to display the images in greyscale, slightly darkened, so that the features will be easily visible. We now have a few options of exactly how we present the feature matches, some of which are listed below.

- Display the two images separately, with their features as dots, and colour matching features similarly. This method may work well if the number of matches is relatively small, but it can be difficult to get an immediate idea of which features in one image correspond with which in the other image.
- Place the two images next to each other, and draw lines from (x_i, y_i) in the one image to (x'_i, y'_i) in the other. This commonly used method may give a general idea of how features shift from one image to the other, but many correspondences can quickly clutter such a plot.
- Show only one image, say image A, and draw line segments from (x_i, y_i) to (x'_i, y'_i). Even though (x'_i, y'_i) does not have much meaning in image A, this method can give a much clearer indication of how features move from A to B, and incorrectly matched points stand out quite clearly.

Figure 3.3 provides examples of these visualization methods.

(a) original images A and B

(b) matches as similarly coloured dots



(c) matches as lines between the images

(d) matches as line segments in one image

Figure 3.3: Different ways of visualizing feature matches between two images. These matches are a subset of those obtained with SIFT.

3.5 SIFT and SURF

The method of window correlation discussed above is not particularly well suited for the task of matching features across images. Its problem lies in the way it describes local regions around detected feature points. In this section we discuss a few alternatives very briefly. These methods are widely used, and various open source implementations are available online.

3.5.1 SIFT

The scale invariant feature transform (or SIFT) is one of the most popular techniques for feature detection and matching. It was devised by David Lowe[3] and is widely considered to be the de facto benchmark for evaluating other methods.

The SIFT matching scheme begins by blurring down- and upsampled versions of the image with Gaussian kernels, to simulate the effect of scale changes. Keypoints are then identified by selecting maxima and minima of differences in such

[3]D. Lowe, *Distinctive image features from scale-invariant keypoints*, IJCV, 60(2):91–110, 2004.

a scale space. The idea is to select points that remain stable in the face of this simulated scaling, so that they will be detectable and matchable across images that vary in scale.

Dominant orientations are then assigned, by looking at the image gradient of a patch surrounding a particular keypoint, and will be aligned in the matching stage to ignore possible in-plane rotation differences. Robustness to local affine distortion (caused for example by changing perspective) is achieved by blurring and resampling local image orientation planes.

A local region around a particular keypoint is normalized with respect to scale, orientation, the small affine distortions mentioned above, and illumination, and a 128-element vector is built that will serve as the feature's descriptor. Feature matching now takes place by searching for minimum Euclidean distance among the descriptors.

Although this technique is a lot more computationally intensive than the simple Harris corner detector and window matching, it is vastly superior because the descriptors are so good at capturing the matchable essence of features.

3.5.2 SURF

The low speed of the SIFT algorithm prompted Bay et al.[4] to suggest in 2008 the speeded up robust features (SURF) detector-descriptor scheme. It approximates a number of the computationally expensive subroutines of SIFT, thereby gaining efficiency at a slight expense of accuracy.

More specifically, SURF improves the speed of SIFT dramatically by approximating the difference-of-Gaussians operator with box filters and the SIFT gradient calculations (for identifying dominant orientation and building descriptors) with Haar wavelets.

As mentioned there are many freely available implementations of both SIFT and SURF, in many different programming languages.

[4] H. Bay, A. Ess, T. Tuytelaars & L. van Gool, *SURF: speeded-up robust features*, Computer Vision and Image Understanding 110(3):346–359, 2008.

Chapter 4

Singular value decomposition

The singular value decomposition (or SVD for short) is one of the basic mathematical tools we use in Computer Vision, and this chapter summarizes some of its important aspects.

The SVD is a special factorization of a matrix A that extracts, in a remarkably clear form, useful information about A. It provides an orthonormal basis for each of the four fundamental spaces (the null space, the column space, the row space and the left null space), is useful for solving homogeneous linear equations (of the form $A\underline{x} = \underline{0}$) both exactly and in a least squares sense, can be used to calculate the pseudo-inverse of a matrix (which we will need later on), and provides a means for low-rank matrix approximation and dimensionality reduction in pattern recognition problems.

4.1 Definition of the SVD

The singular value decomposition (SVD) of an $m \times n$ real matrix A is given by

$$A = U\Sigma V^T, \tag{4.1}$$

where U and V are unitary[1] matrices, of sizes $m \times m$ and $n \times n$ respectively. Σ is a diagonal $m \times n$ matrix containing the *singular values* of A.

For example, if $m = 3$ and $n = 2$ then the SVD has the form

$$\underset{A}{\begin{bmatrix} \cdot & \cdot \\ \cdot & \cdot \\ \cdot & \cdot \end{bmatrix}} = \underset{U}{\begin{bmatrix} \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot \end{bmatrix}} \underset{\Sigma}{\begin{bmatrix} \sigma_1 & 0 \\ 0 & \sigma_2 \\ 0 & 0 \end{bmatrix}} \underset{V^T}{\begin{bmatrix} \cdot & \cdot \\ \cdot & \cdot \end{bmatrix}}.$$

[1] *unitary* (or *orthogonal*) matrices have orthonormal columns and rows, implying that $U^TU = UU^T = I$ for U unitary. All rotation matrices are unitary.

The singular values are (by convention) ordered in non-increasing order of magnitude, so that $\sigma_j \leqslant \sigma_i$ if $j > i$. There are exactly $s = \min(m, n)$ singular values, σ_j, $j = 1, \ldots, s$, some of which may be zero. The number r of nonzero singular values is called the *rank* of A (i.e. $\sigma_j = 0$, $j = r+1, \ldots, s$). If $r = s$ then A is said to be of full rank.

4.2 Computation of the SVD

Expressions for the different factors in the SVD are easily obtained. By multiplying (4.1) from the right and left respectively with A^T, we obtain

$$AA^T U = U\Sigma\Sigma^T \quad \text{and} \quad A^T A V = V\Sigma^T\Sigma. \tag{4.2}$$

Thus the columns of U are the eigenvectors of the symmetric matrix AA^T, and the columns of V are the eigenvectors of the symmetric matrix A^TA. The singular values are the positive square roots of the eigenvalues of AA^T or A^TA. Note that since both AA^T and A^TA are symmetric and positive semi-definite, their eigenvalues are always real and non-negative.

Although the formulas in (4.2) allow us to calculate the SVD by calculating the eigenvalues and eigenvectors of symmetric matrices, there is a loss of information due to rounding errors when AA^T or A^TA is formed. This is easily demonstrated, borrowing an example from the literature[2].

Consider the 2×2 matrix

$$A = \begin{bmatrix} 1 & 1 \\ 0 & \sqrt{\eta} \end{bmatrix}$$

where $\eta = \varepsilon/2$ and ε denotes machine precision, so that $1 + \eta$ rounds to 1. Its singular values are close to $\sqrt{2}$ and $\sqrt{\eta/2}$. Since

$$A^TA = \begin{bmatrix} 1 & 1 \\ 1 & 1+\eta \end{bmatrix}$$

it rounds to

$$\begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix},$$

with singular values given by $\sqrt{2}$ and 0. Thus, rounding $1 + \eta$ to 1 changes the smaller singular value from $\sqrt{\eta/2} = \sqrt{\varepsilon}/2$ to 0. This represents a considerable loss in significant digits in the small singular value. Note that it is the smallest singular values that are affected the most, therefore, for applications such as face recognition (section 4.6) where we ignore small singular values, this usually does not cause any problems.

[2] J. Demmel, *Accurate SVDs of structured matrices*, SIAM Journal on Matrix Analysis and Applications, 21(2):562–580, 1999.

4.3 Reduced form of the SVD

Because of the many zeros in Σ, a number of columns of U or rows in V^T (depending on the shape of A) may be redundant. If for example $m > n$ and A is of full rank, the areas inside the dashed lines in Figure 4.1(a) can always be removed without any loss of information. If $r < s$, i.e. A is not of full rank, even more rows and columns can be removed, as illustrated in Figure 4.1(b). This gives the *reduced form* of the SVD.

We can formally state this as

$$A = U_r \Sigma_r V_r^T, \tag{4.3}$$

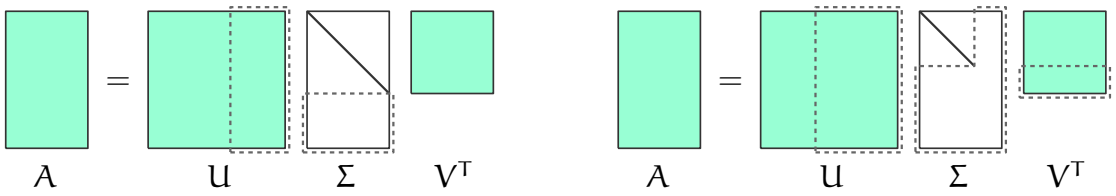
where $\Sigma_r = \mathrm{diag}(\sigma_1 \cdots \sigma_r)$, U_r is an $m \times r$ matrix consisting of the first r columns of U, and V_r is an $n \times r$ matrix consisting of the first r columns of V.

From the reduced form follows that we can write A as a sum of r rank one matrices

$$A = \sum_{j=1}^{r} \sigma_j \underline{u}_j \underline{v}_j^T, \tag{4.4}$$

where $\underline{u}_j$ and $\underline{v}_j$ are the columns of U and V respectively. We leave it as an exercise to show that $\underline{u}\,\underline{v}^T$, where $\underline{u}$ is an m-dimensional column vector and $\underline{v}^T$ an n-dimensional row vector, is a rank one matrix.

In many cases, the singular values σ_j quickly go to zero. Truncating the sum in (4.4) by discarding terms with small singular values, offers an effective means of data compression.



(a) A has full rank

(b) A has less than full rank

Figure 4.1: Removing columns from U and rows from V^T in the reduced form of the SVD.

4.4 Some theoretical consequences of the SVD

In this section we derive some of the most important properties of the SVD — properties that we will repeatedly use in the rest of the module.

Let us first recall the following fundamental subspaces related to a matrix A:

- the column space of A is spanned by its columns;
- the row space of A is spanned by its rows;
- the null space of A is spanned by all vectors $\underline{x}$ such that $A\underline{x} = \underline{0}$;
- the left null space of A is spanned by all vectors $\underline{y}$ such that $\underline{y}^T A = \underline{0}$ (note that $\underline{y}^T$ is a row vector).

Orthonormal bases for these subspaces are obtained directly from the SVD, as the following theorem states.

Theorem 4.4.1. *If there are exactly r nonzero singular values with $r \leqslant s$ then*

(i) the first r columns of U form an orthonormal basis for the column space of A,

(ii) the first r columns of V form an orthonormal basis for the row space of A,

(iii) the last $m - r$ columns of U form an orthonormal basis for the left null space of A,

(iv) the last $n - r$ columns of V form an orthonormal basis for the null space of A.

Proof. From (4.4) follows that every column of A can be written as a linear combination of the first r columns U, thus the column space of A is a subspace of the space spanned by the first r columns of U. Conversely, from (4.4) also follows that every one of the first r columns of U is a linear combination of the columns of A, implying that the first r columns of U lie in the column space of A. Thus (i) is established. A direct calculation shows that $A\underline{v} = \underline{0}$ where $\underline{v}$ is any one of the last $n - r$ columns of V. Thus they all lie in the null space of A. Conversely, suppose $A\underline{x} = \underline{0}$. It follows from the SVD (4.1) that $\Sigma V^T \underline{x} = \underline{0}$, implying that $\underline{x}$ is orthogonal to the first r columns of V. Thus $\underline{x}$ lies in the space spanned by the last $n - r$ columns of the unitary matrix V and (iv) is proven. (ii) and (iii) follow by similar arguments (simply consider A^T instead of A). □

An immediate and important consequence of this theorem is that the column space of A is orthogonal to its left null space, and the row space is orthogonal to the null space.

In practice experimental and numerical errors usually prevent the presence of singular values that are exactly zero.

The following theorem states that a good approximation of A is obtained if the singular values close to zero are ignored. In fact, if we keep only ν singular values, the resulting matrix is the best possible rank ν approximation of A (in both the L^2 and Frobenius norms).

Theorem 4.4.2. *If A_ν is the approximation of A obtained by keeping the first ν terms in (4.4), that is*

$$A_\nu = \sum_{j=1}^{\nu} \sigma_j \underline{u}_j \underline{v}_j^T, \tag{4.5}$$

then

$$\|A - A_\nu\|_2 = \inf_{\substack{B \in \mathbb{R}^{m\times n} \\ rank(B) \leqslant \nu}} \|A - B\|_2 = \sigma_{\nu+1}, \tag{4.6}$$

and

$$\|A - A_\nu\|_F = \inf_{\substack{B \in \mathbb{R}^{m\times n} \\ rank(B) \leqslant \nu}} \|A - B\|_F = \sqrt{\sigma_{\nu+1}^2 + \cdots + \sigma_s^2}, \tag{4.7}$$

where we define $\sigma_{\nu+1} = 0$ if $\nu = s$.

Proof. For a proof, see for example Trefethen and Bau[3]. □

Thus if $\sigma_{\nu+1}$ is sufficiently small it is safe to keep only the first ν singular values. Let us illustrate this with an example. Suppose

$$A = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0.01 & 1 & 1 \end{bmatrix}.$$

Calculating the SVD of A (using Python or MATLAB) we find that

$$\Sigma = \begin{bmatrix} 2.83020006 & 0 & 0 & 0 \\ 0 & 1.41494759 & 0 & 0 \\ 0 & 0 & 1.40995800 & 0 \\ 0 & 0 & 0 & 0.00306759 \\ 0 & 0 & 0 & 0 \end{bmatrix},$$

with U and V not written down. From Σ it is clear that A is of rank 4. However, the 4th singular value is quite small in comparison with the other three and it is therefore possible that it is the result of some error that sneaked into the coefficients of A. Let us therefore find the best rank 3 approximation of A. For that purpose we drop the smallest singular value from Σ to form

$$\widehat{\Sigma} = \begin{bmatrix} 2.83020006 & 0 & 0 & 0 \\ 0 & 1.41494759 & 0 & 0 \\ 0 & 0 & 1.40995800 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}.$$

[3] L. Trefethen & D. Bau, *Numerical Linear Algebra*, SIAM, 1997.

The best rank 3 approximation of A is then given by $\widehat{A} = U\widehat{\Sigma}V^T$, with U and V the left and right singular vector matrices of A respectively, and we find that

$$\widehat{A} = \begin{bmatrix} 0.999 & 1.001 & 1.001 & 0.999 \\ 1.000 & 0.000 & 1.000 & 0.000 \\ 1.001 & 0.999 & -0.001 & 0.001 \\ 0.000 & 1.000 & 0.000 & 1.000 \\ 0.001 & 0.009 & 0.999 & 1.001 \end{bmatrix},$$

written to three decimal places.

In this example errors destroyed the null space of A, and we noticed that it has rank 4. If we need an approximation of the null space of A as it was prior to destruction, the best one can do is to calculate the null space of the best rank 3 approximation of A, i.e. we calculate the null space of $\widehat{A}$. According to Theorem 4.4.1 it is one-dimensional and a basis is given by the last column of V in the SVD of A, i.e.

$$\underline{\widehat{n}} = \begin{bmatrix} -0.499 & 0.501 & 0.498 & -0.502 \end{bmatrix}^T.$$

Assuming that the original matrix, before contamination, is given by

$$A = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \end{bmatrix},$$

we calculate its null space and find that it is given by

$$\underline{n} = \begin{bmatrix} -0.5 & 0.5 & 0.5 & -0.5 \end{bmatrix}^T.$$

Since both $\underline{\widehat{n}}$ and $\underline{n}$ are normalised, a useful comparison is to calculate the angle between the two: $\cos\theta = 0.999997$, accurate to the number of digits given. This means that the two vectors very nearly point in the same direction — a very good approximation of the null space of the original, uncontaminated, matrix indeed.

4.5 The SVD and covariances

In this section we approach the SVD from a slightly different point of view. Imagine that we are given m n-dimensional vectors, $\underline{x}_j$, $j = 1, \ldots, m$ with a zero average, i.e.

$$\underline{a} = \frac{1}{m}\sum_{j=1}^{m} \underline{x}_j = \underline{0}. \tag{4.8}$$

If the average is not zero, we can always form a new system with zero average by subtracting the average from the original system.

Previously we asked questions such as to find an orthonormal basis for the subspace spanned by these vectors. Instead, let us now ask for the average direction in which the vectors are aligned. One can also think of it as the direction of maximum deviation from the origin. In this sense we are looking for a correlation between the vectors. Mathematically, we are looking for the direction $\underline{u}$ that maximizes μ where

$$\mu = \max_{\|\underline{u}\|_2=1} \frac{1}{m} \sum_{j=1}^{m} \left(\underline{u}^T \underline{x}_j\right)^2. \tag{4.9}$$

Introducing the covariance matrix

$$C = \frac{1}{m} \sum_{j=1}^{m} \underline{x}_j \underline{x}_j^T, \tag{4.10}$$

it is possible to rewrite (4.9) as

$$\mu = \max_{\|\underline{u}\|_2=1} \left(\underline{u}^T C \underline{u}\right). \tag{4.11}$$

Since C is symmetric it can be diagonalized with a unitary matrix U, i.e.

$$C = U \Lambda U^T.$$

where $\Lambda = \mathrm{diag}(\lambda_1, \ldots, \lambda_n)$ and we again order the eigenvalues λ_j in decreasing order of magnitude. It is easily shown that the eigenvalues of C are always non-negative. We may now rewrite (4.11) as

$$\mu = \max_{\|\underline{u}\|_2=1} \left(\underline{u}^T U \Lambda U^T \underline{u}\right) = \max_{\|\underline{y}\|_2=1} \left(\underline{y}^T \Lambda \underline{y}\right) \tag{4.12}$$

where $\underline{y} = U^T \underline{u}$. The fact that U is unitary ensures that $\|\underline{y}\|_2 = \|\underline{u}\|_2 = 1$. Thus we are required to calculate

$$\mu = \max_{\|\underline{y}\|_2=1} \sum_{j=1}^{n} \lambda_j |y_j|^2 \quad \text{subject to} \quad \sum_{j=1}^{n} |y_j|^2 = 1. \tag{4.13}$$

It follows that $\mu = \lambda_1$ and $\underline{y} = \underline{e}_1$ where $\underline{e}_j$ has a one in its j-th entry, the rest are all zeros. Since $\underline{u} = U\underline{y} = U\underline{e}_1$, it follows that the direction of maximum deviation, $\underline{u}$, we were looking for is exactly the eigenvector $\underline{u}_1$ of the covariance matrix C, belonging to the largest eigenvalue λ_1. Moreover, λ_1 provides a measure for the deviation in the direction of $\underline{u}_1$.

Let us now ask for the direction of maximum deviation orthogonal to $\underline{u}_1$. Thus we are again required to calculate (4.13) but this time subject to $\sum_{j=1}^{n} |y_j|^2 = 1$ *and* $\underline{e}_1^T \underline{y} = 0$. Since this implies that the first component of $\underline{y}$ equals zero, i.e. $y_1 = 0$, it follows that (4.13) is calculated subject to $\sum_{j=2}^{n} |y_j|^2 = 1$. This is achieved by $\underline{y} = \underline{e}_2$, or $\underline{u} = U\underline{e}_2 = \underline{u}_2$, the eigenvector of C belonging to the second largest eigenvalue.

Continuing in this fashion we arrive at the following result: the first eigenvector of the covariance matrix C points in the direction of maximum variation and the corresponding eigenvalue measures the variation in this direction (the *variance* in this direction). The subsequent eigenvectors point in the directions of maximum variation orthogonal to the previous directions with their associated eigenvalues again measuring the variations in these directions.

Finally notice that the same results are obtained by calculating the SVD of the matrix

$$X = \frac{1}{\sqrt{m}}[\underline{x}_1 \ \cdots \ \underline{x}_m]. \tag{4.14}$$

The directions of maximum variation are then given by the columns of U and the variances by the squares of the singular values.

Let us now illustrate these ideas with an example. Consider the problem of finding a face in an image. One possible (rather naive) approach would be to identify a number of objects that might qualify and measure their width and height. It is easy to imagine that the measurements of any face should fall within a certain range. Anything outside the 'typical' range can be discarded. Anything inside the range can then be investigated in more detail for further face-like characteristics. The problem is to find the 'typical' range of measurements for faces[4].

Figure 4.2 shows the actual measurements of a number of faces from a student population. The width is measured from ear-to-ear and the height from the chin to between the eyes. Note the natural distribution of the sizes of the faces around a certain mean value. Given a particular measurement, the question is whether it belongs to this distribution. It should be clear that one cannot simply use the distance from the mean — the ellipse clearly gives a better indication of the *nature* of the distribution. It therefore makes more sense to classify measurements in the vicinity of the ellipse as belonging to the distribution. Of course this does not mean that it is actually a face, only that it is possibly a face and that a more detailed investigation is in order.

In this case the mean of the 48 facial measurements $\underline{x}_j = [x_j, y_j]^T$ is calculated, $\underline{a} = \sum_{j=1}^{48} \underline{x}_j$, and the matrix X is defined by

$$X = \frac{1}{\sqrt{48}} \left[(\underline{x}_1 - \underline{a}) \ \cdots \ (\underline{x}_{48} - \underline{a}) \right].$$

The SVD of X now gives the 2×2 matrices U and Σ. The columns of U give the principal axes of the ellipse and in this case the size of the ellipse is two times the standard deviation.

[4] Although this example is contrived (the number of features we measure is too low) it is not entirely unrealistic. In fact, one of the earliest identification systems, and at the time a serious rival for fingerprints, was based on a comprehensive set of measurements of individuals.

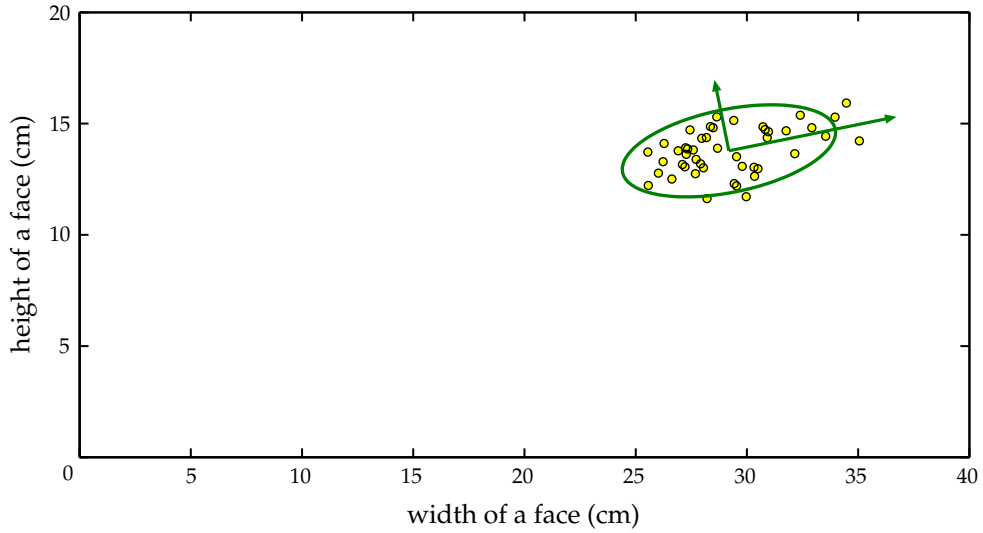


Figure 4.2: Face measurements.

For this example we find that

$$U = \begin{bmatrix} -0.9778 & -0.2095 \\ -0.2095 & 0.9778 \end{bmatrix} \quad \text{and} \quad \Sigma = \begin{bmatrix} 2.43 & 0 \\ 0 & 0.91 \end{bmatrix}.$$

Incidentally, what happens if the mean is not removed from the measurements?

4.6 Application: a basic face recognition system

In this section we provide details of a classic algorithm for automatic face recognition, specifically to illustrate the power of some of the theoretical consequences of the SVD we have discussed so far. This algorithm is based on one introduced by Sirovich and Kirby[5], which has been developed into a baseline for image based face recognition systems.

The aim of the technique is to represent face images in an efficient manner, and to then compare and match those representations rather than the raw images directly.

Suppose the pixels in a face image are stored in the $p \times q$ matrix A. We stack the columns of A sequentially into a vector of length $m = pq$. Note that all $p \times q$ matrices can be reshaped in this way, so that they all occupy an m-dimensional space.

[5] I. Sirovich & M. Kirby, *Low-dimensional procedures for the characterization of human faces*, Journal of the Optical Society of America, 4:519–524, 1987.

The value of m may be quite large, e.g. for 256×256 images we have $m = 65,536$. There is a possibility, however, that the specific matrices under consideration in a face recognition system occupy (by approximation) some lower dimensional subspace. To find such a space we "train" the system on a representative set of face images and then use the SVD to calculate a reduced basis for the space they span.

To this end, consider a collection of n vectors with m components each, $\underline{f}_i$, $i = 1, \ldots, n$. This collection will be called the *training set* of the system, and n is typically much smaller than m. We calculate the *average face* as

$$\underline{a} = \frac{1}{n} \sum_{i=1}^{n} \underline{f}_i, \tag{4.15}$$

and subtract it from every vector in the training set to obtain column vectors $\underline{x}_i = \underline{f}_i - \underline{a}$, $i = 1, \ldots, n$. An $m \times n$ matrix X is built as

$$X = \begin{bmatrix} \underline{x}_1 & \cdots & \underline{x}_n \end{bmatrix}. \tag{4.16}$$

The average is subtracted because vectors constructed from similar images (such as face images) are likely to be clustered around their average, distant from the origin. We wish to determine an orthogonal basis for the space spanned by them, and centring around the origin improves the ability of such a basis to describe a larger range of vectors.

Finding an orthogonal basis for the training set is now a matter of finding a basis for the column space of X. Because the columns are somewhat similar with respect to the entire m-dimensional space, the singular values of X should decrease rapidly. As described in section 4.4 we can now infer an approximate dimension α for the column space by regarding singular values below a certain cut-off as zero.

The first α columns of U in the SVD of X are called the *eigenfaces* of the training set and span, by approximation, the column space of X. Note that since $\alpha < n$ and $n \ll m$, we have $\alpha \ll m$.

If the training set is sufficiently representative the vectors constructed from arbitrary face images should also be contained within that α-dimensional subspace. Indeed, an arbitrary m-dimensional vector $\underline{f}$ is projected orthogonally onto the subspace spanned by the eigenfaces by solving for $\underline{y}$ the over-determined linear system $U_\alpha \underline{y} = \underline{f} - \underline{a}$, in a least squares sense. Here U_α contains the eigenfaces as columns. Therefore, since those columns are orthogonal,

$$\underline{y} = U_\alpha^T (\underline{f} - \underline{a}). \tag{4.17}$$

The $\alpha \times 1$ vector $\underline{y}$ in the above expression is called the *eigenface representation* of $\underline{f}$. We can reconstruct $\underline{f}$ from its eigenface representation as

$$\widehat{\underline{f}} = U_\alpha \underline{y} + \underline{a}. \tag{4.18}$$

Note that $\underline{y}$ is determined as a least squares solution so that, in general, $\widehat{\underline{f}}$ is slightly different from the original vector $\underline{f}$. However if the training set is representative then $\widehat{\underline{f}}$ should include sufficient information to distinguish it from other individuals.

For the recognition of face images we now compare eigenface representations, which carry more distinctiveness within the class of all faces, rather than the images directly. The distance between two eigenface representations $\underline{y}_1$ and $\underline{y}_2$ can for example be measured as the L^2 norm $\|\underline{y}_1 - \underline{y}_2\|$, and if this value is lower than some threshold the system can classify the two faces as belonging to the same person.

In practice the matrix U_α and the vector $\underline{a}$ are calculated once, and when an unknown face is to be recognized we use (4.17) to determine the eigenface representation which is then compared to the ones in a database of known identities.

Figure 4.3(a) shows a few examples from a database[6] taken of 40 individuals. The calculated average face is shown in (b), reshaped into a $p \times q$ image, and a few eigenfaces (columns of U_α) in (c).

Finally we remark that the process of generating the small set of eigenfaces from a large training set is an example of a technique often used in pattern recognition called *principal component analysis* (PCA).

(a) the first five face images of our training set

(b) average

(c) the first four eigenfaces, reshaped to images

Figure 4.3: Examples of face images from the AT&T database, an average face and the first few eigenfaces (reshaped and represented as images).

[6]The AT&T Database of Faces,
`www.cl.cam.ac.uk/research/dtg/attarchive/facedatabase.html`.

4.7 Linear systems of equations

In this section we consider solutions of the following system of equations:

$$A\underline{x} = \underline{b}, \tag{4.19}$$

where A is an $m \times n$ matrix, $\underline{x} \in \mathbb{R}^n$ and $\underline{b} \in \mathbb{R}^m$. If $m = n$ the well-known situation is relatively straightforward, either $\det A \neq 0$ in which case a unique solution exists or $\det A = 0$ in which case no solution exists. If however we insist on a solution of the system, for any choice of A and $\underline{b}$, the situation becomes a little more complicated. Let us illustrate the difficulties with a few examples.

1. Solve for $x = 1$. This example is so trivial that it hardly warrants comment. The point is that it is of the form (4.19) with $m = n = 1$. Since the determinant is non-zero, the solution is trivially obtained.

2. Let us now complicate things slightly and solve simultaneously for

$$\begin{aligned} x &= 1 \\ x &= 2. \end{aligned}$$

   At first glance this does not make any sense at all, x cannot simultaneously equal 1 and 2. Yet, it is of the form (4.19) with $A = \begin{bmatrix} 1 & 1 \end{bmatrix}^T$, and $\underline{b} = \begin{bmatrix} 1 & 2 \end{bmatrix}^T$. If forced to, we might choose $x = 1.5$ as the best compromise. This turns out to be the least squares solution.

3. At the other extreme we may now want to solve for $x + y = 1$. Again this is of the form (4.19) with $A = \begin{bmatrix} 1 & 1 \end{bmatrix}$ and $\underline{b} = 1$. Now the problem is that we have an infinite number of solutions, and we have to select one. The question is to identify a natural choice.

We start with the least squares solution.

4.7.1 Over-determined systems and least squares

An over-determined linear system can be written graphically as

$$\underset{m\times n}{\begin{bmatrix} \\ A \\ \\ \end{bmatrix}} \underset{n\times 1}{\begin{bmatrix} \\ \underline{x} \\ \\ \end{bmatrix}} = \underset{m\times 1}{\begin{bmatrix} \\ \underline{b} \\ \\ \end{bmatrix}} \tag{4.20}$$

where $m > n$. Typically, these systems lack a solution — there are too many constraints imposed on the variables $x_1, x_2, \ldots, x_n$. The question to ask is not what vector $\underline{x}$ makes $A\underline{x} - \underline{b} = \underline{0}$, but what vector $\underline{x}$ makes $\|A\underline{x} - \underline{b}\|$ as small as possible (unless otherwise stated $\|\cdot\|$ refers to the L^2 norm). If we multiply (4.20) from the left with A^T, we obtain a square linear system known as the *normal equations*.

Theorem 4.7.1. *If A is of full rank, the solution to the normal equations*

$$A^T A\underline{x} = A^T \underline{b} \tag{4.21}$$

is unique and provides the least squares solution to $A\underline{x} = \underline{b}$.

Proof. We need to solve the system $A\underline{x} = \underline{b}$ where $\underline{b}$ is outside the column space of A. Therefore an exact solution does not exist. The idea is to find the solution of an alternative system $A\underline{x} = \hat{\underline{b}}$ where $\hat{\underline{b}}$ is the vector in the column space of A as close as possible to $\underline{b}$. This will of course be the case if $\underline{e} = \underline{b} - A\underline{x}$ is orthogonal to the column space of A. It means that $\underline{e}$ is in the left null space of A, or $A^T\underline{e} = \underline{0}$, which is exactly the normal equations (4.21). In order to show that A^TA is non-singular, consider $A^TA\underline{x} = \underline{0}$. Therefore $A\underline{x}$ lies in both the column and left null space of A. Since they are orthogonal it follows that $A\underline{x} = \underline{0}$. On the other hand, the fact that A is of full rank implies that its null space consists of only the zero element. Thus $A^TA\underline{x} = \underline{0}$ implies that $\underline{x} = \underline{0}$, showing that A^TA is non-singular. □

The coefficient matrix in the normal equations is of the form A^TA. This is always a symmetric matrix. Assuming A has rank n (we discuss the general case in the next section), it furthermore is positive definite since $\underline{y}^T(A^TA)\underline{y} = \left\|A\underline{y}\right\|^2 > 0$ whenever $\underline{y} \neq \underline{0}$. This makes a couple of very effective numerical methods available for solving the normal equation system:

- *Cholesky's method* is conceptually similar to Gaussian elimination, but requires no pivoting and has about half the operation count.
- The *conjugate gradient method* is iterative, but often converges very fast. In this algorithm, one needs to repeatedly multiply the coefficient matrix with different vectors — no other operations are performed with the matrix entries. It makes this method particularly attractive if the matrix is sparse, where zero entries can be utilized to save on arithmetic operations.

The big disadvantage with the normal equation approach is that, in some cases, the resulting system can be very sensitive to small errors in A and $\underline{b}$. The process of forming A^TA can lead to an ill-conditioned linear system.

With A split as $A = QR$, we get $A\underline{x} - \underline{b} = QR\underline{x} - \underline{b}$ and (since multiplying any vector by a unitary matrix leaves the L^2 norm unchanged)

$$\|A\underline{x} - \underline{b}\| = \left\|R\underline{x} - Q^T\underline{b}\right\|. \tag{4.22}$$

At this point, a picture illustrating the 'shape' of $R\underline{x} - Q^T\underline{b}$ helps, see Figure 4.4. Regarding the components in the vector $R\underline{x} - Q^T\underline{b}$: choosing different $\underline{x}$ does not change the last $m - n$ rows and we can make the first n rows all zero by solving $R\underline{x} = \underline{b}_1$ where $\underline{b}_1$ is the first n rows of $Q^T\underline{b}$. It should be clear that this minimizes the norm of the residual. The cost in terms of arithmetic operations is somewhat larger than in the normal equations approach, but the algorithm can be shown to be entirely numerically stable.

Let us return to the example mentioned before and find the solution of $x = 1$ and $x = 2$. The normal equations are

$$\begin{bmatrix} 1 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \end{bmatrix} x = \begin{bmatrix} 1 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 2 \end{bmatrix},$$

or $x = 1.5$.

4.7.2 Solution by SVD factorization and the generalized inverse

This procedure is almost identical to the one based on the QR factorization. After noting that

$$\|A\underline{x} - \underline{b}\| = \left\|\Sigma V^T\underline{x} - U^T\underline{b}\right\|, \tag{4.23}$$

we ignore, as before, the last $m - n$ rows and solve the resulting square $n \times n$ system. However, we can do even better. Let us now consider the completely general situation. In this case we want to 'solve' the system $A\underline{x} = \underline{b}$ where $\underline{b}$ does not necessarily lie the column space of A and/or the null space is not empty, i.e. A may also be rank deficient.

We already know the answer to the problem of $\underline{b}$ not being in the column space of A: we project $\underline{b}$ orthogonally onto the column space, i.e. we write

$$\underline{b} = \underline{b}_c + \underline{b}_l, \tag{4.24}$$

with $\underline{b}_c$ and $\underline{b}_l$ in the column and left null spaces of A, respectively. Solving $A\underline{x} = \underline{b}_c$ amounts to solving the normal equations (4.21). However, since A is rank deficient (its null space is not the trivial one), there is not a unique solution

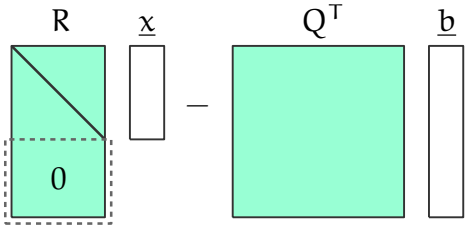


Figure 4.4: The shape of $R\underline{x} - Q^T\underline{b}$.

for this system — adding any element of the null space to a solution gives another solution. For example, let $\underline{x}_p$ be any solution of $A\underline{x}_p = \underline{b}_c$. Clearly $\underline{x} = \underline{x}_p + \underline{x}_n$ for any $\underline{x}_n \in \text{null}(A)$ is also a solution. However, it is possible to identify among all these solutions a unique one with special properties. Suppose we decompose $\underline{x}_p$ and write it (uniquely) as the sum of its orthogonal parts in the row and null spaces of A,

$$\underline{x}_p = \underline{x}_r + \underline{x}_n. \tag{4.25}$$

Then $A\underline{x}_r = \underline{b}_c$, i.e. $\underline{x}_r$ is also a solution. The point is, $\underline{x}_r$ is the *unique* solution in the row space of A. Suppose there is another solution in $\text{row}(A)$, $A\underline{y}_r = \underline{b}_c$. We find that $A(\underline{x}_r - \underline{y}_r) = \underline{0}$, showing that $\underline{x}_r - \underline{y}_r$ lies in both the row and the null space of A. Since these spaces are orthogonal, $\underline{x}_r - \underline{y}_r$ is orthogonal to itself so it has to be zero.

We have now identified a candidate for a unique solution of the general system $A\underline{x} = \underline{b}$, namely the unique vector in the row space of A that solves the system obtained from projecting $\underline{b}$ orthogonally onto the column space of A. Any other solution of this system adds an orthogonal component from the null space of A to $\underline{x}_r$, i.e. any other solution has a larger L^2 norm. Thus the unique element in $\text{row}(A)$ is also the solution with the smallest norm. It only remains to find a formula for $\underline{x}_r$. For this the SVD is particularly convenient.

Using the reduced form of the SVD, we substitute

$$A = U_r \Sigma_r V_r^T \tag{4.26}$$

in the normal equations to obtain

$$V_r^T \underline{x} = \Sigma_r^{-1} U_r^T \underline{b}. \tag{4.27}$$

Since A is not necessarily of full rank this may be an under-determined system with possibly an infinite number of solutions. As explained above, a unique solution exists if we restrict ourselves to the row space of A, i.e. if we let $\underline{x}_r = A^T \underline{y} = V_r \Sigma_r U_r^T \underline{y}$ for some $\underline{y}$. Substitution into the expression above yields

$$\Sigma_r U_r^T \underline{y} = \Sigma_r^{-1} U_r^T \underline{b}. \tag{4.28}$$

Pre-multiplying with V_r yields the solution

$$\underline{x}_r = V_r \Sigma_r^{-1} U_r^T \underline{b}. \tag{4.29}$$

Note that in case A is (square and) non-singular, this solution simply becomes $\underline{x} = A^{-1}\underline{b}$. In case it is rectangular ($m < n$) and of full rank, it reduces to the standard linear least squares solutions. And it also provides a unique solution in the general case when A is any $m \times n$ matrix.

We call the matrix $V_r \Sigma^{-1} U_r^T$ the *generalized inverse* (or *pseudo-inverse*) of A, and denote it by A^+. Formally, if $U_r \Sigma_r V_r^T$ is the reduced SVD of some $m \times n$ matrix A, then the generalized inverse of A can be calculated as

$$A^+ = V_r \Sigma^{-1} U_r^T. \tag{4.30}$$

Although the SVD provides the most general answer in terms of the generalized inverse, it should be kept in mind that it is relatively expensive to compute. For example, the operation count in computing the QR factorization ($\frac{4}{3}n^3$ for a full $n \times n$ matrix A) is significantly lower than for the SVD ($4n^3$ for Σ only, $26n^3$ for U, Σ, V; these numbers are approximate since SVD procedures are iterative). For this reason, QR factorization is usually the preferred approach for least squares problems when A is of full rank.

Returning to the example at the beginning of the section we find the generalized solution of $x + y = 1$. The null space is given by $x + y = 0$, and $\begin{bmatrix} 1 & -1 \end{bmatrix}^T$ is a basis of the null space. We are looking for a solution orthogonal to the null space or, equivalently, a solution in the row space of A. A basis for the row space is $\begin{bmatrix} 1 & 1 \end{bmatrix}^T$. This implies that the solution we are looking for should satisfy $x = y$, i.e. the generalized solution is $\begin{bmatrix} x & y \end{bmatrix}^T = \frac{1}{2}\begin{bmatrix} 1 & 1 \end{bmatrix}^T$. Figure 4.5 clarifies.

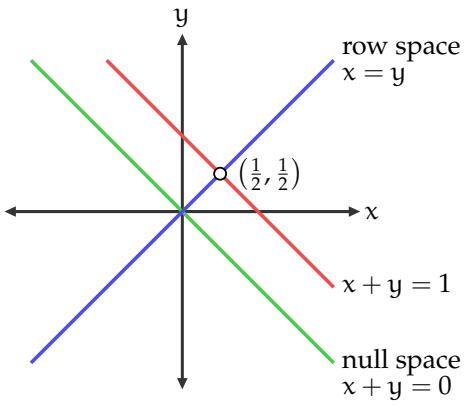


Figure 4.5: The generalized solution of $x + y = 1$.

As additional exercise, consider the system

$$\begin{aligned} x + y &= 1 \\ 2x + 2y &= 1. \end{aligned}$$

1. Write the system in the form $A\underline{x} = \underline{b}$.
2. Calculate the four fundamental spaces of A, and draw them in a figure.
3. Can you calculate a least squares solution of the system? Why not?
4. Plot $\underline{b}$ in your figure, and note that it does not lie in the column space of A. Find the orthogonal projection of $\underline{b}$ on the column space of A. Call it $\widehat{\underline{b}}$.
5. Find all the solutions of $A\underline{x} = \widehat{\underline{b}}$, and draw them in your figure. Find the generalized solution and plot it in the figure.
6. Write down the SVD of A without calculating any eigenvalues and eigenvectors. Explain how you figure out the singular values of A.

Chapter 5

Projective geometry

This chapter is devoted to the theory of planar geometry and the 2D projective plane, as well as the concept of projective transformation. The set of tools we obtain through this discussion will enable a natural and very convenient way of modelling the process of image formation with pinhole cameras.

5.1 Planar geometry and the 2D projective plane

We usually represent a point in the plane by a pair of coordinates $\underline{x} = \begin{bmatrix} x_1 & x_2 \end{bmatrix}^T$, thus we often identify the plane with $\mathbb{R}^2$. In this section we introduce homogeneous coordinates as a representation of points in a plane. This unifies the concept of the intersection of two lines (even parallel lines will be shown to intersect in a well-defined point) and leads us straight into the projective plane, $\mathbb{P}^2$, with its well-defined geometric computations.

5.1.1 Homogeneous representation of lines

A line in the plane is represented by an equation of the form

$$ax + by + c = 0, \tag{5.1}$$

with different choices of a, b and c yielding different lines. For example, the lines

$$2x + y - 1 = 0,$$

with $a = 2$, $b = 1$ and $c = -1$, and

$$x - 1 = 0,$$

with $a = 1$, $b = 0$ and $c = -1$, are illustrated in Figure 5.1.

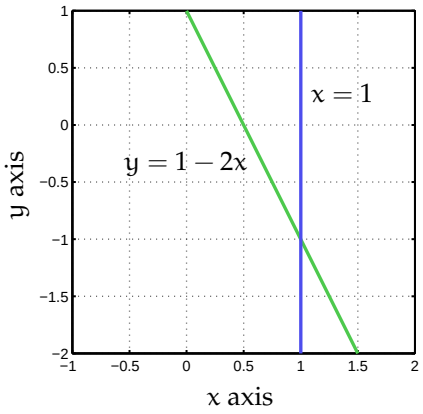


Figure 5.1: Two lines in the plane $\mathbb{R}^2$.

A line can be uniquely represented by the coefficient vector $\underline{l} = \begin{bmatrix} a & b & c \end{bmatrix}^T$. Using this representation, the lines in Figure 5.1 are given by $\underline{l} = \begin{bmatrix} 2 & 1 & -1 \end{bmatrix}^T$ and $\underline{l}' = \begin{bmatrix} 1 & 0 & -1 \end{bmatrix}^T$.

Using this notation, the line represented by the vector $k\underline{l} = \begin{bmatrix} ka & kb & kc \end{bmatrix}^T$, with k any nonzero constant, is the same as the line represented by the vector $\underline{l} = \begin{bmatrix} a & b & c \end{bmatrix}^T$, since x and y satisfy (5.1) if and only if they satisfy

$$kax + kby + kc = 0, \qquad k \neq 0. \tag{5.2}$$

For this reason we consider the vectors $\begin{bmatrix} a & b & c \end{bmatrix}^T$ and $k \begin{bmatrix} a & b & c \end{bmatrix}^T (k \neq 0)$ to be equivalent. An equivalence class of vectors under this equivalence relationship is known as a *homogeneous vector*, where any particular vector $\begin{bmatrix} a & b & c \end{bmatrix}^T$ is a representative of a whole class of equivalent vectors. The set of equivalence classes in $\mathbb{R}^3 \setminus \{\underline{0}^3\}$ (the vector space $\mathbb{R}^3$ with the zero vector $\underline{0}^3$ removed) forms the *projective space* $\mathbb{P}^2$.

5.1.2 Homogeneous representation of points

A point $\begin{bmatrix} x & y \end{bmatrix}^T$ lies on the line $\underline{l} = \begin{bmatrix} a & b & c \end{bmatrix}^T$ if and only if

$$ax + by + c = 0, \tag{5.3}$$

which can be written in terms of an inner product

$$\begin{bmatrix} x & y & 1 \end{bmatrix} \begin{bmatrix} a \\ b \\ c \end{bmatrix} = 0. \tag{5.4}$$

Here the point $\begin{bmatrix} x & y \end{bmatrix}^T$ in $\mathbb{R}^2$ is represented by a 3-vector by adding a third coordinate of 1. Note that, for any nonzero constant k, we have

$$\begin{bmatrix} kx & ky & k \end{bmatrix} \begin{bmatrix} a \\ b \\ c \end{bmatrix} = 0 \tag{5.5}$$

if and only if (5.4) holds. It is therefore natural to consider the vector $\begin{bmatrix} kx & ky & k \end{bmatrix}^T$ (with $k \neq 0$) equivalent to $\begin{bmatrix} x & y & 1 \end{bmatrix}^T$. We therefore have, similar to the case of lines, an equivalence class where the vector $\begin{bmatrix} x & y & 1 \end{bmatrix}^T$ represents the entire class. Here, an arbitrary homogeneous vector $\begin{bmatrix} x_1 & x_2 & x_3 \end{bmatrix}^T$ represents the point $\begin{bmatrix} \frac{x_1}{x_3} & \frac{x_2}{x_3} \end{bmatrix}^T$ in $\mathbb{R}^2$. Points, written as homogeneous vectors, are also elements of $\mathbb{P}^2$.

Theorem 5.1.1. *The point* $\underline{x} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}$ *lies on the line* $\underline{l} = \begin{bmatrix} a \\ b \\ c \end{bmatrix}$ *if and only if*

$$\underline{x}^T \underline{l} = ax_1 + bx_2 + cx_3 = 0.$$

It is clear, in Theorem 5.1.1, that in order to uniquely specify a point in $\mathbb{R}^2$ we need to specify two ratios (x_1/x_3 and x_1/x_3) in $\mathbb{P}^2$ or two values (an x-coordinate and a y-coordinate) in $\mathbb{R}^2$. In a similar manner, a line is specified by two parameters and so has two degrees of freedom. For example, in an inhomogeneous representation of a line, we could uniquely specify a line by its gradient and y-intercept.

Homogeneous vector representations of lines and points allow us to simply express some previously clumsy concepts, for example the intersection of two lines is simply the cross product of the two homogeneous vectors representing the lines, as described in the next theorem.

Theorem 5.1.2. *The intersection point of two lines* $\underline{l} = \begin{bmatrix} a \\ b \\ c \end{bmatrix}$ *and* $\underline{l}' = \begin{bmatrix} a' \\ b' \\ c' \end{bmatrix}$ *is the point* $\underline{x} = \underline{l} \times \underline{l}'$, *where* $\times$ *represents the vector cross product.*

As an example, suppose we wish to determine the intersection point of the lines $x = 1$ and $y = x + 1$. The line $x = 1$ is written as $-x + 0y + 1 = 0$, and thus has the homogeneous representation $\underline{l} = \begin{bmatrix} -1 & 0 & 1 \end{bmatrix}^T$, while the line $y = x + 1$ is written as $-x + y - 1 = 0$, and thus has the homogeneous representation $\underline{l}' = \begin{bmatrix} -1 & 1 & -1 \end{bmatrix}^T$. Then, from Theorem 5.1.2, we calculate the intersection point as

$$\underline{x} = \underline{l} \times \underline{l}' = \begin{vmatrix} \underline{i} & \underline{j} & \underline{k} \\ -1 & 0 & 1 \\ -1 & 1 & -1 \end{vmatrix} = \begin{bmatrix} -1 \\ -2 \\ -1 \end{bmatrix},$$

which is the homogeneous representation of the point $\begin{bmatrix}1 & 2\end{bmatrix}^T$, consistent with the intersection obtained in Figure 5.2(a).

Theorem 5.1.3. *The line joining two points $\underline{x}$ and $\underline{x}'$ is given by $\underline{l} = \underline{x} \times \underline{x}'$, where $\times$ represents the vector cross product.*

Suppose we want to determine the line joining the points $\underline{x} = (0,1)$ and $\underline{x}' = (1,0)$. The homogeneous representation of $\underline{x} = (0,1)$ is $\begin{bmatrix}0 & 1 & 1\end{bmatrix}^T$, and of $\underline{x}' = (1,0)$ is $\begin{bmatrix}1 & 0 & 1\end{bmatrix}^T$. From Theorem 5.1.3 we calculate the homogeneous representation of the line joining the two points as

$$\underline{l} = \underline{x} \times \underline{x}' = \begin{vmatrix} \underline{i} & \underline{j} & \underline{k} \\ 0 & 1 & 1 \\ 1 & 0 & 1 \end{vmatrix} = \begin{bmatrix} 1 \\ 1 \\ -1 \end{bmatrix}$$

or, equivalently, $x_1 + x_2 - x_3 = 0$. The inhomogeneous representation is then $x + y - 1 = 0$ or, equivalently, $y = -x + 1$. Figure 5.2(b) illustrates.

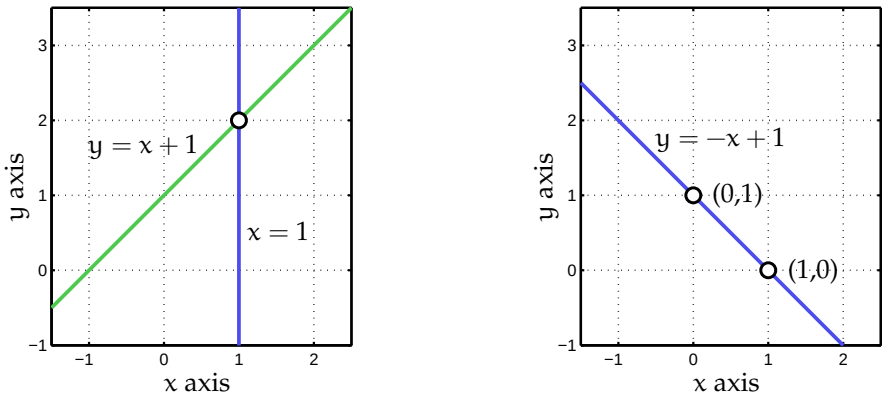


Figure 5.2: (a) intersecting lines and (b) a line joining two points in the plane $\mathbb{R}^2$.

5.1.3 Intersection of parallel lines

Consider two parallel lines

$$ax + by + c = 0 \quad \text{and} \quad ax + by + c' = 0.$$

These two lines are represented by the vectors $\underline{l} = \begin{bmatrix}a & b & c\end{bmatrix}^T$ and $\underline{l}' = \begin{bmatrix}a & b & c'\end{bmatrix}^T$ for which the first two coordinates are the same. Although these lines are parallel, we can compute their intersection $\underline{x}$ using Theorem 5.1.2,

$$\underline{x} = \underline{l} \times \underline{l}' = \begin{vmatrix} \underline{i} & \underline{j} & \underline{k} \\ a & b & c \\ a & b & c' \end{vmatrix} = \begin{bmatrix} bc' - bc \\ ac - ac' \\ 0 \end{bmatrix} = (c' - c) \begin{bmatrix} b \\ -a \\ 0 \end{bmatrix}.$$

Since we are working in homogeneous coordinates, we consider any nonzero multiple of a vector to be equivalent to the original vector, i.e. the point of intersection is

$$\underline{x} = \begin{bmatrix} b \\ -a \\ 0 \end{bmatrix}.$$

If we now try to find the inhomogeneous point of which $\underline{x} = \begin{bmatrix} b & -a & 0 \end{bmatrix}^T$ is the homogeneous representation, we would divide by zero to get $\begin{bmatrix} \frac{b}{0} & -\frac{a}{0} \end{bmatrix}^T$, which makes no sense, unless to suggest that the point of intersection has infinitely large coordinates. This observation agrees with the usual idea, that parallel lines intersect at infinity.

Consider for example the parallel lines $x = 1$ and $x = 2$ with homogeneous representations $\underline{l} = \begin{bmatrix} -1 & 0 & 1 \end{bmatrix}^T$ and $\underline{l}' = \begin{bmatrix} -1 & 0 & 2 \end{bmatrix}^T$. The intersection point of these two lines is at the point

$$\underline{x} = \underline{l} \times \underline{l}' = \begin{vmatrix} \underline{i} & \underline{j} & \underline{k} \\ -1 & 0 & 1 \\ -1 & 0 & 2 \end{vmatrix} = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix},$$

which is the point at infinity in the direction of the y-axis.

In general, points with homogeneous coordinates $\begin{bmatrix} x & y & 0 \end{bmatrix}^T$ do not correspond to any finite point in $\mathbb{R}^2$.

5.1.4 Ideal points and the line at infinity

The projective space $\mathbb{P}^2$ consists of all homogeneous vectors $\underline{x} = \begin{bmatrix} x_1 & x_2 & x_3 \end{bmatrix}^T$, where $\underline{x}$ represents a finite point in $\mathbb{R}^2$ if $x_3 \neq 0$. If $x_3 = 0$ we say $\underline{x}$ is an *ideal point* (or a point at infinity).

The set of all ideal points may be written as $\begin{bmatrix} x_1 & x_2 & 0 \end{bmatrix}^T$. Note that this set lies on a single line $\underline{l} = \begin{bmatrix} 0 & 0 & 1 \end{bmatrix}^T$, since

$$\begin{bmatrix} x_1 & x_2 & 0 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} = 0. \tag{5.6}$$

We call this line the *line at infinity* and denote it by $\underline{l}_\infty = \begin{bmatrix} 0 & 0 & 1 \end{bmatrix}^T$.

The line $\underline{l} = \begin{bmatrix} a & b & c \end{bmatrix}^T$ intersects the line at infinity $\underline{l}_\infty = \begin{bmatrix} 0 & 0 & 1 \end{bmatrix}^T$ at

$$\underline{x} = \underline{l} \times \underline{l}_\infty = \begin{bmatrix} b & -a & 0 \end{bmatrix}^T, \tag{5.7}$$

which is an ideal point. The line $\underline{l}' = \begin{bmatrix} a & b & c' \end{bmatrix}^T$, parallel to $\underline{l}$, intersects $\underline{l}_\infty$ at the same point.

In inhomogeneous coordinates, the vector $\begin{bmatrix} b & -a \end{bmatrix}^T$ is tangent to the line $ax + by + c = 0$, and orthogonal to the line's normal, and so represents the line's direction. If we were to vary the line's direction, the ideal point $\begin{bmatrix} b & -a & 0 \end{bmatrix}^T$ would also vary over $\underline{l}_\infty$. Thus the line at infinity can be thought of as a set of directions of lines in the plane.

5.1.5 The duality principle

Theorem 5.1.4. *For any theorem of 2-dimensional projective geometry there is a dual theorem, which may be derived by interchanging the roles of points and lines in the original theorem.*

5.1.6 A useful way to think of $\mathbb{P}^2$

The study of the geometry of $\mathbb{P}^2$ is known as projective geometry.

A useful way of thinking of $\mathbb{P}^2$ is as a set of rays in $\mathbb{R}^3$, see Figure 5.3. The set of vectors $k\begin{bmatrix} x_1 & x_2 & x_3 \end{bmatrix}^T$ (with k varying) forms a ray through the origin. Such a ray can be thought of representing a single point in $\mathbb{P}^2$.

The corresponding point in $\mathbb{R}^2$ may be obtained by intersecting a particular ray with the plane $x_3 = 1$, i.e. there is a one-to-one correspondence between $\mathbb{R}^2 \setminus \{\underline{0}^2\}$ and $\mathbb{P}^2$:

$$\text{if } \begin{bmatrix} x_1 & x_2 \end{bmatrix}^T \in \mathbb{R}^2 \Rightarrow \begin{bmatrix} x_1 & x_2 & 1 \end{bmatrix}^T \in \mathbb{P}^2$$

and

$$\text{if } \begin{bmatrix} x_1 & x_2 & x_3 \end{bmatrix}^T \in \mathbb{P}^2 \Rightarrow \begin{bmatrix} \frac{x_1}{x_3} & \frac{x_2}{x_3} \end{bmatrix}^T \in \mathbb{R}^2.$$

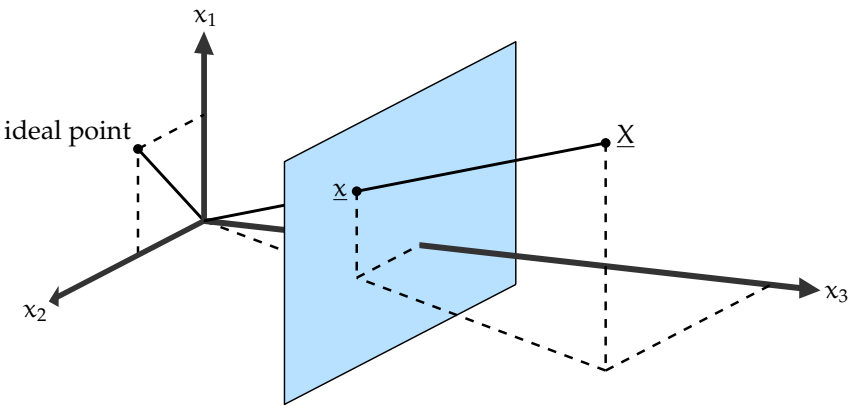


Figure 5.3: A model of the projective plane.

5.2 Projective transformations

A projective transformation is an invertible map h from $\mathbb{P}^2$ to $\mathbb{P}^2$ such that straight lines are mapped to straight lines or, equivalently, such that three points $\underline{u}$, $\underline{v}$ and $\underline{w}$ lie on a straight line if and only if the three points $h(\underline{u})$, $h(\underline{v})$ and $h(\underline{w})$ lie on a straight line.

A projective transformation is also called a *collineation*, a *projectivity* or a *homography*.

Note that in the definition above we have emphasised that straight lines go to straight lines. Since we only work with straight lines, we use the convention that when we say *lines* we mean *straight lines* for the remainder of this chapter.

Theorem 5.2.1. *A mapping $h : \mathbb{P}^2 \to \mathbb{P}^2$ is a projective transformation if and only if there exists a nonsingular, homogeneous 3×3 matrix H such that for any point in $\mathbb{P}^2$ represented by a vector $\underline{x}$ it is true that*

$$h(\underline{x}) = H\underline{x}.$$

This theorem allows us to define a projective transformation in terms of matrix multiplication. A projective transformation is a linear transformation on homogeneous 3-vectors represented by a nonsingular, homogeneous 3×3 matrix,

$$\begin{bmatrix} x_1' \\ x_2' \\ x_3' \end{bmatrix} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}, \tag{5.8}$$

or more briefly, $\underline{x}' = H\underline{x}$.

Note that, since the vectors $\underline{x}$ and $\underline{x}'$ are homogeneous, the projective transformation $\underline{x}' = kH\underline{x}$, for any nonzero k, is equivalent to $\underline{x}' = H\underline{x}$, i.e. an arbitrary scale factor does not change the projective transformation. Consequently, H is a homogeneous matrix since, as in the case of the homogeneous representation of a point, any nonzero multiple of H is equivalent to H.

There are 8 independent elements in the homogeneous matrix H, and so it follows that there are 8 degrees of freedom in a projective transformation.

5.2.1 Transformation of lines

By the definition of a projective transformation we know that if the n points $\underline{x}_1, \underline{x}_2, \ldots, \underline{x}_n$ lie on a line $\underline{l}$, i.e. $\underline{l}^T\underline{x}_i = 0$, $i = 1, \ldots, n$, then the transformed points

$\underline{x}'_1, \underline{x}'_2, \ldots, \underline{x}'_n$, where $\underline{x}'_i = H\underline{x}_i$, $i = 1, \ldots, n$, lie on a line, say $\underline{l}'$. We claim that the line $\underline{l}'$ is given by $\underline{l}' = H^{-T}\underline{l}$. To verify this claim, consider

$$(\underline{l}')^T\underline{x}'_i = (H^{-T}\underline{l})^T(H\underline{x}_i) = \underline{l}^T H^{-1} H\underline{x}_i = \underline{l}^T\underline{x}_i = 0, \tag{5.9}$$

thereby proving that the transformed points $\underline{x}'_i$ all lie on the line $\underline{l}' = H^{-T}\underline{l}$.

5.2.2 The hierarchy of transformations

The most general form of a projective transformation is given in matrix form by

$$\underline{x}' = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} \underline{x}. \tag{5.10}$$

In this section, we consider special cases of the projective transformation, starting with the most restricted case, an isometry, and ending with the most general projective transformation.

Isometric transformations

An *isometric* transformation (or isometry) of the plane $\mathbb{R}^2$ preserves Euclidean distance (*iso* = same, *metric* = measure), e.g. a rotation. The most general isometry is represented by the matrix equation

$$\begin{bmatrix} x'_1 \\ x'_2 \\ 1 \end{bmatrix} = \begin{bmatrix} \epsilon\cos\theta & -\sin\theta & t_x \\ \epsilon\sin\theta & \cos\theta & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ 1 \end{bmatrix}. \tag{5.11}$$

where $\epsilon = \pm 1$. If $\epsilon = 1$, then the isometry is *orientation-preserving* and is a Euclidean transformation (a composition of a rotation and a translation). If $\epsilon = -1$ the isometry reverses orientation, for example, a composition of a reflection and a translation. Here we focus on Euclidean transformations, as they are predominant in the applications.

We can write a Euclidean transformation in block form as

$$\underline{x}' = H_E\underline{x} = \begin{bmatrix} R & \underline{t} \\ \underline{0}^T & 1 \end{bmatrix} \underline{x}, \tag{5.12}$$

where R is a 2×2 rotation matrix (an orthogonal matrix, satisfying $R^TR = RR^T = I$, with determinant 1), $\underline{t}$ is a translation 2-vector and $\underline{0}^T = \begin{bmatrix} 0 & 0 \end{bmatrix}$.

Degrees of freedom: a Euclidean transformation has 3 degrees of freedom: 1 for rotation and 2 for translation. Thus 3 parameters must be specified to define the

transformation. Accordingly, we can compute the transformation from 2 point correspondences.

Invariants: length, angles, area, parallel lines.

As an example, consider the Euclidean transformation where the matrix R is an anticlockwise rotation by $\frac{\pi}{3}$ radians and $\underline{t} = \begin{bmatrix} -\frac{1}{2} & 1 \end{bmatrix}^T$. Then

$$H_E = \begin{bmatrix} \cos(\frac{\pi}{3}) & -\sin(\frac{\pi}{3}) & -\frac{1}{2} \\ \sin(\frac{\pi}{3}) & \cos(\frac{\pi}{3}) & 1 \\ 0 & 0 & 1 \end{bmatrix}.$$

Figure 5.4 shows the result of applying this transformation to the unit square and unit circle. Use this figure to convince yourself of the invariants of the Euclidean transformation.

Remember that, when applying these transformations, input (x, y)-coordinates have to be converted to homogeneous coordinates by adding 1 as a third element and, in order to plot the result, one has to "de-homogenize" $\underline{x}'$ through division of its third element.

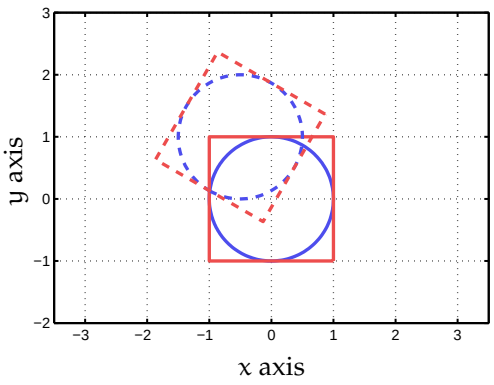


Figure 5.4: Example of an isometric transformation.

Similarity transformations

A *similarity* transformation (or a similarity) is an isometry together with an isotropic scaling (*isotropic* means equal in all directions). In the case of a Euclidean transformation composed with an isotropic scaling, the form of the similarity is given by

$$\begin{bmatrix} x_1' \\ x_2' \\ 1 \end{bmatrix} = \begin{bmatrix} s\cos\theta & -s\sin\theta & t_x \\ s\sin\theta & s\cos\theta & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ 1 \end{bmatrix}. \tag{5.13}$$

This can be written more concisely in block form as

$$\underline{x}' = H_S \underline{x} = \begin{bmatrix} sR & \underline{t} \\ \underline{0}^T & 1 \end{bmatrix} \underline{x}, \tag{5.14}$$

where s is a scalar that represents the scaling and the other symbols are the same as for an isometry.

Degrees of freedom: the similarity transformation has four degrees of freedom: the same three as the isometry plus one for the scaling. We can therefore compute a similarity from two point correspondences.

Invariants: ratio of two lengths, angles, ratio of areas, parallel lines.

For the similarity transformation with scaling factor $s = \frac{1}{2}$, and where the matrix R and the translation vector $\underline{t}$ are the same as the Euclidean transformation example, we have the transformation matrix H_S given by

$$H_S = \begin{bmatrix} \frac{1}{2}\cos(\frac{\pi}{3}) & -\frac{1}{2}\sin(\frac{\pi}{3}) & -\frac{1}{2} \\ \frac{1}{2}\sin(\frac{\pi}{3}) & \frac{1}{2}\cos(\frac{\pi}{3}) & 1 \\ 0 & 0 & 1 \end{bmatrix}.$$

Use Figure 5.5, an illustration of the similarity transformation given in this example, to convince yourself of the invariants.

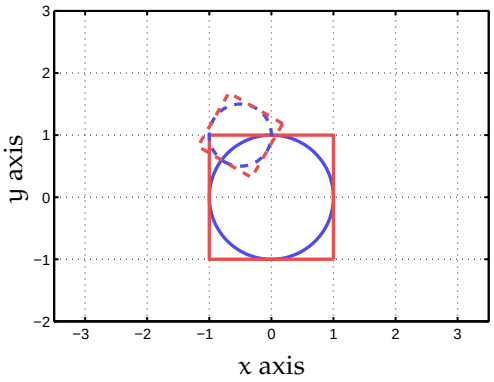


Figure 5.5: Example of a similarity transformation.

Affine transformations

An *affine* transformation (or an affinity) is a non-singular linear transformation followed by a translation. It is represented, in matrix form, by

$$\begin{bmatrix} x_1' \\ x_2' \\ 1 \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & t_x \\ a_{21} & a_{22} & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ 1 \end{bmatrix}, \tag{5.15}$$

or more compactly in block form, by

$$\underline{x}' = H_A \underline{x} = \begin{bmatrix} A & \underline{t} \\ \underline{0}^T & 1 \end{bmatrix} \underline{x}, \tag{5.16}$$

where A is a nonsingular 2×2 matrix, and the other symbols are as before.

Using the SVD factorization of the matrix A, we can develop a useful way of understanding the geometric effects of the linear component A as the composition of a scaling and rotations. Suppose the SVD of a given matrix A is $A = U\Sigma V^T$, where U and V are orthogonal matrices and

$$\Sigma = \begin{bmatrix} \sigma_1 & 0 \\ 0 & \sigma_2 \end{bmatrix}. \tag{5.17}$$

We can regroup this factorization as follows

$$A = U\Sigma V^T = U\left(V^T V\right)\Sigma V^T = \left(UV^T\right)V\Sigma V^T, \tag{5.18}$$

since V is an orthogonal matrix. With $P = UV^T$ we note that P is also orthogonal since $P^T P = \left(UV^T\right)^T UV^T = VU^T UV^T = VV^T = I$, so that multiplication with P is equivalent to a rotation. Suppose P represents a rotation by ϕ and V a rotation by θ, then

$$A = PV\Sigma V^T = R(\phi)R(-\theta)\Sigma R(\theta), \tag{5.19}$$

where $R(\phi)$ and $R(\theta)$ represent rotations by ϕ and θ, respectively. The matrix A is therefore a composition of a rotation (by θ); a non-isotropic scaling by σ_1 and σ_2 respectively in the rotated x and y directions; a rotation (by $-\theta$); and finally another rotation (by ϕ). The essence of an affinity is the scaling in orthogonal directions, oriented at a particular angle.

Degrees of freedom: the SVD argument above leads to the understanding that an affinity has two more degrees of freedom when compared to a similarity. These extra degrees of freedom are due to the non-isotropic scaling, making the total degrees of freedom for an affinity six. The two extra degrees are the angle θ, specifying the scaling direction, and the ratio of the scaling parameters $\frac{\sigma_1}{\sigma_2}$.

Invariants:

- ratio of areas
  Area is scaled by $\det A = \sigma_1\sigma_2$, i.e. the ratio of areas is preserved.

- parallel lines
  Two parallel lines intersect at $\begin{bmatrix} x_1 & x_2 & 0 \end{bmatrix}^T$ (an ideal point). Under an affinity this ideal point is mapped to

  $$\underline{x}' = \begin{bmatrix} A & \underline{t} \\ \underline{0}^T & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ 0 \end{bmatrix} = \begin{bmatrix} A\underline{y} \\ 0 \end{bmatrix},$$

  where $\underline{y} = \begin{bmatrix} x_1 & x_2 \end{bmatrix}^T$. Since the last element of $\underline{x}'$ is zero, it is also an ideal point. Note that the transformed parallel lines do not necessarily have the same direction as the original parallel lines.

- ratio of lengths of parallel segments

For the affinity transformation with matrix $A = \begin{bmatrix} 1 & 2 \\ -1 & \frac{1}{2} \end{bmatrix}$ and translation vector $\underline{t}$ the same as the previous two examples, we have the transformation matrix H_A given by

$$H_A = \begin{bmatrix} 1 & 2 & -\frac{1}{2} \\ -1 & \frac{1}{2} & 1 \\ 0 & 0 & 1 \end{bmatrix}.$$

Use Figure 5.6, an illustration of the affinity transformation given in this example, to convince yourself of the invariants of the affinity transformation.

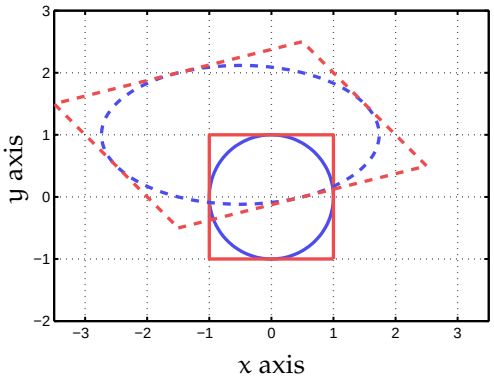


Figure 5.6: Example of an affine transformation.

Projective transformations

A *projective* transformation (or projectivity) as defined in (5.8) is a general non-singular linear transformation of homogeneous coordinates, and generalizes an affinity which is a non-singular linear transformation of inhomogeneous coordinates and a translation. A projectivity is given in block form as

$$\underline{x}' = H_P \underline{x} = \begin{bmatrix} A & \underline{t} \\ \underline{v}^T & v \end{bmatrix} \underline{x}, \tag{5.20}$$

where both $\underline{v}$ and v are not necessarily nonzero, and the other symbols are as before. Note that since v could be zero, we cannot always scale the matrix H_P in such a way that $v = 1$.

Degrees of freedom: the matrix H_A has nine elements, but since it is a homogeneous matrix, any nonzero multiple of H_A, say kH_A, is equivalent to H_A and we therefore only have eight degrees of freedom. We can compute the matrix H_A from a four point correspondence (with no three points being collinear).

Invariants: cross ratio of four points (ratio of ratios of lengths on a line).

Consider the projective transformation with matrix

$$H_P = \begin{bmatrix} -1 & 2 & 1 \\ 1 & -1 & \frac{1}{2} \\ -\frac{1}{2} & 2 & -3 \end{bmatrix}.$$

Use Figure 5.7, an illustration of the projective transformation given in this example, to convince yourself that the invariants listed for the previous transformations (for example parallel lines) are not invariants for this transformation.

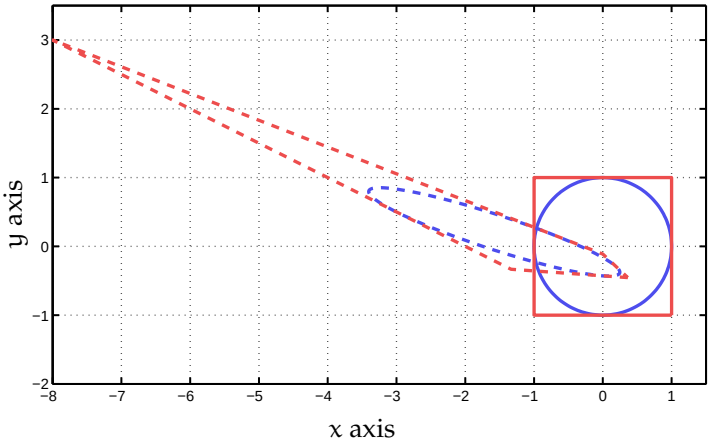


Figure 5.7: Example of a projective transformation.

If a projective transformation preserves parallel lines, it is an affinity. To prove this, suppose a projective transformation is such that parallel lines are mapped to parallel lines, i.e.

$$\begin{bmatrix} A & \underline{t} \\ \underline{v}^T & v \end{bmatrix} \begin{bmatrix} \underline{x} \\ 0 \end{bmatrix} = \begin{bmatrix} A\underline{x} \\ \underline{v}^T\underline{x} \end{bmatrix} = \begin{bmatrix} x_1' \\ x_2' \\ 0 \end{bmatrix}. \tag{5.21}$$

Then it must hold that $\underline{v}^T\underline{x} = 0$ for all $\underline{x}$, so that $\underline{v}$ must necessarily be equal to the zero vector. Since the matrix must be non-singular, it must then also hold that $v \neq 0$, so that the transformation matrix is of the form

$$\begin{bmatrix} A & \underline{t} \\ \underline{0}^T & c \end{bmatrix} \tag{5.22}$$

for a nonzero constant c. The transformation is therefore an affinity.

5.2.3 Decomposition of a projective transformation

A general projective transformation can be decomposed into a chain of transformations: a similarity, an affinity and a projectivity. Suppose H is a general

projective transformation. Then

$$H = H_S H_A H_P = \begin{bmatrix} sR & \underline{t} \\ \underline{0}^T & 1 \end{bmatrix} \begin{bmatrix} K & \underline{0} \\ \underline{0}^T & 1 \end{bmatrix} \begin{bmatrix} I & \underline{0} \\ \underline{v}^T & v \end{bmatrix} = \begin{bmatrix} A & v\underline{t} \\ \underline{v}^T & v \end{bmatrix}, \tag{5.23}$$

where A is a non-singular matrix given by $A = sRK + \underline{t}\,\underline{v}^T$, and K is an upper-triangular matrix normalized so that $\det K = 1$. This decomposition is valid provided $v \neq 0$, and is unique if $s > 0$.

Note that s, R and K are easily calculated given A. First we read v, $\underline{v}$ and $\underline{t}$ directly from H. Thus $sRK = A - \underline{t}\,\underline{v}^T$, its QR factorization gives the factors (R) and (sK), and s is the normalization factor so that $\det K = 1$.

Here the matrix H_S represents the essence of a similarity transformation, i.e. an isotropic scaling, a rotation and a translation; the matrix H_A represents the essence of an affine transformation, i.e. non-isotropic scaling in rotated directions; and H_P represents the essence of a projectivity, i.e. mapping ideal points to finite points since $\underline{v} \neq \underline{0}$.

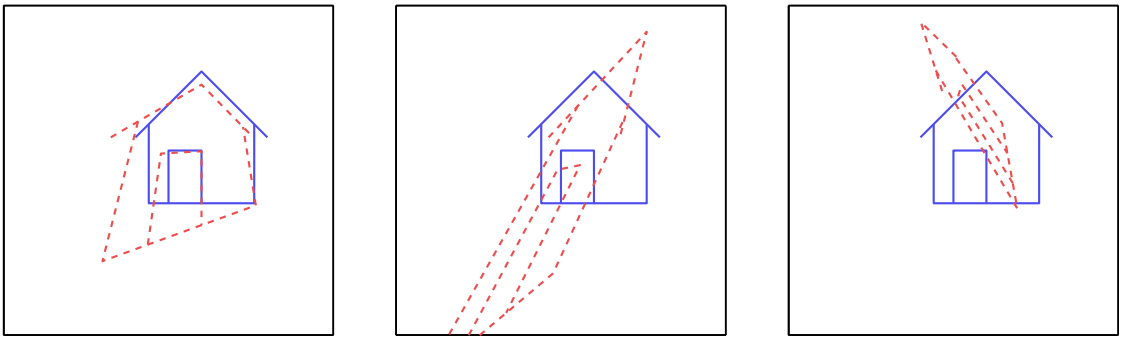
Consider for example the projective transformation with matrix

$$H = \begin{bmatrix} -0.0115 & -0.7406 & -0.5 \\ 0.4896 & 1.1821 & 1 \\ 0.2731 & 0.2491 & 1 \end{bmatrix}.$$

It can be decomposed as

$$H = \begin{bmatrix} \frac{1}{2}\cos(\frac{\pi}{3}) & -\frac{1}{2}\sin(\frac{\pi}{3}) & -0.5 \\ \frac{1}{2}\sin(\frac{\pi}{3}) & \frac{1}{2}\cos(\frac{\pi}{3}) & 1 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0.5 & 1 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0.2731 & 0.2491 & 1 \end{bmatrix},$$

and this chain of transformations is illustrated in Figure 5.8.



(a) H_P applied to an object (b) H_A applied to the result (c) H_S applied to the result

Figure 5.8: Applying a chain of transformations to the blue object shown.

5.3 Removing perspective distortion from a plane

Consider the *central projection* in Figure 5.9. Projection along rays through a common point (the centre of the projection) defines a mapping from one plane to another, say the world plane to the image plane. It maps lines to lines, and so is a projective transformation. If a coordinate system is defined in each of the planes, say world coordinates and image coordinates, and points are represented in homogeneous coordinates then the central projection mapping may be expressed as

$$\underline{x}' = H\underline{x}, \tag{5.24}$$

where H is a non-singular 3×3 matrix. This transformation is called a *perspective transformation* (or plane-to-plane homography) and is actually more restricted than a general projective transformation in that it only has six degrees of freedom if the coordinates are Euclidean.

Shape is distorted under such a perspective transformation but, since the image plane is related to the world plane via a projective transformation, we can undo this distortion. This is done by computing the inverse projective transformation and applying it to the image, resulting in an image where the objects have their correct geometric shape.

In order to compute this inverse transformation, we choose local inhomogeneous coordinates $\underline{x} = (x, y)$ and $\underline{x}' = (x', y')$, both measured directly from the world and the image plane respectively. Then, the projective transformation

$$\begin{bmatrix} x_1' \\ x_2' \\ x_3' \end{bmatrix} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}, \tag{5.25}$$

can be written in inhomogeneous form as

$$x' = \frac{x_1'}{x_3'} = \frac{h_{11}x + h_{12}y + h_{13}}{h_{31}x + h_{32}y + h_{33}}, \qquad y' = \frac{x_2'}{x_3'} = \frac{h_{21}x + h_{22}y + h_{23}}{h_{31}x + h_{32}y + h_{33}}. \tag{5.26}$$

Each point correspondence leads to two linear equations (linear in the unknown coefficients h_{ij}) of the form

$$\begin{aligned} x'(h_{31}x + h_{32}y + h_{33}) &= h_{11}x + h_{12}y + h_{13} \\ y'(h_{31}x + h_{32}y + h_{33}) &= h_{21}x + h_{22}y + h_{23} \end{aligned}$$

or equivalently,

$$\begin{bmatrix} x & y & 1 & 0 & 0 & 0 & -x'x & -x'y & -x' \\ 0 & 0 & 0 & x & y & 1 & -y'x & -y'y & -y' \end{bmatrix} \begin{bmatrix} h_{11} \\ h_{12} \\ \vdots \\ h_{33} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}. \tag{5.27}$$

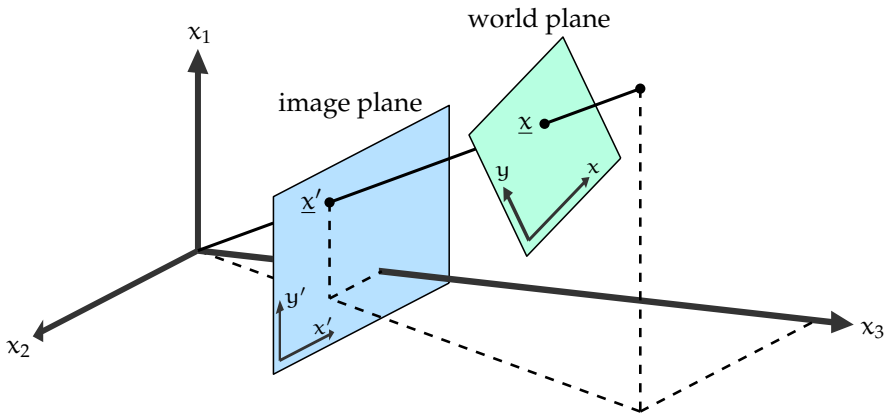


Figure 5.9: Transforming from a world plane to the image plane.

Since H has eight degrees of freedom and each point correspondence results in two linear equations, we need four point correspondences of the form $\underline{x} \leftrightarrow \underline{x}'$ to solve for H. Of course, if one has more than four point correspondences it is even better. The resulting overdetermined system for the null space can then be solved in a least-squares sense. Since measurement errors corrupt the system, the rank of the coefficient matrix will now be 9, but with a small 9th singular value. The 9th singular vector is therefore used as an approximation for the null space.

For four point correspondences, we get the system of equations $A\underline{h} = \underline{0}$ where A is an 8×9 matrix of the form

$$A = \begin{bmatrix} x_1 & y_1 & 1 & 0 & 0 & 0 & -x_1'x_1 & -x_1'y_1 & -x_1' \\ 0 & 0 & 0 & x_1 & y_1 & 1 & -y_1'x_1 & -y_1'y_1 & -y_1' \\ x_2 & y_2 & 1 & 0 & 0 & 0 & -x_2'x_2 & -x_2'y_2 & -x_2' \\ 0 & 0 & 0 & x_2 & y_2 & 1 & -y_2'x_2 & -y_2'y_2 & -y_2' \\ x_3 & y_3 & 1 & 0 & 0 & 0 & -x_3'x_3 & -x_3'y_3 & -x_3' \\ 0 & 0 & 0 & x_3 & y_3 & 1 & -y_3'x_3 & -y_3'y_3 & -y_3' \\ x_4 & y_4 & 1 & 0 & 0 & 0 & -x_4'x_4 & -x_4'y_4 & -x_4' \\ 0 & 0 & 0 & x_4 & y_4 & 1 & -y_4'x_4 & -y_4'y_4 & -y_4' \end{bmatrix}, \tag{5.28}$$

and $\underline{h} = \begin{bmatrix} h_{11} & h_{12} & \cdots & h_{33} \end{bmatrix}^T$. Choosing points so that A has full rank, i.e A is of rank 8, implies that no three points can be collinear.

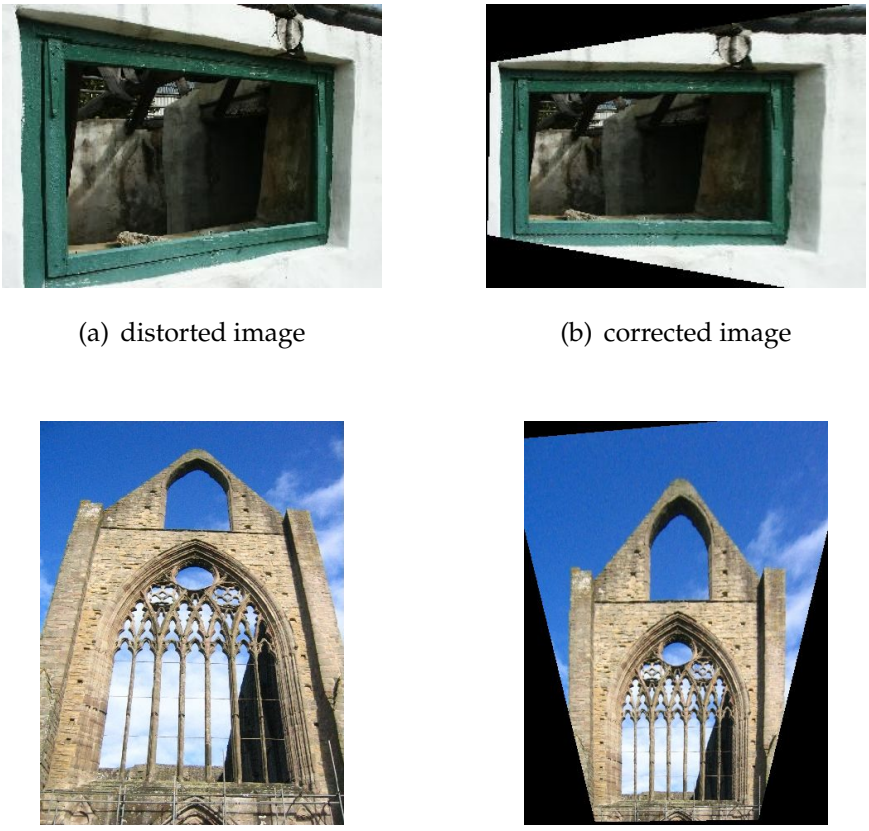
The vector $\underline{h}$ is then a vector in the null space of the matrix A, or any nonzero multiple of it. One way of obtaining $\underline{h}$ is to compute the SVD of the matrix A, i.e. $A = U\Sigma V^T$. Then the last column of V will be a basis for the null space of A, so that we can choose $\underline{h}$ as the last column of V.

Applying this to a perspective distorted image, we can correct the distortion. For example in Figure 5.10(a) we recognise that the four corners of the window are the corners of a rectangle in world coordinates. We then choose the points $\underline{x}_i = \begin{bmatrix} x_i & y_i \end{bmatrix}$, $i = 1, \ldots, 4$, as the four corners of the window and map them to the four corners of a rectangle to get the projective transformation H. Once we have

H, we apply its inverse to the entire distorted image (following the procedure in section 2.7) to obtain the rectified image shown in Figure 5.10(b).

A second example is shown in Figure 5.10(c) and (d).

Note that the computation of the matrix H does not require any knowledge of the camera's parameters (such as the focal length), or the orientation of the plane.



(a) distorted image (b) corrected image

(c) distorted image (d) corrected image

Figure 5.10: Removing perspective distortion with plane-to-plane homographies.

5.4 Recovering affine properties of images

In the previous section we corrected a perspectively distorted image up to a similarity. Suppose now that we are given less information and asked to correct the image up to an affinity (such that parallel lines are restored).

We may accomplish this task by identifying the image of the line at infinity (where parallel lines meet), and mapping it to infinity.

5.4.1 Affinities and the line at infinity

Under a projective transformation ideal points may be mapped to finite points, since

$$\begin{bmatrix} A & \underline{t} \\ \underline{v}^T & v \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ 0 \end{bmatrix} = \begin{bmatrix} A\underline{y} \\ \underline{v}^T\underline{y} \end{bmatrix}, \tag{5.29}$$

where $\underline{y} = \begin{bmatrix} x_1 & x_2 \end{bmatrix}^T$. If $\underline{v}^T\underline{y} \neq 0$, this point is finite. Consequently, a projective transformation may map the line at infinity $\underline{l}_\infty = \begin{bmatrix} 0 & 0 & 1 \end{bmatrix}^T$ to a finite line. We have already witnessed this phenomenon: parallel lines in the distorted images in Figure 5.10 intersect at finite points.

However, if the transformation is an affinity, say

$$H_A = \begin{bmatrix} A & \underline{t} \\ \underline{0}^T & 1 \end{bmatrix}, \tag{5.30}$$

then $\underline{l}_\infty$ is mapped to $\underline{l}_\infty$, and not a finite line. To prove this, we use the fact that if the line $\underline{l}$ is mapped to $\underline{l}'$ under the transformation H, then the line $\underline{l}'$ is given by $H^{-T}\underline{l}$ (refer to section 5.2.1). Then

$$\underline{l}' = H_A^{-T}\underline{l}_\infty = \begin{bmatrix} A^{-T} & \underline{0} \\ -\underline{t}^T A^{-T} & 1 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} = \underline{l}_\infty. \tag{5.31}$$

The converse is also true, i.e. an affine transformation is the most general linear transformation that maps $\underline{l}_\infty$ to $\underline{l}_\infty$. This can be justified as follows. If an ideal point, say $\underline{x} = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix}^T$, is mapped to another ideal point, then the first element of $\underline{v}$ must be zero. Similarly, the second element of $\underline{v}$ must be zero. Therefore the transformation is an affinity.

Note that the line $\underline{l}_\infty$ is not fixed point-wise under an affinity. For example, the affinity

$$H_A = \begin{bmatrix} 1 & 2 & -\frac{1}{2} \\ -1 & \frac{1}{2} & 1 \\ 0 & 0 & 1 \end{bmatrix},$$

maps the ideal point $\begin{bmatrix} 1 & 1 & 0 \end{bmatrix}^T$ to another ideal point, namely

$$\begin{bmatrix} 1 & 2 & -\frac{1}{2} \\ -1 & \frac{1}{2} & 1 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 3 \\ \frac{1}{2} \\ 0 \end{bmatrix}.$$

We will use this property of affinities to recover affine properties by identifying the line at infinity $\underline{l}_\infty$.

5.4.2 Using the line at infinity to recover affine properties

Once the image of the line at infinity has been identified, say $\underline{l} = \begin{bmatrix} \ell_1 & \ell_2 & \ell_3 \end{bmatrix}^T$, it is possible to construct a projectivity that will project it back to $\underline{l}_\infty$. Since an affinity maps $\underline{l}_\infty$ to itself, this transformation is only defined up to an affinity.

It is straightforward to verify that, if $\ell_3 \neq 0$, the projective map

$$H = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ \ell_1 & \ell_2 & \ell_3 \end{bmatrix} \tag{5.32}$$

maps $\underline{l}$ to $\underline{l}_\infty$, since

$$H^{-T}\underline{l} = \begin{bmatrix} 1 & 0 & -\frac{\ell_1}{\ell_3} \\ 0 & 1 & -\frac{\ell_2}{\ell_3} \\ 0 & 0 & \frac{1}{\ell_3} \end{bmatrix} \begin{bmatrix} \ell_1 \\ \ell_2 \\ \ell_3 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}. \tag{5.33}$$

This prescription does not work if $\ell_3 = 0$. We leave it as an exercise to verify that if $\ell_3 = 0$ and $\ell_1 \neq 0$ we may choose

$$H = \begin{bmatrix} 0 & 1 & -1 \\ 0 & 0 & 1 \\ \ell_1 & \ell_2 & 0 \end{bmatrix}, \tag{5.34}$$

or if $\ell_3 = 0$ and $\ell_1 = 0$ (implying that $\ell_2 \neq 0$),

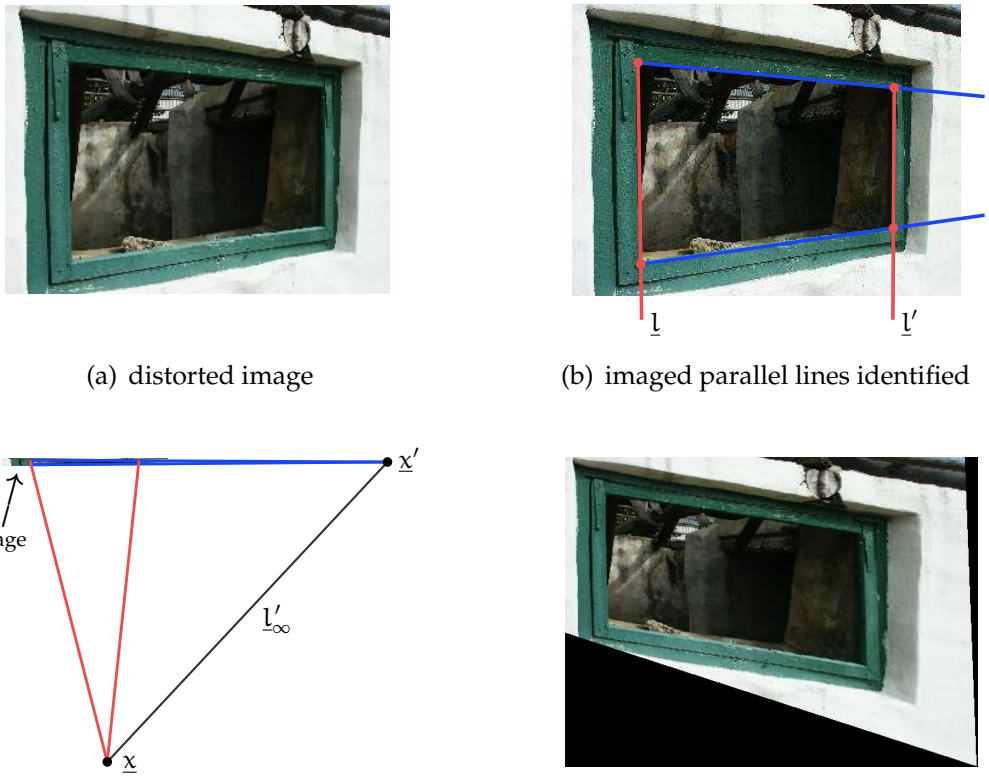
$$H = \begin{bmatrix} 0 & 0 & 1 \\ 1 & 0 & -1 \\ 0 & \ell_2 & 0 \end{bmatrix}. \tag{5.35}$$

Once a mapping H is specified, we may transform the image accordingly (following the procedure discussed in section 2.7).

It only remains to identify the image of the line at infinity. Perhaps the easiest way to do this is to identify two lines that were parallel on the original plane. Their point of intersection provides an image of a point on the line at infinity. If we can find a second such point, the image of the line at infinity is the line through these two points, and we know how to find the expression for a line through two points. We illustrate these ideas by means of an example.

For the image in Figure 5.11(a), we notice that the frame of the window should be rectangular, so that the top and bottom of the window frame should be parallel and the two sides should be parallel. We identify the four corners of the window frame, and then use these four points to calculate the lines that run with the sides of the frame, labelled as $\underline{l}$ and $\underline{l}'$ in Figure 5.11(b). We know that these lines are the images of parallel lines, so that their intersection point, labelled as $\underline{x}$ in Figure 5.11(c), is the image of an ideal point. Similarly, we calculate the intersection

point of the two lines running with the top and bottom of the window frame, to get $\underline{x}'$. Now that we have two points, that are each the image of ideal point, we can calculate the imaged line at infinity as $\underline{l}'_\infty = \underline{x} \times \underline{x}'$. From the theory above we construct the matrix H that maps this imaged line at infinity to the canonical form of the line at infinity, i.e. $\underline{l}_\infty = \begin{bmatrix} 0 & 0 & 1 \end{bmatrix}^T$, and apply the transformation H to the entire image to get the image in Figure 5.11(d), where the sides of the window frame are parallel.



(a) distorted image

(b) imaged parallel lines identified

(c) intersection points of parallel lines and imaged $\underline{l}'_\infty$ identified

(d) image corrected up to an affinity: parallel lines are parallel

Figure 5.11: Recovering affine properties from a distorted image using $\underline{l}_\infty$.

5.4.3 Cross ratios and the vanishing point for a length ratio

Let us now consider the projective geometry of a line, $\mathbb{P}^1$, which proceeds much the same as that of a plane. A point $\underline{x}$ on a line is represented by homogeneous coordinates $\begin{bmatrix} x_1 & x_2 \end{bmatrix}^T$, and a point for which $x_2 = 0$ is an ideal point of the line. A projective transformation of a line is represented by a non-singular 2×2 homogeneous matrix H_2, so that $\underline{x}' = H_2\underline{x}$, and has three degrees of freedom.

The *cross ratio* is the basic projective invariant of $\mathbb{P}^1$. Given four points on a line,

say $\underline{a}$, $\underline{b}$, $\underline{c}$ and $\underline{d}$, then the cross ratio is defined as

$$\text{Cross}(\underline{a}, \underline{b}, \underline{c}, \underline{d}) = \frac{|\underline{a}\,\underline{b}|\;|\underline{c}\,\underline{d}|}{|\underline{a}\,\underline{c}|\;|\underline{b}\,\underline{d}|}, \tag{5.36}$$

where

$$|\underline{a}\,\underline{b}| = \det \begin{bmatrix} a_1 & b_1 \\ a_2 & b_2 \end{bmatrix}, \tag{5.37}$$

and similarly for the other points.

Cross ratios have the following important properties:

- The value of the cross ratio does not depend on which homogeneous representative of a point is used, since the scale cancels between the numerator and denominator.
- The definition of the cross ratio is still valid if one of the points is an ideal point.
- The value of the cross ratio is invariant under any projective transformation of the line (see Figure 5.12) since if $\underline{x}' = H\underline{x}$, then

$$\begin{aligned}
\text{Cross}(\underline{a}', \underline{b}', \underline{c}', \underline{d}') &= \frac{|\underline{a}'\,\underline{b}'|\;|\underline{c}'\,\underline{d}'|}{|\underline{a}'\,\underline{c}'|\;|\underline{b}'\,\underline{d}'|} \\
&= \frac{|(\lambda_1 H\underline{a})\;(\lambda_2 H\underline{b})|\;|(\lambda_3 H\underline{c})\;(\lambda_4 H\underline{d})|}{|(\lambda_1 H\underline{a})\;(\lambda_3 H\underline{c})|\;|(\lambda_2 H\underline{b})\;(\lambda_4 H\underline{d})|} \\
&= \frac{(\lambda_1\lambda_2\,|H|\,|\underline{a}\,\underline{b}|)\;(\lambda_3\lambda_4\,|H|\,|\underline{c}\,\underline{d}|)}{(\lambda_1\lambda_3\,|H|\,|\underline{a}\,\underline{c}|)\;(\lambda_2\lambda_4\,|H|\,|\underline{b}\,\underline{d}|)} \\
&= \frac{|\underline{a}\,\underline{b}|\;|\underline{c}\,\underline{d}|}{|\underline{a}\,\underline{c}|\;|\underline{b}\,\underline{d}|} \\
&= \text{Cross}(\underline{a}, \underline{b}, \underline{c}, \underline{d}).
\end{aligned}$$

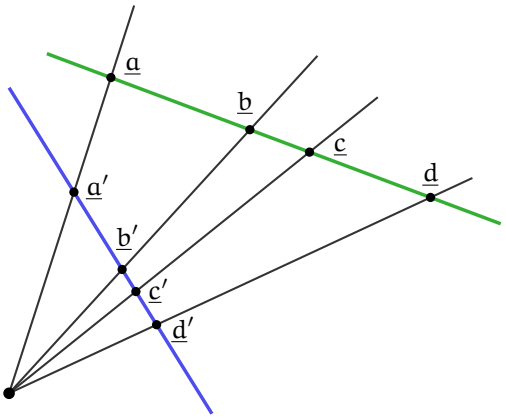


Figure 5.12: Invariance of the cross ratio under a projective transformation.

Suppose $\underline{a}$, $\underline{b}$ and $\underline{c}$ are collinear points in the world reference frame, i.e. they fall on a line in $\mathbb{R}^3$, and suppose further that the ratio of the two intervals defined by these three points is known. If the images of these points, $\underline{a}'$, $\underline{b}'$ and $\underline{c}'$ can be identified, the projective map between the two lines can be calculated. Using this map, the point at infinity on the line is mapped to its image. More specifically, if the length ratio is given by $\frac{d(\underline{a},\underline{b})}{d(\underline{b},\underline{c})} = \frac{m}{n}$ (where $d(\underline{a},\underline{b})$ is the Euclidean distance between $\underline{a}$ and $\underline{b}$), one calculates the image of the point at infinity using the following procedure.

1. Measure the distance ratio in the image, $\frac{d(\underline{a}',\underline{b}')}{d(\underline{b}',\underline{c}')} = \frac{m'}{n'}$.

2. Represent the points $\underline{a}$, $\underline{b}$ and $\underline{c}$ as 0, m and $m+n$ in a coordinate frame on the line going through $\underline{a}$, $\underline{b}$ and $\underline{c}$. These points may be represented by homogeneous 2-vectors $\begin{bmatrix}0 & 1\end{bmatrix}^T$, $\begin{bmatrix}m & 1\end{bmatrix}^T$ and $\begin{bmatrix}m+n & 1\end{bmatrix}^T$. Similarly, $\underline{a}'$, $\underline{b}'$ and $\underline{c}'$ can be represented by the homogeneous 2-vectors $\begin{bmatrix}0 & 1\end{bmatrix}^T$, $\begin{bmatrix}m' & 1\end{bmatrix}^T$ and $\begin{bmatrix}m'+n' & 1\end{bmatrix}^T$.

3. Relative to these coordinate frames, compute the 1D projective transformation H_2 that maps $\underline{a}$ to $\underline{a}'$, $\underline{b}$ to $\underline{b}'$, and $\underline{c}$ to $\underline{c}'$.

4. Calculate the image of the ideal point $\begin{bmatrix}1 & 0\end{bmatrix}^T$ under H_2. It is the image of the vanishing point on the line through $\underline{a}'$, $\underline{b}'$ and $\underline{c}'$.

How would you find the image of a point at infinity using the cross ratio, assuming you know the world coordinates of three collinear points, as well as their images, as above?

Chapter 6

Robust parameter estimation

We now deviate slightly from the theory of projective geometry and, before we apply this theory to derive a camera model, discuss a very useful technique for estimating model parameters form data corrupted by outliers.

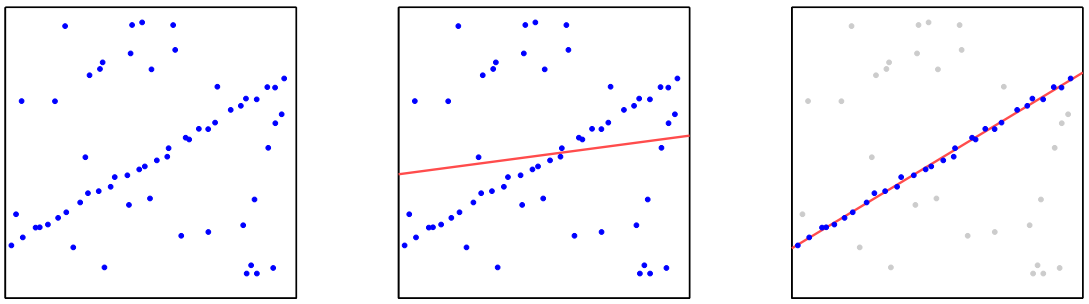
We saw in section 5.3 that it is possible to calculate a plane-to-plane homography from at least 4 point correspondences. It would be useful if we could obtain such point correspondences automatically by means of, say, feature detection and matching. A technique such as SIFT or SURF would be ideal but, as we've seen in Chapter 3, would also typically return many incorrect matches that may complicate matters. The so-called RANSAC algorithm is remarkably effective at distinguishing between correct and incorrect matches. Once those correct matches have been identified, a least squares fit can be performed in order to find a homography.

6.1 The problem with outliers

The problem at hand is to fit some model, e.g. by estimating its parameters, to a set of data points. Let us assume that this data consists of *inliers* (data that is explained well by a single model) and *outliers* (data that do not fit a model well). We are given only the data, and we would like to be able to determine which points are inliers and which are outliers, so that the model can be estimated using only the inliers.

Consider the following simple example. Suppose we want to fit a straight line to the data points in Figure 6.1(a). In this case our model is a straight line having two parameters (a slope and a y-intercept) and, clearly, the data is corrupted by outliers. A rather naive approach for fitting a straight line to this data would be to calculate the least squares solution. This line is shown in (b), and we observe that the outliers have a decidedly unwanted effect on our model.

If we can somehow identify and ignore the outliers, and perform least squares on only the set of inliers, we would be able to obtain a line such as the one shown in Figure 6.1(c).



(a) the given data (b) least squares line (c) line through the inliers

Figure 6.1: Fitting a straight line to data corrupted by outliers.

6.2 RANSAC

RANSAC[1], short for random sample consensus, is an iterative method for estimating the parameters of a mathematical model from a set of observed data. We assume here, as before, that the observed data consists of inliers and outliers. In addition, the data can be subject to noise.

We demonstrate the idea behind RANSAC by returning to our problem of fitting a straight line through points such as those in Figure 6.1(a):

1. repeat many times:
   choose two of the data points at random (this is our random sample);
   fit a straight line through the sample (our hypothesis);
   count the number of data points that fit the hypothesis (our consensus set)
2. pick the line that resulted in the largest consensus set
3. fit a least squares line through the consensus set

If samples are randomly drawn from the data then, eventually, two points that both happen to be inliers will be picked and the hypothesis will be close to the straight line that we are after. The consensus set will consist of many points (most of the inliers) and has a good chance of ultimately being selected as the set of inliers. Note that if we can assume that there is no structure in the outliers (i.e. there is not a single line that explains the outliers well) then RANSAC should work even if the number of outliers exceeds the number of inliers.

[1]M. Fischler & R. Bolles, *Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography*, Communications of the ACM, 24(6):381–395, 1981.

In general we may formulate the RANSAC algorithm as follows.

1. Repeat many times:
   1.1 randomly choose a subset of the data, the minimum required to fit a model;
   1.2 fit a model to this sample;
   1.3 form a consensus set by checking which other data points agree with the model.
2. Pick the model that resulted in the largest consensus set.
3. Fit a model to all the data in this consensus set (typically in a least squares sense).

There are various ways in which to decide exactly how many times to repeat the first part of the algorithm. One fairly common approach is to compute the number of iterations as

$$k = \frac{\log(1-p)}{\log(1-w^n)}, \tag{6.1}$$

with p the probability that the algorithm will pick a good subset (one consisting of only inliers), w the probability of choosing an inlier when a single point is drawn, and n the size of the sample.

6.3 Estimating a homography from feature matches

Suppose we have feature matches $(x_i, y_i) \leftrightarrow (x'_i, y'_i)$ between two images determined, for example, by means of SIFT. Suppose it is further known that (many of) these features are in fact located on some plane. Here we must be aware of the fact that SIFT may return many incorrect matches. See for example Figure 6.2(b).

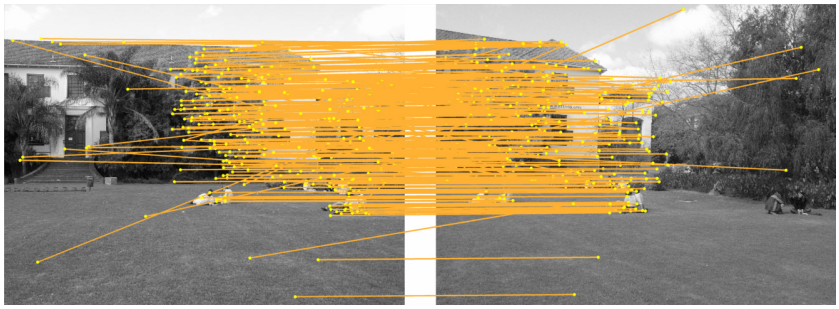
We are looking for a homography H such that

$$\begin{bmatrix} x'_i \\ y'_i \\ 1 \end{bmatrix} \approx H \begin{bmatrix} x_i \\ y_i \\ 1 \end{bmatrix} \tag{6.2}$$

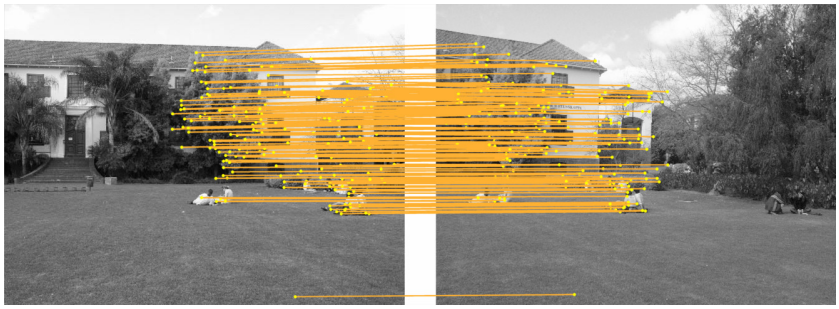
for all correct matches $(x_i, y_i) \leftrightarrow (x'_i, y'_i)$. Once more we face a problem because we do not yet know which matches are correct, and a naive least squares homography estimation, using all the available matches, would not be very useful.

Luckily, RANSAC is perfectly suited for this task. In this case the data is the set of SIFT correspondences and our model is the homography. Recall from section 5.3 that a homography can be calculated from at least 4 point correspondences.

(a) input images A and B



(b) putative feature matches as determined by SIFT



(c) largest set of inlier matches found by RANSAC

(d) images stitched with homography calculated from inliers

Figure 6.2: Fitting a homography with RANSAC using SIFT feature correspondences. Note that RANSAC is used to first remove outliers from the feature matches. One of the images is then transformed with the homography and placed on top of the other.

At every RANSAC iteration we pick 4 matches at random, and calculate a homography H from them (refer to section 5.3). We then map all points (x_i, y_i) with this H, and compare with (x'_i, y'_i). Those that are close enough (within some threshold distance) form the consensus set. Once a largest consensus set has been identified, such as the one shown in Figure 6.2(c), a homography can be re-estimated using all of those inlier matches.

In employing this technique for estimating a homography between two images, we make a crucial assumption about the structure of the observed scene. The homography we get is a mapping between two planes and, therefore, we have to assume that many of the detected features in the scene are in fact coplanar. This assumption is not particularly valid for the example in Figure 6.2, and we get slight misalignment of the two images.

Chapter 7

The pinhole camera model

Here we develop a basic camera model. The hardest part of this model is keeping track of the different coordinate systems. Let us therefore summarize them at the outset. First we'll define a camera coordinate system. This is centred at the focus of the camera with its X and Y axes aligned parallel to, and the Z axis perpendicular to, the image plane. The image plane is also provided with a coordinate system to record the position of features on the image. In practice of course, positions will be measured in pixel coordinates, so ultimately we'll have to make provision to measure in pixel coordinates. The object or scene to be captured is described in terms of a world coordinate system. It is therefore often convenient to fix the world coordinate system to the object or scene.

This chapter is about deriving the relationships between these coordinate systems. Written in homogeneous coordinates it is a linear relationship expressed in terms of a matrix called the camera matrix.

7.1 Central projection in homogeneous coordinates

Here we consider the central projection of a point $\underline{X} = \begin{bmatrix} X & Y & Z \end{bmatrix}^T$ in the camera coordinate system, with origin at the camera centre $\underline{C}$, onto the image plane. The image plane is located at $Z = f$ in the camera coordinate system where f is known as the focal length of the camera. The point where the Z axis pierces the image plane is known as the principal point and the Z axis as the principal axis. The origin of the image coordinate system is chosen, for now, as the principal point and its x- and y axes are aligned with the X and Y axes of the camera coordinate system. All of this is illustrated in Figure 7.1.

If a point $\underline{X} \in \mathbb{R}^3$ has coordinates $\begin{bmatrix} X & Y & Z \end{bmatrix}^T$ relative to the camera coordinate system, $\underline{X}$ projects onto the point $\underline{x}$ on the image plane, with $\underline{C}$ the centre of the projection, as in Figure 7.1. Using homogeneous coordinates this projection is

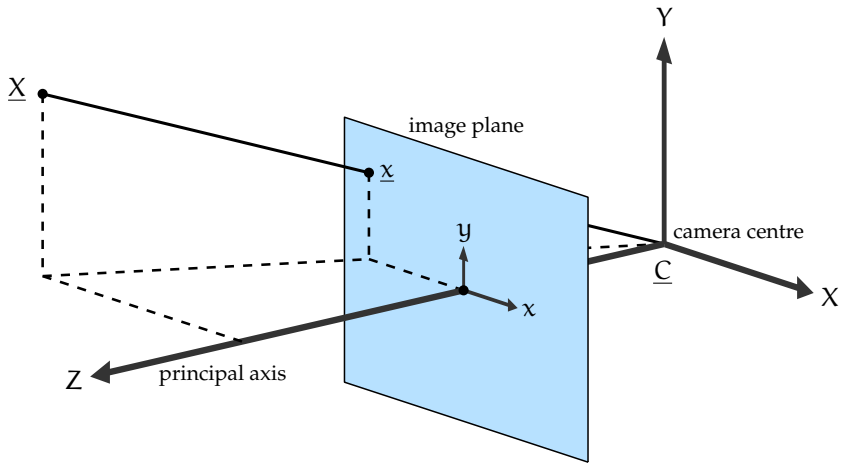


Figure 7.1: Illustrating basic pinhole camera geometry.

described by a matrix P. We'll variously refer to this matrix as the camera matrix or the projection matrix, depending on which aspect we wish to emphasize.

Using similar triangles, it follows immediately that $\underline{x} = \begin{bmatrix} f\frac{X}{Z} & f\frac{Y}{Z} \end{bmatrix}^T$ (in Euclidean coordinates). This is a nonlinear map that becomes linear if homogeneous coordinates are used.

Realising that $\begin{bmatrix} f\frac{X}{Z} & f\frac{Y}{Z} \end{bmatrix}^T$ can be written as the homogeneous vector $\begin{bmatrix} fX & fY & Z \end{bmatrix}^T$, the map from homogeneous camera coordinates to homogeneous image coordinates is given by

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}. \tag{7.1}$$

The matrix in this expression can be written as $\mathrm{diag}(f, f, 1)\,[I \mid \underline{0}]$ where

$$\mathrm{diag}(f, f, 1) = \begin{bmatrix} f & 0 & 0 \\ 0 & f & 0 \\ 0 & 0 & 1 \end{bmatrix}, \tag{7.2}$$

i.e. the diagonal matrix with $(f, f, 1)$ on the diagonal, and

$$[I \mid \underline{0}] = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}. \tag{7.3}$$

Using this notation, we can describe the central projection from $\underline{X}$ to $\underline{x}$ as

$$\underline{x} = P\underline{X} \tag{7.4}$$

where P is the 3×4 homogeneous *camera projection matrix*. This defines the camera matrix for the central projection as

$$P = \mathrm{diag}(f, f, 1)\,[I \mid \underline{0}]\,. \tag{7.5}$$

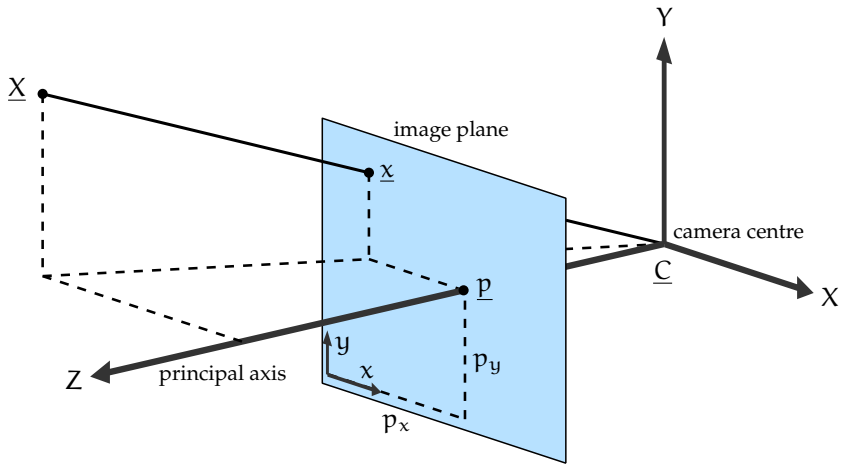


Figure 7.2: Illustrating pinhole camera geometry with offset image coordinates.

The camera matrix derived above assumes that the origin of the image coordinate system is at the principal point $\underline{p}$. However, this is not usually the case in practice. If the coordinates of the principal point $\underline{p}$ are (p_x, p_y) in the image coordinate system, as depicted in Figure 7.2, then the mapping of $\underline{X}$ to $\underline{x}$ is given by

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} \quad \text{is mapped to} \quad \begin{bmatrix} f\frac{X}{Z} + p_x \\ f\frac{Y}{Z} + p_y \end{bmatrix}, \tag{7.6}$$

or equivalently in homogeneous coordinates by

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} fX + Zp_x \\ fY + Zp_y \\ Z \end{bmatrix} = \begin{bmatrix} f & 0 & p_x & 0 \\ 0 & f & p_y & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}. \tag{7.7}$$

Introducing the *camera calibration matrix*

$$K = \begin{bmatrix} f & 0 & p_x \\ 0 & f & p_y \\ 0 & 0 & 1 \end{bmatrix}, \tag{7.8}$$

the camera matrix P is given by

$$P = K\,[I \mid \underline{0}]\,. \tag{7.9}$$

Emphasizing the fact that we are projecting features described in terms of the camera coordinate system, we rewrite the projection as

$$\underline{x} = K\,[I \mid \underline{0}]\,\underline{X}_{\text{cam}}. \tag{7.10}$$

The next step is to introduce the world coordinate system and relate it to the camera coordinate system.

7.2 The world coordinate system

In general, 3D objects are described in terms of coordinate systems fixed to the objects as shown in Figure 7.3. In homogeneous coordinates it is given by

$$\underline{X} = \begin{bmatrix} X & Y & Z & 1 \end{bmatrix}^T. \tag{7.11}$$

Since we already know how to project a feature in the camera coordinate system onto the image coordinate system, we only need to relate the world– and camera coordinate systems, i.e. $\underline{X}$ and $\underline{X}_{\text{cam}}$. Since the two coordinate systems are related by a rotation and a translation, as is clear from Figure 7.3, we may write

$$\widetilde{\underline{X}}_{\text{cam}} = R\left(\widetilde{\underline{X}} - \widetilde{\underline{C}}\right) = R\widetilde{\underline{X}} + \underline{t}, \tag{7.12}$$

where the tilde denotes Euclidean coordinates, so for example

$$\underline{X} = \begin{bmatrix} \widetilde{\underline{X}}^T & 1 \end{bmatrix}^T. \tag{7.13}$$

The Euclidean vector $\widetilde{\underline{C}}$ in (7.12) is the coordinates of the camera centre in the world coordinate system, and R is a 3×3 rotation matrix describing the rotation of the world coordinate system relative to the camera coordinate system. Also note that $\widetilde{\underline{X}}_{\text{cam}} = \underline{0}$ if $\widetilde{\underline{X}} = \widetilde{\underline{C}}$, i.e. the camera coordinate is zero at the camera centre, as expected.

If we use $\underline{X}_{\text{cam}}$ and $\underline{X}$ to denote the homogeneous representations of $\widetilde{\underline{X}}_{\text{cam}}$ and $\widetilde{\underline{X}}$ respectively, then we have

$$\underline{X}_{\text{cam}} = \begin{bmatrix} R & -R\widetilde{\underline{C}} \\ \underline{0}^T & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} = \begin{bmatrix} R & -R\widetilde{\underline{C}} \\ \underline{0}^T & 1 \end{bmatrix} \underline{X}. \tag{7.14}$$

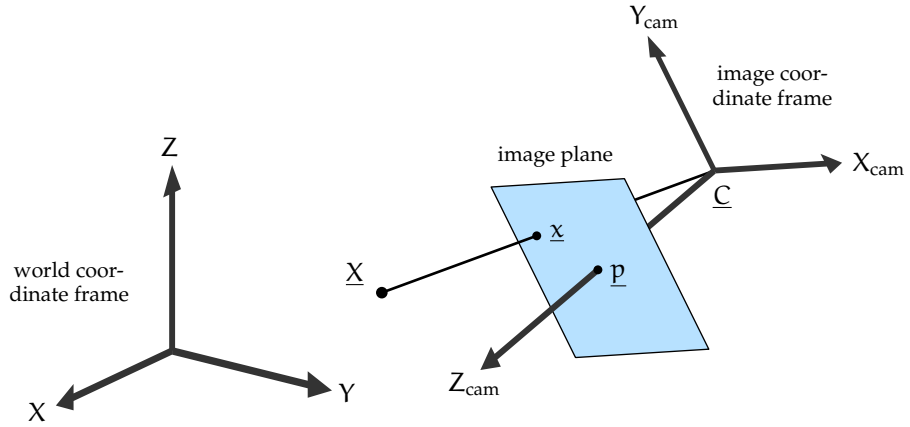


Figure 7.3: Pinhole camera geometry in a general world coordinate system.

Combining this with (7.10), we get

$$\underline{x} = KR\left[I \mid -\widetilde{\underline{C}}\right]\underline{X},$$

where $\underline{X}$ is now given in the world coordinate system.

Note that all the parameters that refer to the specific type of camera are contained in K; these parameters are referred to as the intrinsic parameters. R and $\widetilde{\underline{C}}$ describe the external orientation of the world coordinate system to the camera coordinate system and are therefore referred to as the extrinsic parameters.

7.3 The general camera calibration matrix K

In the models above we assume that the image coordinate frame is Euclidean with equal scales in both axial directions, which is not always true. In the case of CCD cameras there is the additional possibility of having rectangular pixels. In particular, if the number of pixels per unit distance in image coordinates are m_x and m_y in the x and y directions, respectively, then the calibration matrix becomes

$$K = \begin{bmatrix} m_x & 0 & 0 \\ 0 & m_y & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} f & 0 & p_x \\ 0 & f & p_y \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} \alpha_x & 0 & x_0 \\ 0 & \alpha_y & y_0 \\ 0 & 0 & 1 \end{bmatrix} \tag{7.15}$$

where $\alpha_x = fm_x$ and $\alpha_y = fm_y$ represent the focal length of the camera in terms of pixel coordinates in the x and y directions, respectively. Similarly, (x_0, y_0) is the principal point in terms of pixel coordinates with $x_0 = m_x p_x$ and $y_0 = m_y p_y$.

For added generality, we use a calibration matrix of the form

$$K = \begin{bmatrix} \alpha_x & s & x_0 \\ 0 & \alpha_y & y_0 \\ 0 & 0 & 1 \end{bmatrix}, \tag{7.16}$$

where the added parameter s is referred to as the *skew* parameter. The skew parameter will be zero for most normal cameras, but has some important applications.

With this calibration matrix K, we call the camera

$$P = KR\left[I \mid -\widetilde{\underline{C}}\right] \tag{7.17}$$

a *finite projective camera*.

We note that the general pinhole camera has eleven degrees of freedom: 5 for the calibration matrix K (the elements α_x, α_y, x_0, y_0 and s), 3 for the rotation matrix R and 3 for the camera centre $\widetilde{\underline{C}}$. This is the same number of degrees of freedom for a 3×4 matrix, defined up to a scale.

Note that the 3×3 submatrix $K R$ is non-singular ($\det KR \neq 0$). Conversely, any 3×4 matrix for which the left hand 3×3 submatrix is non-singular, is the camera matrix of some finite projective camera, because P can then be decomposed as $P = KR\left[I \mid -\widetilde{\underline{C}}\right]$, using the QR matrix factorization (more on this in section 8.2).

The final step in this hierarchy of projective cameras is to remove the condition of non-singularity on the left hand 3×3 submatrix. A *general projective* camera is one represented by an arbitrary homogeneous 3×4 matrix of rank 3. It has eleven degrees of freedom. We need a rank 3 matrix, since if the rank is 1 or 2, the range of the projective transformation is a line or point and not the whole plane, i.e. not a 2D image.

7.4 The camera projection matrix P

We will soon discover how it is possible to retrieve the camera matrix P through a process termed calibration. Here we take a closer look at the anatomy of P, and ask: how do we extract information about the camera (for example its focal point) from a given camera matrix?

Since P is a 3×4 matrix, we can write

$$P = \begin{bmatrix} p_{11} & p_{12} & p_{13} & p_{14} \\ p_{21} & p_{22} & p_{23} & p_{24} \\ p_{31} & p_{32} & p_{33} & p_{34} \end{bmatrix} = \begin{bmatrix} \underline{p}_1 & \underline{p}_2 & \underline{p}_3 & \underline{p}_4 \end{bmatrix} = \left[M \mid \underline{p}_4 \right], \tag{7.18}$$

where $\underline{p}_i$, $i = 1, 2, 3, 4$, are the columns of P and $M = \begin{bmatrix} \underline{p}_1 & \underline{p}_2 & \underline{p}_3 \end{bmatrix}$.

7.4.1 The null space of P

It follows quite straightforwardly that $\text{rank} P = 3$, implying that the dimension of the null space of P equals 1. Since the null space of P is the set of all vectors $\underline{y}$ such that

$$P\underline{y} = \underline{0}, \tag{7.19}$$

and its dimension is 1, it follows that the null space of P consists of a single homogeneous vector. This has special significance since it turns out that this vector is the camera centre. To show this, suppose that $\underline{a}$ is an arbitrary point in 3-space, and $\underline{y}$ is in the null space of P. Then the line in 3-space through $\underline{a}$ and $\underline{y}$ is parameterized (in homogeneous coordinates) as

$$\underline{X}(\lambda) = \lambda \underline{a} + (1 - \lambda)\underline{y}. \tag{7.20}$$

Each point $\underline{X}(\lambda)$ on this line maps to

$$\underline{x}(\lambda) = P\left(\underline{X}(\lambda)\right) = P\left(\lambda \underline{a} + (1 - \lambda)\underline{y}\right) = \lambda P\underline{a} + (1 - \lambda)P\underline{y} = \lambda P\underline{a}, \tag{7.21}$$

which is the same as $P\underline{a}$ in homogeneous coordinates. Thus the entire line through $\underline{y}$ and $\underline{a}$ maps to a single point $P\underline{a}$. Therefore $\underline{X}(\lambda)$ is a ray through the *camera centre* $\underline{C}$. Since this holds for any choice of $\underline{a}$, it follows that $\underline{y} = \underline{C}$ which is the camera centre. Figure 7.4 clarifies.

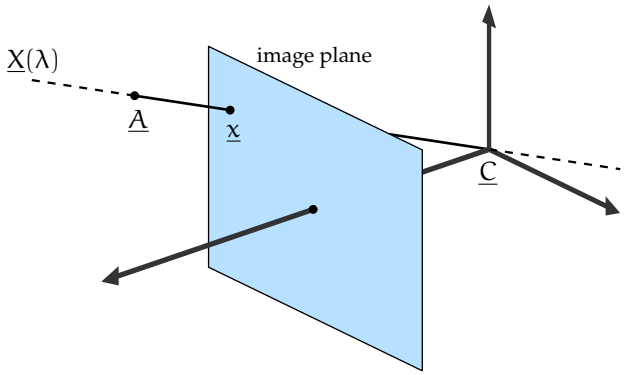


Figure 7.4: The ray through $\underline{A}$ and the camera centre.

It is not surprising that a vector in the null space of P is the camera centre of P, since the image point $\begin{bmatrix}0 & 0 & 0\end{bmatrix}^T = P\underline{C}$ is not defined, and the camera centre is the unique point for which the image is not defined.

For finite cameras one can calculate the null space of P directly. Moreover, since the homogeneous representation of the camera centre is $\underline{C} = \begin{bmatrix}\widetilde{\underline{C}}^T & 1\end{bmatrix}^T$ it follows that

$$P\underline{C} = KR\left[I \mid -\widetilde{\underline{C}}\right]\begin{bmatrix}\widetilde{\underline{C}}\\ 1\end{bmatrix} = \underline{0}, \tag{7.22}$$

which shows that the camera center in homogeneous coordinates, is indeed the null space of P. For finite cameras we also have that M is non-singular, and so M^{-1} exists. We can therefore rewrite the camera matrix as

$$P = M\left[I \mid M^{-1}\underline{p}_4\right], \tag{7.23}$$

so that we have the camera centre given by

$$\underline{C} = \begin{bmatrix}\widetilde{\underline{C}}\\ 1\end{bmatrix} = \begin{bmatrix}-M^{-1}\underline{p}_4\\ 1\end{bmatrix}, \tag{7.24}$$

which is a finite point.

If $P = \left[M \mid \underline{p}_4\right]$ with M singular, then there exists a $\underline{d}$ such that $M\underline{d} = \underline{0}$, and

$$P\underline{C} = \underline{0}, \quad \text{for} \quad \underline{C} = \begin{bmatrix}\underline{d}\\ 0\end{bmatrix}.$$

Thus $\underline{C} = \begin{bmatrix}\underline{d}^T & 0\end{bmatrix}^T$ is the unique camera centre of P, proving that if M is singular the camera centre is at infinity.

7.4.2 The column vectors of P

The column vectors of the camera matrix P give information about the orientation of the world coordinate frame.

If $P = \begin{bmatrix} \underline{p}_1 & \underline{p}_2 & \underline{p}_3 & \underline{p}_4 \end{bmatrix}$ then $\underline{p}_1$, $\underline{p}_2$ and $\underline{p}_3$ are the images of the vanishing points of the world X, Y and Z axes, respectively. For example, the homogeneous representation of the vanishing point of the X axis in world coordinates is $\begin{bmatrix} 1 & 0 & 0 & 0 \end{bmatrix}^T$, which maps to

$$P \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} \underline{p}_1 & \underline{p}_2 & \underline{p}_3 & \underline{p}_4 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} = \underline{p}_1. \tag{7.25}$$

Similarly, the vanishing points of the Y and Z axes map to $\underline{p}_2$ and $\underline{p}_3$, respectively. Finally $\underline{p}_4$ is the image of the origin of the world coordinate frame, since

$$P \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} \underline{p}_1 & \underline{p}_2 & \underline{p}_3 & \underline{p}_4 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} = \underline{p}_4. \tag{7.26}$$

7.4.3 The row vectors of P

The row vectors of the camera matrix P are 4-vectors which may be interpreted geometrically as certain object planes. In order to investigate these planes, we write

$$P = \begin{bmatrix} p_{11} & p_{12} & p_{13} & p_{14} \\ p_{21} & p_{22} & p_{23} & p_{24} \\ p_{31} & p_{32} & p_{33} & p_{34} \end{bmatrix} = \begin{bmatrix} \underline{r}_1^T \\ \underline{r}_2^T \\ \underline{r}_3^T \end{bmatrix}.$$

Representing a plane in homogeneous coordinates

A plane in $\mathbb{R}^3$ is determined by the expression $ax + by + cz + d = 0$, or

$$\underline{n}^T \underline{x} = 0,$$

where $\underline{n} = \begin{bmatrix} a & b & c & d \end{bmatrix}^T$ and $\underline{x} = \begin{bmatrix} x & y & z & 1 \end{bmatrix}^T$. Analogous to the case of lines and points, we can represent the plane by the vector $\underline{n}$. Note that $\begin{bmatrix} a & b & c \end{bmatrix}^T$ is in fact the normal to the plane. In keeping with standard notation, we write $\underline{\pi} = \begin{bmatrix} \pi_1 & \pi_2 & \pi_3 & \pi_4 \end{bmatrix}^T$ for the plane consisting of all points $\underline{x} = \begin{bmatrix} x & y & z & 1 \end{bmatrix}^T$ such that $\underline{\pi}^T \underline{x} = 0$.

The normal of the plane $\underline{\pi} = \begin{bmatrix} \pi_1 & \pi_2 & \pi_3 & \pi_4 \end{bmatrix}^T$ is $\begin{bmatrix} \pi_1 & \pi_2 & \pi_3 \end{bmatrix}^T$ in Euclidean coordinates. This may be represented by the point $\begin{bmatrix} \pi_1 & \pi_2 & \pi_3 & 0 \end{bmatrix}^T$ on the plane at infinity.

Principal plane

The principal plane is the plane through the camera centre parallel to the image plane, as indicated in Figure 7.5. It consists of all the points $\underline{X}$ that are mapped to the line at infinity on the image plane. Explicitly, the principal plane consists of all points $\underline{X}$ such that

$$P\underline{X} = \begin{bmatrix} \underline{r}_1^T \\ \underline{r}_2^T \\ \underline{r}_3^T \end{bmatrix} \underline{X} = \begin{bmatrix} x \\ y \\ 0 \end{bmatrix}.$$

Thus a point $\underline{X}$ lies on the principal plane of a camera if and only if $\underline{r}_3^T \underline{X} = 0$. In other words, $\underline{r}_3$ is the vector representing the principal plane of the camera, in the world coordinate system.

Note that, with $\underline{C}$ denoting the camera centre, $P\underline{C} = \underline{0}$, so that in particular $\underline{r}_3^T \underline{C} = 0$. Thus the camera centre $\underline{C}$ lies on the principal plane of the camera, as it should.

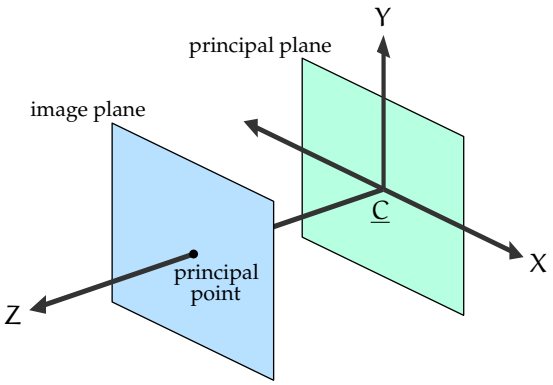


Figure 7.5: The principal plane.

Axis planes

Consider the set of points $\underline{X}$ on the plane $\underline{r}_1$. This set satisfies

$$\underline{r}_1^T \underline{X} = 0,$$

and so is imaged at

$$P\underline{X} = \begin{bmatrix} \underline{r}_1^T \\ \underline{r}_2^T \\ \underline{r}_3^T \end{bmatrix} \underline{X} = \begin{bmatrix} 0 \\ y \\ z \end{bmatrix},$$

which are points on the image y-axis. Again, with $\underline{C}$ denoting the camera centre, it follows from $P\underline{C} = 0$ that $\underline{r}_1^T \underline{C} = 0$, so that $\underline{C}$ also lies on the plane defined by $\underline{r}_1$. Consequently, the plane $\underline{r}_1$ is defined by the camera centre and the line $x = 0$ in the image. Similarly, the plane $\underline{r}_2$ is defined by the camera centre and the line $y = 0$ in the image.

Note that the planes $\underline{r}_1$ and $\underline{r}_2$ depend on the image coordinate system, i.e. they depend on the internal parameters of the camera. Thus these planes are less tightly related to the natural geometry of the camera than the principal plane. Also, the intersection point between the planes $\underline{r}_1$ and $\underline{r}_2$ is the line through the camera centre and the image origin, i.e. the back projection of the image origin. In general, this line will not coincide with the camera principal axis.

7.4.4 The principal point

Recall that the principal axis is the line from the camera centre $\underline{C}$ perpendicular to the image plane (refer to Figure 7.1). Using the theory developed in the previous sections, we have that the image plane is parallel to the principal plane, which is represented by $\underline{r}_3$, thus one can define the principal axis as being the line through $\underline{C}$, perpendicular to the plane $\underline{r}_3$. The principal point is where the principal axis intersects the image plane (denoted by $\underline{p}$ in Figure 7.1).

Using the point $\begin{bmatrix}\pi_1 & \pi_2 & \pi_3 & 0\end{bmatrix}^T$ on the plane at infinity to represent the direction of the normal to the plane $\underline{\pi} = \begin{bmatrix}\pi_1 & \pi_2 & \pi_3 & \pi_4\end{bmatrix}^T$, we determine the principal point as follows. The normal of the principal plane $\underline{r}_3$ is given by $\widehat{\underline{r}}_3 = \begin{bmatrix}p_{31} & p_{32} & p_{33} & 0\end{bmatrix}^T$. The principal point $\underline{p}$ is just the projection of $\widehat{\underline{r}}_3$, i.e. $\underline{p} = P\widehat{\underline{r}}_3$.

With the camera matrix given by

$$P = \begin{bmatrix} p_{11} & p_{12} & p_{13} & p_{14} \\ p_{21} & p_{22} & p_{23} & p_{24} \\ p_{31} & p_{32} & p_{33} & p_{34} \end{bmatrix} = \begin{bmatrix} M \mid \underline{p}_4 \end{bmatrix} = \left[\begin{array}{c|c} \underline{m}_1^T & \\ \underline{m}_2^T & \underline{p}_4 \\ \underline{m}_3^T & \end{array}\right], \tag{7.27}$$

it follows that $\widehat{\underline{r}}_3 = \begin{bmatrix}p_{31} & p_{32} & p_{33} & 0\end{bmatrix}^T = \begin{bmatrix}\underline{m}_3^T & 0\end{bmatrix}^T$, the third row of M augmented with a zero. Using this notation, the principal point is given by

$$\underline{p} = P\widehat{\underline{r}}_3 = \begin{bmatrix} M \mid \underline{p}_4 \end{bmatrix} \widehat{\underline{r}}_3 = \left[\begin{array}{c|c} \underline{m}_1^T & \\ \underline{m}_2^T & \underline{p}_4 \\ \underline{m}_3^T & \end{array}\right] \begin{bmatrix} \underline{m}_3 \\ 0 \end{bmatrix} = M\underline{m}_3. \tag{7.28}$$

Observe that only the 3×3 submatrix M of P is involved in the formula.

It is clear that the principal point does not depend on the choice of world coordinate system, i.e. that $\underline{p}$ is the same regardless of the values of R and $\widetilde{\underline{C}}$.

7.4.5 The principal axis vectors

With the camera matrix written as in (7.27) we showed that $\underline{m}_3$ points in the direction of the principal axis, with the sign undetermined. We are only interested in objects in front of the image plane, so would like to define the principal axis

vector in such a way that it points in the direction towards the front of the camera (the *positive* direction). Since P is homogeneous it is defined only up to a sign. This leaves an ambiguity as to whether $\underline{m}_3$ or $-\underline{m}_3$ points in the positive direction. We proceed to resolve this ambiguity.

Assuming a finite camera, a feature $\underline{X}_{cam}$ measured relative to the camera coordinate system is mapped to

$$\underline{x} = P_{cam}\,\underline{X}_{cam} = K\,[I \mid \underline{0}]\,X_{cam}. \tag{7.29}$$

In this case we define $\underline{v} = \det(M)\underline{m}_3$. Since

$$M = K = \begin{bmatrix} f & 0 & p_x \\ 0 & f & p_y \\ 0 & 0 & 1 \end{bmatrix},$$

it follows that $\underline{v} = \det(M)\underline{m}_3 = f^2 \begin{bmatrix} 0 & 0 & 1 \end{bmatrix}^T$ points towards the front of the camera, in the direction of the principal axis. This is irrespective of the homogeneous scaling of P_{cam}. If for example, P_{cam} is replaced with kP_{cam}, then $\underline{v}$ becomes $k^4\underline{v}$ which has the same direction as $\underline{v}$.

If the 3D point is expressed in terms of the world coordinate frame, then the camera matrix becomes $P = k\,K\left[R \mid -R\widetilde{\underline{C}}\right] = \left[M \mid \underline{p}_4\right]$, where $M = k\,KR$. Since R is a rotation matrix (no reflection) $\det(R) = 1$, and therefore $\underline{v} = \det(M)\underline{m}_3$ is again unaffected by the scaling. Since we are free to normalize P, a normalization for which $\det(M) > 0$ and $\|\underline{m}_3\| = 1$ conveniently lets $\underline{m}_3$ be a unit vector pointing in the direction of the principal axis.

In summary, $\underline{v} = \det(M)\underline{m}_3$ is a vector in the direction of the principal axis, directed towards the front of the camera. Also note that $\underline{v}$ is totally unaffected by $\widetilde{\underline{C}}$, or R, as it should. The direction of the principal axis does not depend on the choice of world coordinate system.

7.5 The action of a projective camera on points

We proceed to study two aspects of a finite projective camera: the action on points and the depth of these points.

7.5.1 Forward projection of a point at infinity

If $\underline{X} = \begin{bmatrix} \underline{d}^T & 0 \end{bmatrix}^T$ lies on the plane at infinity, then

$$\underline{x} = P\underline{X} = P\begin{bmatrix} \underline{d} \\ 0 \end{bmatrix} = \begin{bmatrix} M \mid \underline{p}_4 \end{bmatrix}\begin{bmatrix} \underline{d} \\ 0 \end{bmatrix} = M\underline{d}. \tag{7.30}$$

Note that $\underline{x}$ is only determined by M, the first 3×3 submatrix of P.

7.5.2 Back-projection of points to rays

Suppose we are given $\underline{x}$ on the image plane, and we need to determine all the points $\underline{X}$ that are mapped to $\underline{x}$, i.e. given $\underline{x}$ and P we are looking for all $\underline{X}$ such that

$$\underline{x} = P\underline{X}. \tag{7.31}$$

Of course these are just the points on the line through the camera centre $\underline{C}$ and $\underline{x}$. Since P is of rank 3, $\underline{x} \in \text{col}(P)$, and $PP^+ = I$, where P^+ is the pseudo-inverse of P. We elaborate more on the pseudo-inverse in the section 7.5.3.

Since $\underline{C} \in \text{null}(P)$ it is straightforward to show that the line

$$\underline{X}(\lambda) = P^+\underline{x} + \lambda\underline{C} \tag{7.32}$$

projects to $\underline{x}$, i.e. $\underline{x} = P\underline{X}(\lambda)$. To wit,

$$\begin{aligned} P\underline{X}(\lambda) &= P\left(P^+\underline{x} + \lambda\underline{C}\right) \\ &= PP^+\underline{x} + \lambda P\underline{C} \\ &= \underline{x} + \underline{0} = \underline{x}. \end{aligned}$$

As a side note, one might be tempted to define the line as $\underline{X}(\lambda) = \underline{x} + \lambda\underline{C}$. Why is this wrong?

7.5.3 The pseudo-inverse P^+ of P

Consider a 3×4 camera matrix P and its reduced SVD

$$P = U_r\Sigma_r V_r^T, \tag{7.33}$$

where U_r is 3×3, Σ_r is 3×3 and V_r^T is 3×4. We have seen in section 4.7.2 that the pseudo-inverse (or generalized inverse) is then given by

$$P^+ = V_r\Sigma_r^{-1}U_r^T. \tag{7.34}$$

This is easy to check:

$$PP^+ = (U_r\Sigma_r V_r^T)(V_r\Sigma_r^{-1}U_r^T) = U_r\Sigma_r(V_r^T V_r)\Sigma_r^{-1}U_r^T = U_r(\Sigma_r\Sigma_r^{-1})U_r^T = U_r U_r^T = I.$$

Let us now derive an alternative expression for P^+. Since P is 3×4 and of rank 3, its null space has dimension 1, and the left hand side of (7.31) is always in the column space of P. This implies that if $\underline{X}_r$ is any particular solution of (7.31), then $P(\underline{X}_r + \underline{X}_n) = \underline{x}$ for all $\underline{X}_n \in \text{null}(P)$. The pseudo-inverse of P selects the unique particular solution in the row space of P, as explained in section 4.7.2, i.e.

$$\underline{X_r} = P^T\underline{y}. \tag{7.35}$$

We want to find $\underline{y}$ such that $P\underline{X}_r = P(P^T\underline{y}) = PP^T\underline{y} = \underline{x}$. Since P is of maximum rank $(PP^T)^{-1}$ exists (see Theorem 7.5.1), so that $\underline{y}$ is given by

$$\underline{y} = \left(PP^T\right)^{-1}\underline{x}. \tag{7.36}$$

Combining (7.36) and (7.35) we have

$$\underline{X}_r = P^T\underline{y} = P^T\left(PP^T\right)^{-1}\underline{x}. \tag{7.37}$$

Therefore, with

$$P^+ = P^T\left(PP^T\right)^{-1}, \tag{7.38}$$

all points on the line $\underline{X}(\lambda) = P^+\underline{x} + \lambda\underline{C}$ are mapped to $\underline{x}$.

Theorem 7.5.1. *For P a 3×4 matrix, rank$(P) = 3$ implies that $(PP^T)^{-1}$ exists.*

Proof. If we can prove that the only $\underline{x}$ that satisfies $PP^T\underline{x} = \underline{0}$ is $\underline{x} = \underline{0}$, then we can conclude that the inverse of PP^T exists. Therefore, consider the set of all $\underline{x}$ such that

$$PP^T\underline{x} = \underline{0}.$$

For this set we have that $P^T\underline{x} \in$ row(P) and that $P^T\underline{x} \in$ null(P). But the intersection of row(P) and null(P) contains only the zero vector, it must therefore hold that $P^T\underline{x} = \underline{0}$. Now using that the dimension of the left null space of P or, equivalently, the dimension of the null space of P^T is zero, we conclude that $\underline{x} = \underline{0}$. Hence PP^T is invertible. □

7.5.4 Depth of points

Next we consider the distance a point lies in front of the principal plane of the camera. Consider a camera matrix

$$P = \left[M \mid \underline{p}_4\right], \tag{7.39}$$

projecting a point

$$\underline{X} = \begin{bmatrix} X & Y & Z & 1 \end{bmatrix}^T = \begin{bmatrix} \widetilde{\underline{X}}^T & 1 \end{bmatrix}^T \tag{7.40}$$

to the image point $\underline{x} = w\begin{bmatrix} x & y & 1 \end{bmatrix}^T = P\underline{X}$. Thus w is simply the homogeneous factor when $\underline{X}$ is projected onto $\underline{x}$. We now show that w also has physical meaning. Accordingly, let $\underline{C} = \begin{bmatrix} \widetilde{\underline{C}}^T & 1 \end{bmatrix}^T$ be the camera centre. Since $P\underline{C} = \underline{0}$, it follows that

$$w = \underline{r}_3^T\underline{X} = \underline{r}_3^T(\underline{X} - \underline{C}).$$

However,

$$\underline{r}_3^T(\underline{X} - \underline{C}) = \underline{m}_3^T(\widetilde{\underline{X}} - \widetilde{\underline{C}}),$$

where $\underline{m}_3$ is the principal ray direction, so that $w = \underline{m}_3^T(\widetilde{\underline{X}} - \widetilde{\underline{C}})$ can be interpreted as the dot product of the ray from the camera centre to the point $\underline{X}$, with the principal ray direction.

If the camera is normalized so that $\det M > 0$ and $\|\underline{m}_3\| = 1$, then $\underline{m}_3$ is a unit vector pointing in the *positive* axial direction and w is the depth of the point $\underline{X}$ from the camera centre $\underline{C}$ in the direction of the principal ray. Said differently, w is the (perpendicular) distance of $\widetilde{\underline{X}}$ from the principal plane.

Chapter 8

Single camera calibration

In this chapter we discuss some aspects of calculating the camera matrix P for a real camera. Firstly, given a number of point correspondences, we use our linear model to derive a set of equations that generate the matrix P. Then we provide a means of decomposing a calculated camera matrix P into its components K, R and $\widetilde{\underline{C}}$. Finally we consider the nonlinear effect of radial distortion.

8.1 Basic equations

Here we want to calculate the camera matrix P from n given point correspondences between 3D points $\underline{X}_i$ and 2D image points $\underline{x}_i$, i.e. we want to find a 3×4 camera matrix P such that

$$\underline{x}_i = P\underline{X}_i, \quad i = 1, \ldots, n. \tag{8.1}$$

Since P is an homogeneous matrix, defined only up to an arbitrary scale, it has 11 degrees of freedom. We'll shortly find that each point correspondence gives us two equations. Thus at least $5\frac{1}{2}$ point correspondences are needed to calculate P.

Setting up the necessary equations is just a little tricky since we are working in homogeneous coordinates. The equations in (8.1) are not strict equalities, they mean that $\underline{x}_i$ and $P\underline{X}_i$ are equivalent homogeneous vectors, that is they point in the same directions but need not have the same magnitudes. This can be expressed in terms of the vector cross product as $\underline{x}_i \times P\underline{X}_i = \underline{0}$. This form enables us to derive a simple linear solution for P.

If we write the camera matrix in terms of its rows as

$$P = \begin{bmatrix} \underline{r}_1^T \\ \underline{r}_2^T \\ \underline{r}_3^T \end{bmatrix}, \text{ and } \underline{x}_i = \begin{bmatrix} x_i \\ y_i \\ 1 \end{bmatrix}, \tag{8.2}$$

then

$$P\underline{X}_i = \begin{bmatrix} \underline{r}_1^T \\ \underline{r}_2^T \\ \underline{r}_3^T \end{bmatrix} \underline{X}_i = \begin{bmatrix} \underline{r}_1^T \underline{X}_i \\ \underline{r}_2^T \underline{X}_i \\ \underline{r}_3^T \underline{X}_i \end{bmatrix}, \tag{8.3}$$

and the vector cross product $\underline{x}_i \times P\underline{X}_i = \underline{0}$ becomes

$$\begin{bmatrix} \underline{0}^T & -\underline{X}_i^T & y_i \underline{X}_i^T \\ \underline{X}_i^T & \underline{0}^T & -x_i \underline{X}_i^T \\ -y_i \underline{X}_i^T & x_i \underline{X}_i^T & \underline{0}^T \end{bmatrix} \begin{bmatrix} \underline{r}_1 \\ \underline{r}_2 \\ \underline{r}_3 \end{bmatrix} = \underline{0}. \tag{8.4}$$

This equation has the form $A_i \underline{r} = \underline{0}$, where A_i is a 3×12 matrix, and $\underline{r}$ is a 12 element vector made up of the entries of the camera matrix P.

The following are important to note regarding the above equations:

- The equation $A_i \underline{r} = \underline{0}$ is a linear equation in the unknown $\underline{r}$.
- Although there are three equation in (8.4), only two of them are linearly independent (check this yourself!), so that (8.4) reduces to

$$\begin{bmatrix} \underline{0}^T & -\underline{X}_i^T & y_i \underline{X}_i^T \\ \underline{X}_i^T & \underline{0}^T & -x_i \underline{X}_i^T \end{bmatrix} \begin{bmatrix} \underline{r}_1 \\ \underline{r}_2 \\ \underline{r}_3 \end{bmatrix} = \underline{0}. \tag{8.5}$$

  From a set of n correspondences we obtain $2n$ equations, and a $2n \times 12$ matrix A by stacking up these $2n$ equations.
- The camera matrix P has 11 degrees of freedom, so that we need at least five and a half point correspondences to solve for $\underline{r}$ in $A\underline{r} = \underline{0}$.
- In practice, the rule of thumb for good estimation of a camera matrix is that the number of measurements should exceed the number of unknowns by a factor 5, so that in this case we would use at least 28 point correspondences and the SVD to find the best (least squares) solution.

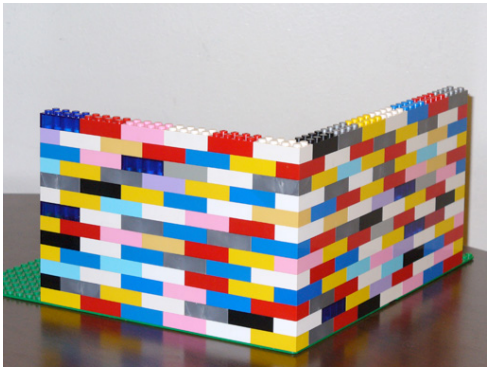


Figure 8.1: A Lego calibration object.

Finally a quick word on how to calculate the camera matrix in practice. The easiest way is to construct a calibration object with known dimensions. A cube with a chessboard pattern is very common. If you don't have access to that, build a wall from Lego bricks as shown in Figure 8.1. Lego bricks are manufactured to high tolerances with dimensions of $32 \times 16 \times 9.6$mm. Choosing the origin of the world coordinate system conveniently on the object, the corners of the bricks are known in world coordinates. It is now a matter of finding the corresponding points in pixel coordinates. It is a good idea to let the calibration object fill the frame and use as many points as possible.

8.2 Decomposing the camera matrix

Suppose we've calculated a 3×4 camera matrix P. We know that this matrix is of the form

$$\mathsf{P} = \mathsf{K}\mathsf{R}\left[\mathrm{I} \mid -\widetilde{\underline{\mathrm{C}}}\right] \tag{8.6}$$

with K the upper-triangular calibration matrix (containing the intrinsic parameters of the camera), and R and $\widetilde{\underline{\mathrm{C}}}$ the rotation and translation that relate the camera coordinate frame with the world coordinate frame.

The question we ask now is: how can we extract K, R and $\widetilde{\underline{\mathrm{C}}}$ from a given P? It would be useful if we could perform such a decomposition of P. For example, we would then get the intrinsic parameters of the camera which remain constant regardless of subsequent camera motion.

By writing P as

$$\mathsf{P} = \left[\mathsf{K}\mathsf{R} \mid -\mathsf{K}\mathsf{R}\widetilde{\underline{\mathrm{C}}}\right], \tag{8.7}$$

we observe that the first 3 columns of P are in fact KR where K is upper-triangular and R is orthogonal. We immediately think of QR factorization that decomposes a matrix into an orthogonal part and an upper-triangular part. There is a slight problem: QR factorization gives "orthogonal times upper-triangular", but we want "upper-triangular times orthogonal".

Let $\mathsf{P}_{1:3}$ be the first 3 columns of P and $\underline{\mathrm{p}}_4$ its 4th column. Here is the trick:

1. let $W = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{bmatrix}$, and note that $W^{-1} = W^{\mathsf{T}} = W$
2. QR decompose $A = (W\mathsf{P}_{1:3})^{\mathsf{T}}$, such that $A = \widehat{\mathsf{Q}}\widehat{\mathsf{R}}$
3. then let $\mathsf{K} = W\widehat{\mathsf{R}}^{\mathsf{T}}W$, $\mathsf{R} = W\widehat{\mathsf{Q}}^{\mathsf{T}}$ and $\widetilde{\underline{\mathrm{C}}} = -(\mathsf{P}_{1:3})^{-1}\underline{\mathrm{p}}_4$.

As an exercise, convince yourself that this procedure is indeed correct. That is to say, show that K is upper-triangular, R is orthogonal, and $\mathsf{K}\mathsf{R}[\mathrm{I} \mid -\widetilde{\underline{\mathrm{C}}}] = \mathsf{P}$.

We should also keep in mind that QR factorization is unique up to sign. We may fix the signs of the rows and columns of K and R by ensuring that the diagonal entries of K are all positive (as they should be — they represent the focal length of the camera).

8.3 Radial distortion

Until now, we have assumed that our cameras are linear, i.e. that the relationship between world– and image coordinates satisfy the linear relationship (7.31). For real lenses this is not true since the lenses introduce nonlinear distortions. The most important nonlinear effect is that of radial distortion. This is where a camera lens loses accuracy towards its edges, which causes straight lines, especially close to the edges of the image, to bend. An example of severe radial distortion is shown in the left hand image of Figure 8.2. Note for example the distortion of the line where the wall meets the ceiling. In the right hand image of the figure the radial distortion has been corrected and the line now appears as being straight.

(a) distorted image (b) corrected image

Figure 8.2: Removing radial distortion.

The idea is that one should first correct an image for radial distortion and then apply the theory derived in the previous lectures.

The correction for radial distortion takes place on the image plane. Suppose that the correct, undistorted coordinates are given, in pixel coordinates by $\begin{bmatrix}\widehat{x} & \widehat{y}\end{bmatrix}^T$, and the measured, distorted coordinates by $\begin{bmatrix}x & y\end{bmatrix}^T$. Assuming only radial distortion, we write the relationship between the distorted and undistorted coordinates as

$$\begin{bmatrix}\widehat{x}\\ \widehat{y}\end{bmatrix} = \begin{bmatrix}x_c\\ y_c\end{bmatrix} + L(r)\begin{bmatrix}x - x_c\\ y - y_c\end{bmatrix}, \tag{8.8}$$

where $r = \sqrt{(x - x_c)^2 + (y - y_c)^2}$ is the distance from the centre $\begin{bmatrix} x_c & y_c \end{bmatrix}^T$ of the radial distortion. Note that the distortion factor $L(r)$ is a function of radius r only. Also note that if the aspect ratio (pixel size in x and y direction) is not unitary, it has to be taken into account.

The distortion factor $L(r)$ is only defined for positive values of r and satisfies $L(0) = 1$. For an arbitrary function $L(r)$, we may use a Taylor expansion,

$$L(r) = 1 + \kappa_1 r + \kappa_2 r^2 + \kappa_3 r^3 + \cdots,$$

to approximate the distortion factor. An optimization procedure is used to estimate the distortion centre $\begin{bmatrix} x_c & y_c \end{bmatrix}^T$ as well as the expansion coefficients. The distortion parameters,

$$r_p = \{\kappa_1, \kappa_2, \kappa_3, \ldots, x_c, y_c\}, \tag{8.9}$$

are then considered part of the internal calibration of the camera.

In order to remove radial distortion in practice one has to identify one, preferably more, straight lines in the scene. Or, rather, lines that are supposed to be straight but are radially distorted. The radial distortion is corrected by the transformation (8.8), and we need to calculate its parameters (8.9). Choose a point $\begin{bmatrix} x & y \end{bmatrix}^T$ on the distorted curve in the image. Use an optimization procedure to minimize the distance between $\begin{bmatrix} \widehat{x} & \widehat{y} \end{bmatrix}^T$ in (8.8) and the straight line connecting the end points of the line. As initial guess set $\kappa_j = 0$ and the distortion centre equal to the centre of the image. How many distortion parameters one uses depend on the accuracy requirements of the application. Using only κ_1 and setting the rest equal to zero already gives pretty good results.

Chapter 9

The geometry of two views

In this chapter we introduce epipolar geometry, which is the intrinsic projective geometry between two views. It is independent of scene structure, and only depends on the two cameras' internal parameters and relative positions. The fundamental matrix F is the algebraic representation of the epipolar geometry.

9.1 Epipolar geometry

Suppose we have two calibrated cameras observing the same feature with coordinates $\underline{X}$ in the world coordinate system, as illustrated in Figure 9.1. With $\underline{C}$ denoting the centre of the first camera and $\underline{C}'$ the camera centre of the second camera, the line joining $\underline{C}$ and $\underline{C}'$ is called the *baseline*. Note that the baseline is independent of the feature $\underline{X}$.

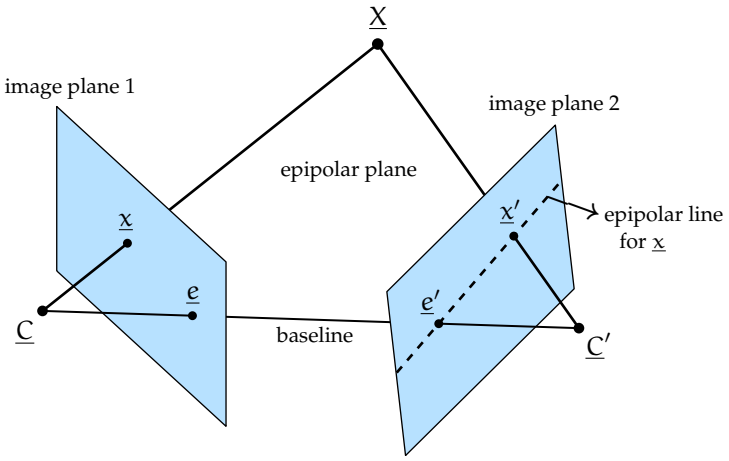


Figure 9.1: Epipolar geometry.

Furthermore, note that $\underline{X}$, $\underline{C}$ and $\underline{C}'$ form a plane, called the epipolar plane, denoted by π. Epipolar geometry is essentially the geometry of the intersection of this plane with the image planes of the two cameras.

If we are dealing with two calibrated cameras we know the camera matrices P and P' of the two cameras, hence the coordinates of their centres $\underline{C}$ and $\underline{C}'$ in a common world coordinate system. If we can also identify the projections $\underline{x} = P\underline{X}$ and $\underline{x}' = P'\underline{X}$ of $\underline{X}$ onto the two images planes, it becomes relatively straightforward to recover $\underline{X}$ through back-projection. Our immediate problem is to identify the corresponding points.

Suppose that we know only $\underline{x}$, and need to determine the corresponding point $\underline{x}'$ in the second image. The epipolar plane π is the plane through the baseline and $\underline{x}$, both known. The intersections of the epipolar plane with the two image planes (also known) are called the epipolar lines and denoted by $\underline{l}$ and $\underline{l}'$ respectively. It should be clear from Figure 9.1 that the corresponding point $\underline{x}'$ lies on the epipolar line $\underline{l}'$. Thus, given $\underline{x}$, we need only search along the epipolar line $\underline{l}'$ for its corresponding point $\underline{x}'$, a significant reduction in the search space. The first step therefore in the process of finding the corresponding point is to find the epipolar line $\underline{l}'$. In order to do this, the epipoles play an important role.

The epipoles $\underline{e}$ and $\underline{e}'$ are the intersections of the baseline with the first and second image planes respectively. Alternatively,

$$\underline{e} = P\underline{C}' \tag{9.1}$$

$$\underline{e}' = P'\underline{C}. \tag{9.2}$$

In summary, the terminology of epipolar geometry is:

- **Baseline:** The line joining the two camera centres.
- **Epipoles:** The points of intersection of the baseline and the image planes.
- **Epipolar plane:** The plane through the baseline and $\underline{x}$. The 3D feature $\underline{X}$ we want to reconstruct also lies on this plane. There is a one-parameter family of epipolar planes, all passing through the baseline.
- **Epipolar lines:** Intersection of the epipolar planes with the image planes. All epipolar lines intersect at the epipole of the corresponding image plane.

9.2 The fundamental matrix F

9.2.1 Derivation

Given $\underline{x}$ on the first image plane we would like an expression for the epipolar line $\underline{l}'$ in the second image plane, and this can be achieved by finding two points on $\underline{l}'$.

One point is already known, namely the epipole $\underline{e}'$ that is common to all epipolar lines. We need to find a second. The image on the second image plane of any point on the back-projected ray through $\underline{C}$ and $\underline{x}$ (see (7.32)) ,

$$\underline{X}(\lambda) = P^{+}\underline{x} + \lambda\underline{C}, \tag{9.3}$$

lies on the epipolar line $\underline{l}'$. For simplicity choose $P^{+}\underline{x}$. The epipolar line $\underline{l}'$ therefore passes through $\underline{e}'$ and $P'P^{+}\underline{x}$, and is given by

$$\underline{l}' = \underline{e}' \times P'P^{+}\underline{x}. \tag{9.4}$$

It is convenient to associate with any 3-vector $\underline{a} = \begin{bmatrix} a_1 & a_2 & a_3 \end{bmatrix}^{T}$ the skew-symmetric matrix

$$[\underline{a}]_{\times} = \begin{bmatrix} 0 & -a_3 & a_2 \\ a_3 & 0 & -a_1 \\ -a_2 & a_1 & 0 \end{bmatrix}. \tag{9.5}$$

We leave it as an exercise to prove that the cross product can be written as

$$\underline{a} \times \underline{b} = [\underline{a}]_{\times}\,\underline{b}. \tag{9.6}$$

Using the notation of (9.6) the epipolar line $\underline{l}'$ is given by

$$\begin{aligned} \underline{l}' &= [\underline{e}']_{\times} P'P^{+}\underline{x} \\ &= F\underline{x}, \end{aligned}$$

where

$$F = [\underline{e}']_{\times} P'P^{+} \tag{9.7}$$

is called the *fundamental matrix*.

Any point $\underline{x}'$ lying on the epipolar line $\underline{l}'$ must satisfy

$$\underline{x}'^{T}\underline{l}' = 0,$$

or

$$\begin{aligned} \underline{x}'^{T}\,[\underline{e}']_{\times} P'P^{+}\underline{x} &= 0, \\ \underline{x}'^{T}F\underline{x} &= 0. \end{aligned} \tag{9.8}$$

As an example, suppose the camera matrices of a calibrated stereo rig with the world origin at the first camera centre are

$$P = K\,[I \mid \underline{0}]\,, \qquad P' = K'\,[R \mid \underline{t}]\,.$$

Then

$$\underline{C} = \begin{bmatrix} \underline{0} \\ 1 \end{bmatrix}, \quad \text{and} \quad \underline{C}' = \begin{bmatrix} -R^{T}\underline{t} \\ 1 \end{bmatrix},$$

so that the epipole $\underline{e}'$ is given by

$$\underline{e}' = P'\underline{C} = K'\,[R \mid \underline{t}] \begin{bmatrix} \underline{0} \\ 1 \end{bmatrix} = K'\underline{t}.$$

For the camera $P = K\,[I \mid \underline{0}] = [K \mid \underline{0}]$ we have

$$P^T = \begin{bmatrix} K^T \\ \underline{0}^T \end{bmatrix},$$

so that

$$P\,P^T = [K \mid \underline{0}] \begin{bmatrix} K^T \\ \underline{0}^T \end{bmatrix} = K\,K^T,$$

and with the expression $P^+ = P^T \left(PP^T\right)^{-1}$ as derived in section 7.5.3, the pseudo-inverse is given by

$$P^+ = \begin{bmatrix} K^T \\ \underline{0}^T \end{bmatrix} \left(K\,K^T\right)^{-1} = \begin{bmatrix} K^T \\ \underline{0}^T \end{bmatrix} K^{-T}\,K^{-1} = \begin{bmatrix} K^{-1} \\ \underline{0}^T \end{bmatrix}.$$

Then, using the formula $F = [\underline{e}']_\times\, P'P^+$ as derived earlier, we get

$$F = [K'\underline{t}]_\times\, K'\,[R \mid \underline{t}] \begin{bmatrix} K^{-1} \\ \underline{0}^T \end{bmatrix} = [K'\underline{t}]_\times\, K'RK^{-1}.$$

Note that the only special choice we have made was to put the world coordinate system at the first camera centre, and to align its axes with those of the camera coordinate system, i.e. we have chosen $P = K\left[I \mid \underline{0}\right]$. We are free to choose the world coordinate system in any convenient way. We have not lost any generality.

9.2.2 Epipolar lines are related by a projectivity

Suppose $\underline{l}$ and $\underline{l}'$ are corresponding epipolar lines. Here we show that they are related by a projective transformation.

Let $\underline{k}$ be *any* line in the first image not passing through the epipole $\underline{e}$, i.e. $\underline{k}^T\underline{e} \neq 0$. Then the intersection $\underline{y}$ of the lines $\underline{l}$ and $\underline{k}$ is

$$\underline{y} = \underline{k} \times \underline{l} = [\underline{k}]_\times\, \underline{l}. \tag{9.9}$$

The epipolar lines corresponding to the point $\underline{y}$ are related by

$$\underline{l}' = F\underline{y} = F\,[\underline{k}]_\times\, \underline{l}, \tag{9.10}$$

i.e. the projective transformation $H^{-T} = F\,[\underline{k}]_\times$ maps $\underline{l}$ to $\underline{l}'$.

We can use the line $\underline{e}$ as the line $\underline{k}$, since $\underline{e}^T\underline{e} \neq 0$, so that this line does not pass through the epipole $\underline{e}$. Therefore

$$\underline{l}' = F\,[\underline{e}]_\times\, \underline{l}, \quad \text{and} \quad \underline{l} = F^T\,[\underline{e}']_\times\, \underline{l}'. \tag{9.11}$$

9.2.3 Point correspondences

Theorem 9.2.1. *For any two corresponding points $\underline{x}$ and $\underline{x}'$, it holds that*

$$\underline{x}'^{\mathsf{T}} F \underline{x} = 0. \tag{9.12}$$

Proof. If the points $\underline{x}$ and $\underline{x}'$ are corresponding points, then $\underline{x}'$ lies on the epipolar line of the second camera, hence $\underline{x}'^{\mathsf{T}} \underline{l}' = 0$. This epipolar line is given by $\underline{l}' = F\underline{x}$, so that $\underline{x}'^{\mathsf{T}} F \underline{x} = 0$. □

What about the converse, if two points $\underline{x}$ and $\underline{x}'$ satisfy $\underline{x}'^{\mathsf{T}} F \underline{x} = 0$, are they corresponding points? Hint: The rays defined by these points are co-planar. Are there any degeneracies?

The above theorem provides a convenient way of calculating the fundamental matrix directly without having to first calculate the camera matrices.

9.3 Computation of F

According to Theorem 9.2.1, each pair of corresponding points defines a relationship between the components of the fundamental matrix F. Since F is a homogeneous 3×3 matrix of rank 2 it has 7 degrees of freedom. (If this statement bothers you, you have a reason to — we haven't shown that F has rank 2. If you can't convince yourself that it is true, in a moment we'll explicitly calculate its left– and right null spaces.) Thus 7 point correspondences are needed in order to calculate F. Suppose $\underline{x} = \begin{bmatrix} x & y & 1 \end{bmatrix}^{\mathsf{T}}$ and $\underline{x}' = \begin{bmatrix} x' & y' & 1 \end{bmatrix}^{\mathsf{T}}$ are corresponding points, then they satisfy (9.8), written component-wise as

$$\begin{bmatrix} x' & y' & 1 \end{bmatrix} \begin{bmatrix} f_{11} & f_{12} & f_{13} \\ f_{21} & f_{22} & f_{23} \\ f_{31} & f_{32} & f_{33} \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = 0$$

$$f_{11}xx' + f_{12}yx' + f_{13}x' + f_{21}xy' + f_{22}yy' + f_{23}y' + f_{31}x + f_{32}y + f_{33} = 0,$$

or more conveniently,

$$\begin{bmatrix} xx' & yx' & x' & xy' & yy' & y' & x & y & 1 \end{bmatrix} \underline{f} = 0, \tag{9.13}$$

where $\underline{f} = \begin{bmatrix} f_{11} & f_{12} & f_{13} & f_{21} & f_{22} & f_{23} & f_{31} & f_{32} & f_{33} \end{bmatrix}^{\mathsf{T}}$. From this we conclude that for every point correspondence we get 1 equation, and so to solve for F we need at least 7 point correspondences from the facts that F is homogeneous and has rank 2. For this case, we have a 7×9 matrix A, and we need to solve for $\underline{f}$ from

$$A\underline{f} = \underline{0}.$$

Written like this, rank(A)= 7, and it means that the null space of A is two-dimensional. An additional nonlinear constraint, namely $\det(F) = 0$, is required to uniquely determine F. Since there are in any case measurement errors it is also necessary from an accuracy point of view to use more than 7 point correspondences. Assuming n point correspondences we need to solve for the null space of the $n \times 9$ rank 8 matrix A, i.e. we need to solve

$$A\underline{f} = \underline{0}, \tag{9.14}$$

where A is $n \times 9$. Measurement errors again destroy the null space of A and in practice one finds that its rank is 9 (the maximum possible). One is therefore forced to find a least squares approximation $\widetilde{\underline{f}}$ to the null space of A using the SVD, leading to a least squares approximation $\widetilde{F}$ of the fundamental matrix. The final step is to ensure that $\widetilde{F}$ is indeed rank 2. Again the SVD of $\widetilde{F}$ is the method of choice. We decompose $\widetilde{F}$,

$$\widetilde{F} = U\Sigma V^T = U \begin{bmatrix} \sigma_1 & 0 & 0 \\ 0 & \sigma_2 & 0 \\ 0 & 0 & \sigma_3 \end{bmatrix} V^T,$$

and then replace σ_3 with zero to obtain the rank 2 matrix

$$F = U \begin{bmatrix} \sigma_1 & 0 & 0 \\ 0 & \sigma_2 & 0 \\ 0 & 0 & 0 \end{bmatrix} V^T.$$

We now have a rank 2 homogeneous matrix F that approximates our fundamental matrix.

Supposing that the point correspondences are obtained using a feature detector and matcher, such as SIFT or SURF, how would you use RANSAC to fit a fundamental matrix robustly in the presence of mismatches?

9.4 Properties of F

- If F is the fundamental matrix for the cameras (P, P'), then F^T is the fundamental matrix for the cameras (P', P).
- For $\underline{x}$ in image 1, the epipolar line in image 2 is given by $\underline{l}' = F\underline{x}$. For $\underline{x}'$ in image 2, the epipolar line in image 1 is given by $\underline{l} = F^T\underline{x}'$.
- For any $\underline{x}$ (not equal to $\underline{e}$), $\underline{l}' = F\underline{x}$ contains the epipole $\underline{e}'$, i.e. $\underline{e}'^T F\underline{x} = 0$ for all $\underline{x} \neq \underline{e}$. If we rewrite this as $\left(\underline{e}'^T F\right)\underline{x} = 0$, for all $\underline{x} \neq \underline{e}$, we conclude that the null space of $\underline{e}'^T F$ is all $\underline{x}$, so that $\underline{e}'^T F = \underline{0}$. From this we observe that $\underline{e}'^T$ is in the left null space of F. Similarly, since $F\underline{e} = \underline{0}$, we have that $\underline{e}$ is in the right null space of F.
- F is a projective mapping from a point to a line. Here the point $\underline{x}$ in the first image defines a line $\underline{l}' = F\underline{x}$ (the epipolar line) in the second image.

9.5 The fundamental matrix for special motion

Here we look at the special structure of the fundamental matrix F for pure translation, and then use the insights gained there to interpret the case of general motion.

9.5.1 Pure translation

Suppose we have two identical cameras with the same orientation, observing the same 3D point from positions differing only by a translation, say $\underline{t}$, as illustrated in Figure 9.2. Of course this is equivalent to having one camera and translating the object by $-\underline{t}$, see Figure 9.3.

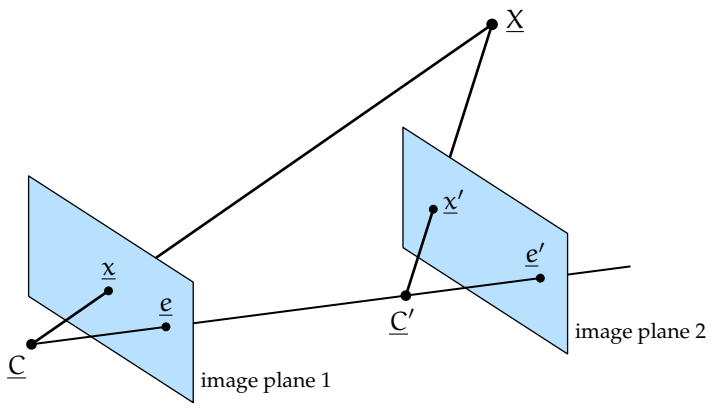


Figure 9.2: Camera undergoing pure translation (no rotation).

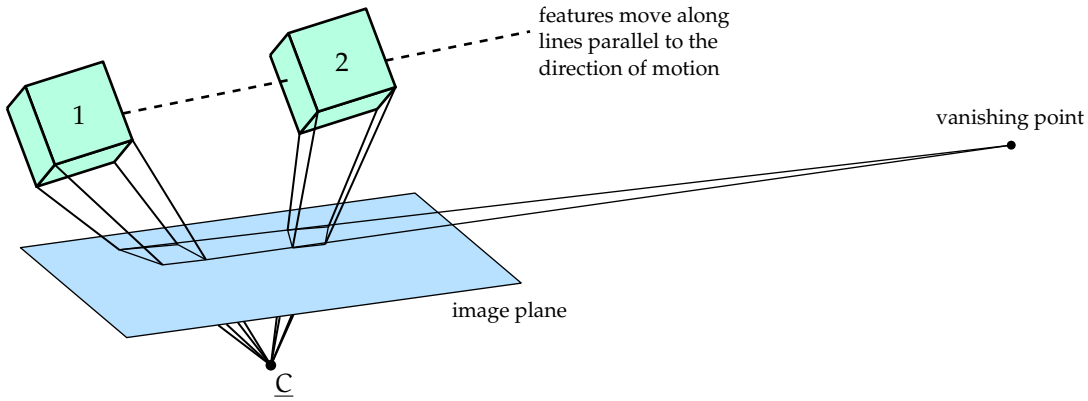


Figure 9.3: Object undergoing pure translation.

Now instead of thinking of two identical cameras, imagine a single camera that is translated, without any rotation, in the direction $\underline{t}$. During this motion the image of a 3D feature $\underline{X}$ traces out a line on the image plane of the camera. This line is the epipolar line of the second camera if we again think two identical cameras translated through a distance $\underline{t}$. To see this, note that the camera center, $\underline{C}'$ in Figure 9.2, moves along the baseline. The epipolar plane, defined by $\underline{X}$, $\underline{C}$ and $\underline{C}'$, therefore remains unchanged as the camera is translated. The image $\underline{x}'$ falls on the intersection of the epipolar plane and the image plane, i.e. it falls on the epipolar line.

Let us now imagine a rigid object with several features $\underline{X}_i$. Note that the features move along epipolar lines as the camera is translated. But epipolar lines intersect at the epipoles. Thus as we watch the images of the features $\underline{X}_i$ in the image plane of the moving camera, they all converge at the epipole as the camera moves further away from the object. Alternatively, we can imagine a stationary camera and an object moving in a straight line. This means that the features move along parallel lines in 3D, but their images converge at the epipole on the image plane.

Maybe it is clearer if one imagines the object moving all the way to infinity. Then all the features appear at the same point, i.e. all the points are mapped to one point — the epipole. Now imagine the object moving closer, then all the features emerge out of the epipole.

Let us put some mathematics behind the intuition. Consider the two cameras

$$P = K\,[I \mid \underline{0}]\,, \qquad P' = K\,[I \mid \underline{t}]\,. \tag{9.15}$$

Note these cameras only differ by a translation of $\underline{t}$, as in Figure 9.2. Then, from the example in section 9.2.1 the fundamental matrix of the camera pair (P, P') is given by

$$F = [\underline{e}']_\times K K^{-1} = [\underline{e}']_\times\,, \tag{9.16}$$

and the fundamental matrix of the camera pair (P', P) is similarly given by

$$F^T = [\underline{e}]_\times\,. \tag{9.17}$$

Since F is anti-symmetric this shows that the epipoles are identical for pure translation, easily confirmed by a direct calculation (keeping in mind that we are using homogeneous coordinates),

$$\underline{e}' = P'\underline{C} = K\,[I \mid \underline{t}] \begin{bmatrix} \underline{0} \\ 1 \end{bmatrix} = K\underline{t}, \tag{9.18}$$

and

$$\underline{e} = P\underline{C}' = K\,[I \mid \underline{0}] \begin{bmatrix} -\underline{t} \\ 1 \end{bmatrix} = -K\underline{t}. \tag{9.19}$$

Furthermore the anti-symmetry of F shows that $\underline{x}'^T F \underline{x} = 0$ and $\underline{x}^T F \underline{x} = 0$, i.e. $\underline{x}$ and $\underline{x}'$ lie on the same epipolar line.

Let us now specialize even further and investigate the case of translation parallel to the x-axis, so that $\underline{t} = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix}^T$. For the calibration matrix

$$K = \begin{bmatrix} \alpha_x & s & x_0 \\ 0 & \alpha_y & y_0 \\ 0 & 0 & 1 \end{bmatrix}, \tag{9.20}$$

it follows that

$$\underline{e} = \underline{e}' = \begin{bmatrix} \alpha_x & s & x_0 \\ 0 & \alpha_y & y_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} \alpha_x \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} \tag{9.21}$$

in homogeneous coordinates, which is a point at infinity in the direction of the x-axis. From (9.16), the fundamental matrix for this translation is

$$F = [\underline{e}']_\times = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & -1 \\ 0 & 1 & 0 \end{bmatrix}, \tag{9.22}$$

and

$$0 = \underline{x}'^T F \underline{x} = \begin{bmatrix} x' & y' & z' \end{bmatrix} \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & -1 \\ 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} x' & y' & z' \end{bmatrix} \begin{bmatrix} 0 \\ -z \\ y \end{bmatrix} = -y'z + z'y, \tag{9.23}$$

or, equivalently,

$$\frac{y'}{z'} = \frac{y}{z}. \tag{9.24}$$

But these are just the y coordinates of the image of $\underline{X}$ in Euclidean coordinates. Thus the epipolar lines are parallel to the image x-axis. This really should be no surprise given that the epipoles are at $\begin{bmatrix} 1 & 0 & 0 \end{bmatrix}^T$. We will encounter this situation again when we discuss image rectification.

Let us now investigate how a 3D feature $\underline{X} = \begin{bmatrix} X & Y & Z & 1 \end{bmatrix}^T$ projects onto a camera moving in the x-direction, i.e. $\underline{t}(t) = \begin{bmatrix} t & 0 & 0 \end{bmatrix}^T$. Assuming a calibration matrix of the form given in (9.20), initially (at $t = 0$) the image of $\underline{X}$ is

$$\underline{x}_0 = P\underline{X} = K \begin{bmatrix} I \mid \underline{0} \end{bmatrix} \underline{X} = K \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = K\widetilde{\underline{X}}, \tag{9.25}$$

or

$$\begin{bmatrix} x_0 \\ y_0 \\ z_0 \end{bmatrix} = \begin{bmatrix} \alpha_x & s & x_0 \\ 0 & \alpha_y & y_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix}, \tag{9.26}$$

where $\widetilde{\underline{X}}$ denotes the inhomogeneous coordinates of $\underline{X}$. Thus $z_0 = Z$ which is just the depth of the feature as discussed in section 7.5.4.

For the moving camera,

$$\underline{x}(t) = K \begin{bmatrix} I & | \underline{t}(t) \end{bmatrix} \begin{bmatrix} \widetilde{\underline{X}} \\ 1 \end{bmatrix} = \underline{x}_0 + K\underline{t}(t). \tag{9.27}$$

Note that the depth of $\underline{X}$ remains the same, i.e. $z(t) = z_0 = Z$ as it should. Let us now calculate the x-component of image of $\underline{X}$ in Euclidean coordinates. From (9.27) it follows that

$$\frac{x(t)}{z} = \frac{x_0}{z} + \frac{1}{Z}\alpha_x t. \tag{9.28}$$

This describes a familiar observation: objects closer to the observer, i.e. smaller depth Z appear to be moving faster that objects further away, i.e. larger Z.

9.5.2 General motion

The pure translation, discussed in the previous section, provides additional understanding into the case of general motion.

Given two arbitrary cameras, with corresponding points $\underline{x}$ and $\underline{x}'$ respectively, we may rotate the first camera so that the two cameras are aligned. This rotation may be simulated by applying a projective transformation to the first image. A further correction may be applied to the first image to account for any difference in the calibration matrices of the two cameras. The result of these two corrections is a projective transformation H of the first image, so that the corrected points $\hat{\underline{x}}$ are related to the original points through $\hat{\underline{x}} = H\underline{x}$.

If one assumes that these corrections have been made, then the effective relationship of the two cameras to each other is that of a pure translation. Consequently, the fundamental matrix becomes

$$\widehat{F} = [\underline{e}']_\times ,$$

so that corresponding points satisfy $\underline{x}'^T \widehat{F} \hat{\underline{x}} = 0$, where $\hat{\underline{x}} = H\underline{x}$ is the corrected point in the first image. From this we deduce that

$$\underline{x}'^T [\underline{e}']_\times H\underline{x} = 0,$$

and so the fundamental matrix relating the original corresponding points $\underline{x}$ and $\underline{x}'$ is $F = [\underline{e}']_\times H$. We return to this topic when we discuss image rectification in Chapter 11.

Chapter 10

Stereo calibration

In this chapter we consider the problem of calibrating a set of two cameras, i.e. the problem of determining the two camera matrices. We know from the discussion in the previous chapter that it is possible to compute a fundamental matrix from point correspondences. The question is: can we recover camera matrices from such a fundamental matrix?

10.1 Camera matrices from a fundamental matrix

We observed that the fundamental matrix has seven degrees of freedom. Each camera matrix has eleven degrees of freedom. One would therefore not expect to be able to derive the camera matrices directly from the fundamental matrix. In the following section we investigate exactly how much of the camera matrices P and P' can be recovered from the fundamental matrix F.

10.1.1 Projective invariance and canonical cameras

As hinted above, it is not possible to retrieve the camera matrices from the fundamental matrix. The best we can do is to recover them up to a projective transformation. Consider a pair of cameras P and P', and another pair $\widetilde{P} = PH$ and $\widetilde{P}' = P'H$, where H is a non-singular 4×4 matrix, i.e. the two pairs are related by a projective transformation in 3 space. Then the fundamental matrix of the first pair is

$$F = [\underline{e}']_\times P'P^+, \tag{10.1}$$

where P^+ is the pseudo-inverse of P, while the fundamental matrix of the second pair is

$$\widetilde{F} = [\widetilde{\underline{e}}']_\times \widetilde{P}'\widetilde{P}^+, \tag{10.2}$$

where $\widetilde{P}^+$ is the pseudo-inverse of $\widetilde{P}$. Since we have the relation between the two pairs of cameras, we expect that we can write the fundamental matrix of the second pair $\widetilde{F}$ in terms of the first pair's fundamental matrix. In fact, the two fundamental matrices are equal. To see this, let $\underline{\widetilde{C}}$ be the camera centre of the camera with camera matrix $\widetilde{P}$, i.e.

$$\widetilde{P}\underline{\widetilde{C}} = PH\underline{\widetilde{C}} = \underline{0}. \tag{10.3}$$

Thus the camera centre $\underline{C}$ of camera P satisfies $\underline{C} = H\underline{\widetilde{C}}$. Since H is non-singular, $\underline{\widetilde{C}} = H^{-1}\underline{C}$. Relating the epipoles of each pair, it follows that

$$\underline{\widetilde{e}}' = \widetilde{P}'\underline{\widetilde{C}} = P'H\left(H^{-1}\underline{C}\right) = P'\underline{C} = \underline{e}'. \tag{10.4}$$

It is easily confirmed that a pseudo-inverse of $\widetilde{P} = PH$ is $\widetilde{P}^+ = H^{-1}P^+$. Note that this is not *the* pseudo-inverse, but *a* pseudo-inverse in the sense that

$$\widetilde{P}\widetilde{P}^+ = (PH)\left(H^{-1}P^+\right) = PHH^{-1}P^+ = PP^+ = I_{3\times 3}, \tag{10.5}$$

which is all that we require. The fundamental matrix of the second pair is then given by

$$\widetilde{F} = \left[\underline{\widetilde{e}}'\right]_\times \widetilde{P}'\widetilde{P}^+ = [\underline{e}']_\times (P'H)\left(H^{-1}P^+\right) = [\underline{e}']_\times P'HH^{-1}P^+ = [\underline{e}']_\times P'P^+ = F.$$

This proves that although a set of camera matrices P and P' uniquely determines a corresponding fundamental matrix F, a fundamental matrix F determines a pair of camera matrices at best up to a right-multiplication by a projective transformation in 3-space. In section 10.1.2 below we prove that this is the full extent of the ambiguity.

Since there is always a projective ambiguity in the pair of camera matrices corresponding to a given fundamental matrix, it is common practice to define a specific canonical form of a pair of camera matrices, where the first camera is in the simple form $[I \mid \underline{0}]$. In order to show that this is indeed possible, assume a general 3×4 camera matrix P and define

$$P^* = \begin{bmatrix} P \\ \underline{r}^\mathsf{T} \end{bmatrix}, \tag{10.6}$$

where $\underline{r}$ is any 4-vector so that P^* is non-singular. This is always possible since P is of rank 3. Then

$$[I \mid \underline{0}]\, P^* = [I \mid \underline{0}] \begin{bmatrix} P \\ \underline{r}^\mathsf{T} \end{bmatrix} = P, \tag{10.7}$$

and since $H = (P^*)^{-1}$ exists, the canonical form is obtained from the projective transformation

$$[I \mid \underline{0}] = PH. \tag{10.8}$$

Since a pair of camera matrices retrieved from a fundamental matrix is only known up to a projective transformation, we might just as well position one of them at the world origin with the simplest possible camera matrix.

The fundamental matrix for a canonical camera pair,

$$P = [I \mid \underline{0}], \qquad \text{and} \qquad P' = [M \mid \underline{m}], \tag{10.9}$$

follows easily from the example in section 9.2.1. There we showed that for the camera pair

$$P = K[I \mid \underline{0}], \qquad P' = K'[R \mid \underline{t}], \tag{10.10}$$

the fundamental matrix is

$$F = [K'\underline{t}]_\times K'RK^{-1}. \tag{10.11}$$

For canonical cameras $K = I$, $K'R = M$ and $K'\underline{t} = \underline{m}$, the fundamental matrix is given by

$$F = [K'\underline{t}]_\times K'RK^{-1} = [\underline{m}]_\times M. \tag{10.12}$$

10.1.2 Projective ambiguity of cameras given F

We now show that the only ambiguity in reconstructing the camera matrices from a fundamental matrix is a projective ambiguity.

Let F be the fundamental matrix of two pairs of cameras, (P, P') and $(\widetilde{P}, \widetilde{P}')$ respectively. We want to show that there exists a non-singular 4×4 matrix H such that $\widetilde{P} = PH$ and $\widetilde{P}' = P'H$. The problem may be simplified by writing both pairs of camera matrices in canonical form, $P = \widetilde{P} = \begin{bmatrix} I & \underline{0} \end{bmatrix}$, by applying a projective transformation to each pair as necessary. Writing the camera pairs as $P = \widetilde{P} = \begin{bmatrix} I \mid \underline{0} \end{bmatrix}$, $P' = \begin{bmatrix} A \mid \underline{a} \end{bmatrix}$ and $\widetilde{P}' = \begin{bmatrix} \widetilde{A} \mid \underline{\widetilde{a}} \end{bmatrix}$ then F is given by

$$F = [\underline{a}]_\times A = [\underline{\widetilde{a}}]_\times \widetilde{A}. \tag{10.13}$$

This implies that $\underline{a} = k^{-1}\underline{\widetilde{a}}$ and $\widetilde{A} = k^{-1}(A + \underline{a}\,\underline{v}^T)$ for some non-zero k and 3-vector $\underline{v}$. Indeed, both $\underline{a}$ and $\underline{\widetilde{a}}$ lie in the left null space of F, i.e. $\underline{a}^T F = \underline{0}^T = \underline{\widetilde{a}}^T F$, since F has rank 3 (its left null space is one-dimensional), it implies that $\underline{a} = k^{-1}\underline{\widetilde{a}}$. Together with (10.13), this implies that $[\underline{a}]_\times(k\widetilde{A} - A) = 0$. The null space of $[\underline{a}]_\times$ is also one-dimensional, spanned by $\underline{a}$. Therefore $k\widetilde{A} - A = \underline{a}\,\underline{v}^T$, or

$$\widetilde{A} = k^{-1}(A + \underline{a}\,\underline{v}^T), \tag{10.14}$$

and the camera matrices become

$$P' = \begin{bmatrix} A & \underline{a} \end{bmatrix} \tag{10.15}$$

$$\widetilde{P}' = \begin{bmatrix} k^{-1}\left(A + \underline{a}\underline{v}^T\right) & k\underline{a} \end{bmatrix}. \tag{10.16}$$

A direct calculation with

$$H = \begin{bmatrix} k^{-1}I & \underline{0} \\ k^{-1}\underline{v}^T & k \end{bmatrix}, \tag{10.17}$$

shows that they are projectively related: $PH = \begin{bmatrix} I & \underline{0} \end{bmatrix} H = k^{-1}\widetilde{P}$, and $P'H = \begin{bmatrix} A & \underline{a} \end{bmatrix} H = \begin{bmatrix} k^{-1}(A + \underline{a}\underline{v}^T) & k\underline{a} \end{bmatrix} = \begin{bmatrix} \widetilde{A} & \widetilde{\underline{a}} \end{bmatrix} = \widetilde{P}'$.

This can be understood intuitively by a counting argument: the two camera matrices P and P' are both 3×4 homogeneous matrices, and therefore have 11 degrees of freedom each, making a total of 22 degrees of freedom. The fundamental matrix F is a 3×3 homogeneous matrix of rank 2, i.e. it has 7 degrees of freedom. The difference is 15 degrees of freedom, which account for the 15 degrees of freedom in the 4×4 homogeneous matrix H which represents the projective transformation in 3-space.

10.1.3 Canonical cameras given F

We have shown that F determines a camera pair up to a projective transformation of 3-space. Here we quote a few theorems concerning a fundamental matrix, for the proofs see section 9.5.3 in Hartley and Zisserman's book[1]. We conclude by quoting a specific formula for a pair of cameras with canonical form given F.

Firstly, a characterization of the fundamental matrix F corresponding to a pair of camera matrices.

Theorem 10.1.1. *A nonzero matrix F is the fundamental matrix corresponding to a pair of camera matrices P and P' if and only if ${P'}^T F P$ is skew-symmetric.*

A particular camera pair in canonical form for a given fundamental matrix can be obtained as described in the next theorem.

Theorem 10.1.2. *Let F be a fundamental matrix and S any skew-symmetric matrix. Define the pair of camera matrices*

$$P = \begin{bmatrix} I \mid \underline{0} \end{bmatrix}, \qquad \text{and} \qquad P' = \begin{bmatrix} SF \mid \underline{e}' \end{bmatrix},$$

where $\underline{e}'$ is the epipole such that ${\underline{e}'}^T F = \underline{0}$, and P' is rank 3. Then F is the fundamental matrix corresponding to the camera pair (P, P').

The above gives a method of constructing a set of valid cameras matrices for a given fundamental matrix. We are free to choose any skew-symmetric matrix S, as long as P' is of rank 3. The result below gives a valid choice for S.

Theorem 10.1.3. *The camera matrices corresponding to a fundamental matrix F may be chosen as*

$$P = \begin{bmatrix} I \mid \underline{0} \end{bmatrix}, \qquad \text{and} \qquad P' = \begin{bmatrix} \begin{bmatrix} \underline{e}' \end{bmatrix}_\times F \mid \underline{e}' \end{bmatrix}.$$

[1] R. Hartley & A. Zisserman, *Multiple View Geometry in Computer Vision*, 2nd ed., Cambridge University Press, 2003.

Note that for this choice, the camera P' has a left 3×3 sub-matrix $\left[\underline{e}'\right]_\times F$ which has rank 2. This corresponds to a camera at infinity. However, there is no particular reason to avoid this situation, but one can prove the following:

Theorem 10.1.4. *Given a fundamental matrix F, the general formula for a pair of camera matrices in canonical form is given by*

$$P = \left[I \mid \underline{0}\right], \qquad \textit{and} \qquad P' = \left[\left[\underline{e}'\right]_\times F + \underline{e}'\underline{v}^T \mid \lambda\underline{e}'\right],$$

where $\underline{v}$ is any 3-vector and λ is a nonzero scalar.

10.2 The essential matrix E

Since it is possible to calibrate cameras separately, with respect to different coordinate systems, it often happens that the camera calibration matrix K is known but not the full projection matrix P. The question then arises how to exploit this additional knowledge. For instance, we showed that it is possible to calculate the fundamental matrix directly from the scene, provided a sufficient number of corresponding points can be identified. In the previous section we investigated to what extent it is possible to retrieve the full camera matrices from the fundamental matrix. The question that we address in this section is to what extent can we retrieve the full camera matrices if the intrinsic calibration parameters K are known.

10.2.1 The essential matrix

If the calibration matrix K is known, one can remove the effect of the camera on the fundamental matrix by using normalized image coordinates. Writing the camera matrix in the form

$$P = K\left[R \mid \underline{t}\right], \tag{10.18}$$

with the calibration matrix K known, the normalized coordinates are defined as $\widehat{\underline{x}} = K^{-1}\underline{x}$, where $\underline{x}$ is a point in image coordinates. This means that $\widehat{\underline{x}} = \left[R \mid \underline{t}\right]\underline{X}$, can be thought of as the image of the point $\underline{X}$ with respect to the camera $\left[R \mid \underline{t}\right]$ with I as calibration matrix. Supposing that we are dealing with two cameras one can again without loss of generality assume that the world coordinate system coincides with the first camera coordinate system so that the camera matrices are given by $P = \left[I \mid \underline{0}\right]$ and $P' = \left[R \mid \underline{t}\right]$. The *essential matrix* E is the fundamental matrix of these special camera matrices, i.e.

$$E = [\underline{t}]_\times R = R[R^T\underline{t}]_\times. \tag{10.19}$$

The defining equation for the essential matrix is

$$\widehat{\underline{x}}'^T E\widehat{\underline{x}} = 0, \tag{10.20}$$

in terms of the normalized coordinates for corresponding points $\widehat{\underline{x}}$ and $\widehat{\underline{x}}'$. Transforming back to image coordinates give $\underline{x}'^{\mathsf{T}}K'^{-\mathsf{T}}EK^{-1}\underline{x} = 0$. Since the fundamental matrix satisfies $\underline{x}'^{\mathsf{T}}F\underline{x} = 0$, it follows that $F = K'^{-\mathsf{T}}EK^{-1}$, or

$$E = K'^{\mathsf{T}}FK. \tag{10.21}$$

10.2.2 Properties of the essential matrix

Both the translation $\underline{t}$ and the rotation R in the expression for the essential matrix (10.19) has three degrees of freedom, so that, together with the fact that it is a homogeneous matrix (defined up to a scale), the essential matrix has five degrees of freedom. Compare this to the fundamental matrix's seven degrees of freedom. The reduced degrees of freedom are a result of extra constraints that are satisfied by the essential matrix.

The following result follows directly from the real Schur decomposition theorem.

Lemma 10.2.1. *If the 3×3 matrix S is real and skew-symmetric, then $S = UZU^{\mathsf{T}}$ where*

$$Z = \lambda \begin{bmatrix} 0 & 1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}.$$

Two of the eigenvalues of S are imaginary, and the third one is zero.

Making use of this result one can show that an essential matrix is characterized in terms of its singular values.

Theorem 10.2.2. *The 3×3 matrix A is an essential matrix if and only if two of its singular values are equal and the third is zero.*

Proof. The essence of the proof lies in the factorization $E = [\underline{t}]_\times R = SR$ where S is a skew symmetric matrix and R a rotation. First note that for

$$\Sigma = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \quad Z = \begin{bmatrix} 0 & 1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \quad \text{and} \quad W = \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \tag{10.22}$$

$Z = -\Sigma W$ and $Z = \Sigma W^{\mathsf{T}}$. This becomes the source of the ambiguities in finding the camera matrices, given the essential matrix. From Lemma 10.2.1 follows that S can be factorized as $S = kUZU^{\mathsf{T}}$, where U is an orthogonal matrix, and k is a scalar. Since $Z = -\Sigma W$, it follows that $S = -kU\Sigma WU^{\mathsf{T}}$, and $E = -kU\Sigma\left(WU^{\mathsf{T}}R\right)$. Since $W^{\mathsf{T}}W = WW^{\mathsf{T}} = I$, this is the singular value decomposition of E, with two equal singular values and the third one zero.

Conversely, a matrix with two equal singular values and the third singular value equal to zero, may be factorized into SR in this way. □

The SVD of an essential matrix is given by $E = c\,U\Sigma V^T$, where c is a scalar, but since the essential matrix E is homogeneous, we can write it as $E = U\Sigma V^T$.

10.3 Extraction of cameras from an essential matrix

The essential matrix can be computed directly from

$$\widehat{\underline{x}}'^T E \widehat{\underline{x}} = 0, \tag{10.23}$$

using normalized image coordinates, or using the relationship between the fundamental matrix and the essential matrix,

$$E = K'^T F K. \tag{10.24}$$

Given the essential matrix E, one can retrieve the camera matrices up to a scale and a four-fold ambiguity, i.e. there are four possible solutions (except for scale). Recall that in the case of the fundamental matrix, we could only retrieve cameras up to a projective ambiguity.

Given the essential matrix E, there is no loss in generality to assume that $P = \begin{bmatrix} I \mid \underline{0} \end{bmatrix}$ and it remains to determine $P' = [R \mid \underline{t}]$. For two cameras of this form the essential matrix is given by

$$E = [\underline{t}]_\times R = SR, \tag{10.25}$$

where $S = [\underline{t}]_\times$ is a skew-symmetric matrix and R is a rotation matrix. Note that once we have S and R, we have P'. The task therefore is: given E, factorize it into a skew-symmetric matrix times a rotation matrix. The following result gives us the factors S and R in terms of the SVD of E.

Theorem 10.3.1. *Suppose the SVD of a* 3×3 *matrix* E *is*

$$E = U\Sigma V^T,$$

then, using the notation of (10.22), there are two possible factorizations (ignoring signs) of $E = SR$, *as follows*

$$S = UZU^T,$$

and

$$R = UWV^T \quad \textit{or} \quad R = UW^TV^T.$$

Proof. First we verify that these factorizations are valid. To see this, calculate

$$SR = \left(UZU^T\right)\left(UWV^T\right) = UZU^TUWV^T = UZWV^T,$$

or

$$SR = \left(UZU^T\right)\left(UW^TV^T\right) = UZU^TUWV^T = UZW^TV^T.$$

Since

$$ZW = \begin{bmatrix} 0 & 1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} = \Sigma,$$

and

$$ZW^T = \begin{bmatrix} 0 & 1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} 0 & 1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} -1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 0 \end{bmatrix} = -\Sigma$$

we have that $S R = U \Sigma V^T = E$, or $S R = U\ (-\Sigma)\ V^T = E$.

Also, since Z is skew-symmetric, we have that

$$S^T = \left(UZU^T\right)^T = UZ^TU^T = -UZU^T = -S,$$

proving that S is skew-symmetric as required, and, since W is orthogonal,

$$R^TR = \left(UWV^T\right)^T \left(UWV^T\right) = VW^TU^T\, UWV^T = I,$$

(similarly $R\,R^T = I$), proving that R is orthogonal as required. It only remains to prove that these two factorizations are the only two valid factorizations.

Suppose $E = SR$, then the form of S is determined by the fact that its left null space is the same as that of E, as the following argument shows. Since R is a rotation matrix, hence invertible, $\underline{x}^TS = \underline{0}^T \Leftrightarrow \underline{x}^TSR = \underline{0}^T$, thus $\underline{x}^TS = \underline{0}^T \Leftrightarrow \underline{x}^TE = \underline{0}^T$. We know that S can be written as $S = U'ZU'^T$, and if we write $U = \begin{bmatrix} \underline{u}_1 & \underline{u}_2 & \underline{u}_3 \end{bmatrix}$ where $E = U\Sigma V^T$, the fact that S and E have the same null spaces shows that $\underline{u}_3^TS = \underline{0}^T$, i.e. the first two columns of U' are orthogonal to $\underline{u}_3$, hence that $\underline{u}_3$ is the third column of U'. This does not necessarily imply that $U' = U$. Note, however, that since the two singular values of E are equal, its SVD is not unique. Any two orthogonal vectors in the plane orthogonal to $\underline{u}_3$ are valid. This is exactly the same condition we have on the first two columns of U', and we may therefore without any loss in generality assume that $U' = U$, and $S = UZU^T$.

The rotation R may be written as $R = UXV^T$, where X is some rotation matrix. Then

$$U\,\Sigma\,V^T = E = SR = \left(UZU^T\right)\left(UXV^T\right) = U\,(UZX)\,V^T.$$

From this we conclude that $ZX = \Sigma$. If we write

$$X = \begin{bmatrix} \underline{x}_1^T \\ \underline{x}_2^T \\ \underline{x}_3^T \end{bmatrix}$$

then

$$ZX = \begin{bmatrix} \underline{x}_2^T \\ -\underline{x}_1^T \\ \underline{0}^T \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} = \Sigma.$$

Thus $\underline{x}_2^T = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix}$, $\underline{x}_1^T = \begin{bmatrix} 0 & -1 & 0 \end{bmatrix}$, and since X is a rotation matrix, i.e. its columns are orthogonal, it follows that $\underline{x}_3^T = \begin{bmatrix} 0 & 0 & \pm 1 \end{bmatrix}$. Thus, $X = W$ or $X = W^T$, as required. □

Once we have factorized $E = SR$, $\underline{t}$ is determined from $S = [\underline{t}]_\times$. Since E is homogeneous, we know S, hence $\underline{t}$, only up to scale. However, the following choice of scale for S, $S = UZU^T = U\Sigma WU^T = U\Sigma\left(UW^T\right)^T$, induces a scale on $\underline{t}$. To see what it is, note that from the choice of S, its Frobenius norm, $\|S\|_F = \sqrt{2}$. A direct calculation of $S = [\underline{t}]_\times$, on the other hand, shows that $\sqrt{2} = \|S\|_F = \sqrt{2}(t_1^2+t_2^2+t_3^2)$. Thus $\|\underline{t}\| = 1$, which is a convenient normalization for the baseline of the two cameras. Since we have that $S\underline{t} = \underline{0}$, we conclude that $\underline{t}$ is in the null space of S, so that $\underline{t}$ is the last column of UW^T, i.e. $\underline{t} = U\begin{bmatrix}0 & 0 & 1\end{bmatrix}^T = \underline{u}_3$ (the last column of U). Since the sign of E remains undetermined, we have four possible choices for the factorization of E, as summarized in the following result.

Theorem 10.3.2. *For a given essential matrix with SVD*

$$E = U\Sigma V^T,$$

and $P = \begin{bmatrix}I \mid \underline{0}\end{bmatrix}$*, using the notation of (10.22), there are four possiblities for* P'*:*

(a) $\begin{bmatrix}UWV^T \mid \underline{u}_3\end{bmatrix}$, *(b)* $\begin{bmatrix}UWV^T \mid -\underline{u}_3\end{bmatrix}$, *(c)* $\begin{bmatrix}UW^TV^T \mid \underline{u}_3\end{bmatrix}$, *(d)* $\begin{bmatrix}UW^TV^T \mid -\underline{u}_3\end{bmatrix}$.

It is easy to see that the difference between (a) and (b) (and then also (c) and (d)) in Theorem 10.3.2 is simply that the direction of translation has been reversed, see Figure 10.1. The difference between (a) and (c) requires a little more work.

It is easily verified that

$$\begin{bmatrix}UW^TV^T \mid \underline{u}_3\end{bmatrix} = \begin{bmatrix}UWV^T \mid \underline{u}_3\end{bmatrix}\begin{bmatrix}VW^TW^TV^T & \underline{0} \\ \underline{0}^T & 1\end{bmatrix},$$

and that $VW^TW^TV^T = V\,\mathrm{diag}(-1,-1,1)V^T$ is a rotation through 180^o about the baseline, see Figure 10.1. Similarly for the cameras in (b) and (d).

To decide which of the four possible factorizations is the one that describes a particular setup, choose a specific world point, then test it to determine the situation where it is in front of both cameras.

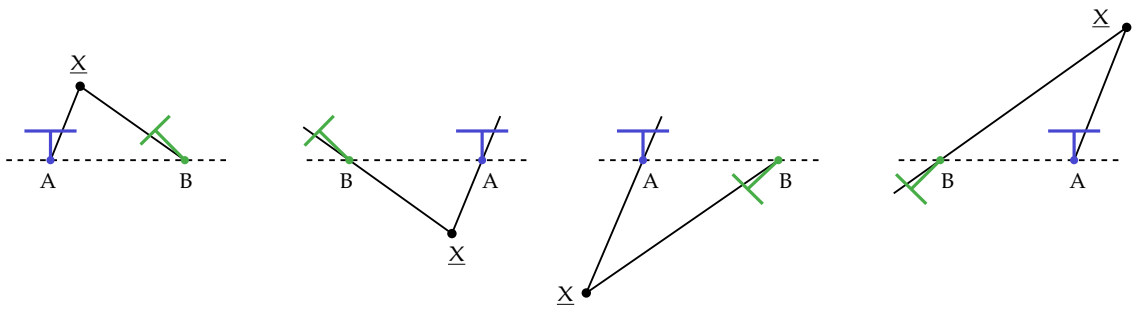


Figure 10.1: Illustrating the four possible configurations of two cameras given an essential matrix E.

Chapter 11

Image rectification

One of the main motivations for our study of epipolar geometry is that it makes life a little easier finding corresponding points in two images — given a feature in one image, its corresponding feature in the second image falls on the epipolar line. Thus one only has to search along epipolar lines to find corresponding points. Image rectification takes this a step further — the epipolar lines are projectively transformed to become parallel, restricting the search space for corresponding features even further.

Here we develop some techniques to re-sample two images obtained from a stereo setup so that the resulting images have parallel epipolar lines.

11.1 Mapping the epipole to infinity

We want to develop a projective transformation that results in epipolar lines being parallel to the x-axis, which means that we want a projective transformation H that maps the epipole of an image to the point at infinity $\begin{bmatrix}1 & 0 & 0\end{bmatrix}^T$.

This requirement does not uniquely determine a projective transformation, there are still degrees of freedom open for H, and if we are careless in choosing H, the re-sampled image will be severely distorted. We would like the re-sampled image to look similar to the original image, i.e. keep the distortion to a minimum.

One way of ensuring good results is to require that the projective transformation H should behave similar to an isometric transformation (rotation and scaling) in the neighborhood of some chosen point $\underline{x}_0$. A suitable choice for $\underline{x}_0$ could be the centre of the image.

Now suppose that we choose $\underline{x}_0$ as the origin and that the epipole is $\underline{e} = \begin{bmatrix}f & 0 & 1\end{bmatrix}^T$ (lies on the x-axis).

Then the transformation

$$G = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ -\frac{1}{f} & 0 & 1 \end{bmatrix} \tag{11.1}$$

satisfies all our above-mentioned requirements. Firstly, the transformation takes the epipole to

$$G\underline{e} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ -\frac{1}{f} & 0 & 1 \end{bmatrix} \begin{bmatrix} f \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} f \\ 0 \\ 0 \end{bmatrix}, \tag{11.2}$$

a point at infinity in the direction of the x-axis. A point $\begin{bmatrix} x & y & 1 \end{bmatrix}^T$ is mapped to the point

$$\begin{bmatrix} \widehat{x} \\ \widehat{y} \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ -\frac{1}{f} & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} x \\ y \\ 1 - \frac{x}{f} \end{bmatrix}.$$

We also need to convince ourselves that in a neighborhood of the image origin $\begin{bmatrix} x & y & 1 \end{bmatrix}^T = \begin{bmatrix} 0 & 0 & 1 \end{bmatrix}^T$ the transformation G behaves similar to an isometry. If $|\frac{x}{f}| < 1$, then

$$\begin{bmatrix} \widehat{x} \\ \widehat{y} \\ 1 \end{bmatrix} = \begin{bmatrix} x \\ y \\ 1 - \frac{x}{f} \end{bmatrix} = \begin{bmatrix} x\left(1 + \frac{x}{f} + \cdots\right) \\ y\left(1 + \frac{x}{f} + \cdots\right) \\ 1 \end{bmatrix},$$

and the Jacobian is

$$\frac{\partial(\widehat{x}, \widehat{y})}{\partial(x, y)} = \begin{bmatrix} 1 + 2\frac{x}{f} + \cdots & 0 \\ \frac{y}{f} + \cdots & 1 + \frac{x}{f} + \cdots \end{bmatrix}.$$

Since we are interested in the neighborhood of the origin, we can assume that both x and y are small. Then, if x and y are small, the Jacobian reduces to the identity. In other words, in a neighborhood of the origin, G is approximated by the identity mapping (the simplest isometry).

For a point $\underline{x}_0$ chosen away from the origin, and a general epipole $\underline{e}$, we can translate the image with the mapping T so that $\underline{x}_0$ is mapped to the origin, and then rotate the image about the origin with R so that the epipole in mapped to a point $\begin{bmatrix} f & 0 & 1 \end{bmatrix}^T$ on the x-axis. Then we can use the transformation G as given above to map the epipole to a point at infinity. In summary, the transformation in this case is $H = GRT$.

This method is adequate for mapping the epipole in one of the two image to infinity. A similar approach can be followed to transform the other image. The method does, however, require a few "fixes" (to determine R and T as above). Moreover, the two images are transformed separately and, although we obtain horizontal epipolar lines, we still need to relate them. The cameras may have different focal lengths, for example, yielding epipolar lines that are differently spaced in the two images.

11.2 Transforming stereo images to be coplanar

We now provide an alternative method for performing image rectification, that attempts to not only transform the epipolar lines to be horizontal but also such that corresponding epipolar lines are easily identified (in fact, they will have the same image y-coordinates). Contrary to the previous technique, however, this technique does require camera matrices.

Consider two camera matrices

$$P_1 = K_1 R_1 [I \mid -\widetilde{C}_1] \quad \text{and} \quad P_2 = K_2 R_2 [I \mid -\widetilde{C}_2]. \tag{11.3}$$

We can follow the procedure outlined in section 8.2 to decompose these camera matrices and obtain K_1, K_2, R_1, R_2, $\widetilde{C}_1$ and $\widetilde{C}_2$.

Our aim is to projectively transform image pairs captured by these two cameras so that epipolar lines become perfectly horizontal. We will specify a new calibration matrix K_n and a new rotation matrix R_n in such a way that the transformed camera matrices will become

$$P_1' = K_n R_n [I \mid -\widetilde{C}_1] \quad \text{and} \quad P_2' = K_n R_n [I \mid -\widetilde{C}_2]. \tag{11.4}$$

Note that these new camera matrices have the same intrinsic parameters, so that it will be easy to associate horizontal epipolar lines, and they have the same orientation but different camera centres. This is exactly what we want — if the two image planes are coplanar, the epipoles will be at infinity. Figure 11.1 clarifies.

K_n can be chosen arbitrarily, as long as it remains upper-triangular with strictly positive diagonal entries, but a good choice that should minimize image distortion would simply be

$$K_n = \frac{1}{2}(K_1 + K_2). \tag{11.5}$$

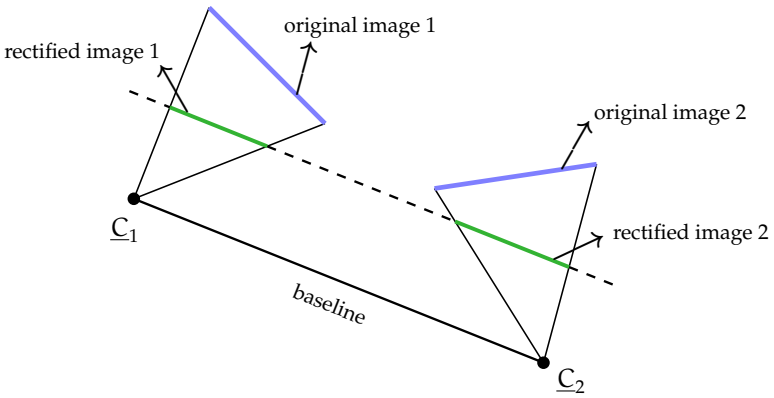


Figure 11.1: A top-down view of what image rectification aims to do.

For R_n we compute the following vectors:

$$\underline{r}_1 = \frac{\widetilde{C}_2 - \widetilde{C}_1}{\|\widetilde{C}_2 - \widetilde{C}_1\|} \tag{11.6}$$

$$\underline{r}_2 = \frac{\underline{k} \times \underline{r}_1}{\|\underline{k} \times \underline{r}_1\|} \quad \text{with } \underline{k} = R_1^T \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \tag{11.7}$$

$$\underline{r}_3 = \underline{r}_1 \times \underline{r}_2, \tag{11.8}$$

and let

$$R_n = \begin{bmatrix} \underline{r}_1^T \\ \underline{r}_2^T \\ \underline{r}_3^T \end{bmatrix}. \tag{11.9}$$

Note that $\underline{r}_1$, $\underline{r}_2$ and $\underline{r}_3$ point in the direction of the transformed camera's x-, y- and z-axes, respectively (why?). In choosing these three vectors as we do, we ensure that the two cameras point in a direction perpendicular to the baseline.

We now define transformation matrices

$$T_1 = K_n R_n R_1^T K_1^{-1} \quad \text{and} \quad T_2 = K_n R_n R_2^T K_2^{-1}. \tag{11.10}$$

We apply the transformation T_1 to image 1 (in the same way as we would apply any 3×3 homography to an image) to obtain its rectified version, and apply the transformation T_2 to image 2. That is to say, if $\underline{x}_1$ is a point in the original image 1, then $\underline{x}_1' = T_1 \underline{x}_1$ is its location in the rectified image 1. Similarly, $\underline{x}_2' = T_2 \underline{x}_2$ is the rectified location of $\underline{x}_2$ in image 2.

Note that, with T_1 and T_2 defined as in (11.10), we do obtain camera matrices of the form (11.4). To see this, suppose $\underline{x}_1$ is the projection onto the first camera's image plane of a world point $\underline{X}$, i.e. $\underline{x} = P_1 \underline{X}$. We transform this image point to $\underline{x}_1' = T_1 \underline{x}_1$, so that

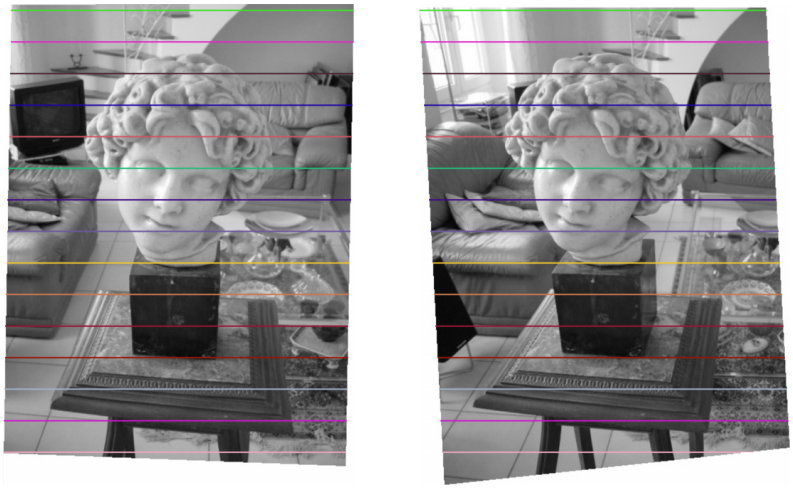
$$\underline{x}_1' = (K_n R_n R_1^T K_1^{-1}) K_1 R_1 [I \mid -\widetilde{C}_1] \underline{X} = K_n R_n [I \mid -\widetilde{C}_1] \underline{X} = P_1' \underline{X}. \tag{11.11}$$

Similarly, if $\underline{X}$ projects to a point $\underline{x}_2$ in the second image plane, its rectified coordinates are given by

$$\underline{x}_2' = (K_n R_n R_2^T K_2^{-1}) K_2 R_2 [I \mid -\widetilde{C}_2] \underline{X} = K_n R_n [I \mid -\widetilde{C}_2] \underline{X} = P_2' \underline{X}. \tag{11.12}$$

Figure 11.2 shows two images of scene captured from slightly different viewpoints. Corresponding epipolar lines in these images are shown in (a), and those after image rectification in (b). Notice that the epipolar lines in the latter are perfectly horizontal. In fact, any feature in the one rectified image will match with a point in the other at the same vertical coordinate of that feature.

(a) original images with epipolar lines



(b) rectified images with epipolar lines

Figure 11.2: An example of stereo image rectification. Corresponding epipolar lines are drawn using similar colours.

11.3 The fundamental matrix for rectified images

All that remains is to calculate the fundamental matrix and the epipolar lines for the rectified images. Let us therefore assume that we have two rectified cameras with matrices

$$P_1' = M[I \mid -\widetilde{\underline{C}}_1] \quad \text{and} \quad P_2' = M[I \mid -\widetilde{\underline{C}}_2], \tag{11.13}$$

where $M = K_n R_n$. We would like to derive an expression for the fundamental matrix F' that corresponds to these two camera matrices.

Recall from section 9.2.1 that

$$F' = [\underline{e}_2']_\times P_2' (P_1')^+, \tag{11.14}$$

where $\underline{e}_2'$ is the epipole in the rectified image 2, and $(P_1')^+$ is the pseudo-inverse of P_1'. In this case a useful expression for the pseudo-inverse is

$$(P_1')^+ = \begin{bmatrix} M^{-1} \\ \underline{0}^T \end{bmatrix}. \tag{11.15}$$

It is straightforward to verify that, with this $(P_1')^+$, we still have that $P_1'(P_1')^+ = I$. Regarding the epipole $\underline{e}_2'$, we see that

$$\begin{aligned} \underline{e}_2' &= P_2'\underline{C}_1 \\ &= K_n R_n \left[I \mid -\widetilde{\underline{C}}_2 \right] \begin{bmatrix} \widetilde{\underline{C}}_1 \\ 1 \end{bmatrix} \\ &= K_n R_n (\widetilde{\underline{C}}_1 - \widetilde{\underline{C}}_2) \\ &= K_n \begin{bmatrix} \underline{r}_1^T \underline{v} \\ \underline{r}_2^T \underline{v} \\ \underline{r}_3^T \underline{v} \end{bmatrix}, \end{aligned}$$

with $\underline{v} = \widetilde{\underline{C}}_1 - \widetilde{\underline{C}}_2$. Since $\underline{r}_1$ was defined as in (11.6), and $\underline{r}_2$ and $\underline{r}_3$ are both perpendicular to $\underline{r}_1$, it follows that $\underline{r}_1^T \underline{v} = -\|\underline{v}\|$ and $\underline{r}_2^T \underline{v} = \underline{r}_3^T \underline{v} = 0$. Hence

$$\underline{e}_2' = K_n \begin{bmatrix} -\|\underline{v}\| \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} \tag{11.16}$$

in homogeneous coordinates. We see that, as expected, the epipole is a point at infinity in the direction of the image x-axis.

Therefore, using (11.14),

$$F' = [\underline{e}_2']_\times M M^{-1} = [\underline{e}_2']_\times = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & -1 \\ 0 & 1 & 0 \end{bmatrix}. \tag{11.17}$$

Corresponding points $\underline{x}_1' = \begin{bmatrix} x_1 & y_1 & 1 \end{bmatrix}^T$ and $\underline{x}_2' = \begin{bmatrix} x_2 & y_2 & 1 \end{bmatrix}^T$ in the two rectified images must satisfy

$$\underline{x}_2'^T F' \underline{x}_1' = 0. \tag{11.18}$$

Written out, we have

$$\begin{bmatrix} x_2 & y_2 & 1 \end{bmatrix} \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & -1 \\ 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ y_1 \\ 1 \end{bmatrix} = 0, \tag{11.19}$$

implying that $y_1 = y_2$. This means that the image coordinates of corresponding points in the rectified image have the same y-coordinates. Thus we note that the choice of rectification was indeed a good one.

Chapter 12

3D reconstruction

Given two calibrated cameras and a pair of corresponding points in the two images, the 3D coordinates of the feature is simply calculated using a triangulation procedure.

12.1 Triangulation

Given two calibrated cameras P and P' as well as a pair of corresponding points $\underline{x} = \begin{bmatrix} x & y & z \end{bmatrix}^T$ and $\underline{x}' = \begin{bmatrix} x' & y' & z' \end{bmatrix}^T$ the idea is to calculate the 3D feature $\underline{X}$ such that

$$\underline{x} = P\underline{X}, \tag{12.1}$$
$$\underline{x}' = P'\underline{X}. \tag{12.2}$$

As before these equations cannot be solved directly for $\underline{X}$ since we are working in homogeneous coordinates, i.e. the equality signs merely indicate that the two sides of the equality sign are vectors pointing in the same direction.

Taking cross products as usual in this situation, e.g. $\underline{x} \times P\underline{X} = \underline{0}$, we get

$$\begin{aligned} y\underline{p}_3^T\underline{X} - z\underline{p}_2^T\underline{X} &= 0 \\ z\underline{p}_1^T\underline{X} - x\underline{p}_3^T\underline{X} &= 0 \\ x\underline{p}_2^T\underline{X} - y\underline{p}_1^T\underline{X} &= 0, \end{aligned}$$

where the $\underline{p}_i^T$ are the rows of P. Taking the first two equations (why not all three?) of the two camera systems, we get a system of the form

$$A\underline{X} = \underline{0}, \tag{12.3}$$

where

$$A = \begin{bmatrix} y\underline{p}_3^T - z\underline{p}_2^T \\ z\underline{p}_1^T - x\underline{p}_3^T \\ y'\underline{p}_3'^T - z'\underline{p}_2'^T \\ z'\underline{p}_1'^T - x'\underline{p}_3'^T \end{bmatrix}.$$

Note that we are measuring the image coordinates directly in Euclidean coordinates, so that $z = 1 = z'$. Thus we again need to solve for the right singular vector associated with the smallest singular value of A.

12.2 Sampson correction

The procedure described above may work quite well. A problem that can potentially cause problems is the fact that two lines in general do not intersect in 3D, see Figure 12.1. Although the triangulation described in the previous section gives a solution, this solution is not projectively invariant, since distance is not an invariant in projective spaces. There is no convenient distance measure that one can minimize.

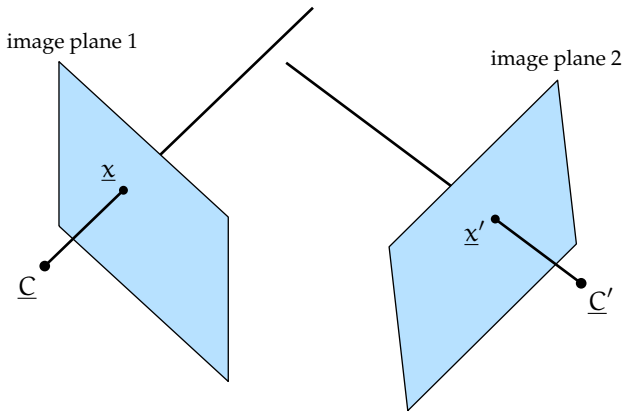


Figure 12.1: Lines in 3D in general do not intersect.

The idea is to minimize the geometric error, as defined on the image planes where one does have a distance measure in Euclidean image coordinates. Thus we are looking for $\hat{\underline{x}}$ and $\hat{\underline{x}}'$ as close as possible to $\underline{x}$ and $\underline{x}'$ respectively, that satisfy the epipolar constraint. Defining the error function

$$E(\hat{\underline{x}}, \hat{\underline{x}}')^2 = d(\underline{x}, \hat{\underline{x}})^2 + d(\underline{x}', \hat{\underline{x}}')^2, \tag{12.4}$$

we minimize $E(\hat{\underline{x}}, \hat{\underline{x}}')^2$ subject to $\hat{\underline{x}}'^T F \hat{\underline{x}} = 0$, see Figure 12.2.

If we now write $\hat{\underline{x}} = \underline{x} + \Delta\underline{x}$ and $\hat{\underline{x}}' = \underline{x}' + \Delta\underline{x}'$, the problem can be rewritten as

$$\text{minimize} \quad \|\Delta\underline{x}\|^2 + \|\Delta\underline{x}'\|^2 \tag{12.5}$$

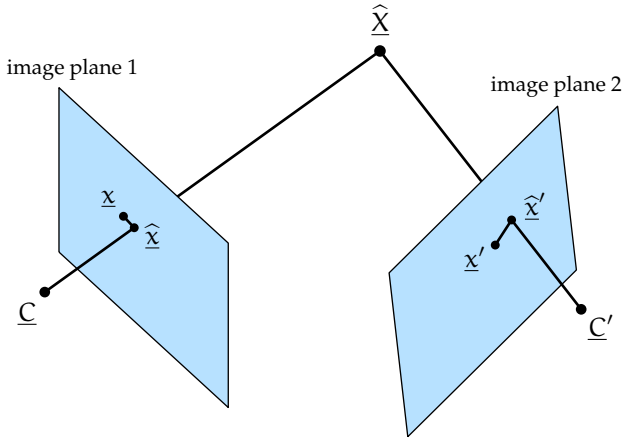


Figure 12.2: Minimizing the geometric error and forcing the lines to intersect.

subject to

$$(\underline{x}' + \Delta\underline{x}')^T F(\underline{x} + \Delta\underline{x}) = 0. \tag{12.6}$$

Keeping only first order quantities the constraint is approximated by

$$\underline{x}'^T F\underline{x} + \Delta\underline{x}'^T F\underline{x} + \underline{x}' F\Delta\underline{x} = 0. \tag{12.7}$$

Now we have to keep in mind that we work in Euclidean image coordinates, i.e. we correct only the x and y Euclidean image coordinates so that

$$\Delta\underline{x} = \begin{bmatrix} \Delta x \\ \Delta y \\ 0 \end{bmatrix}, \tag{12.8}$$

and similarly for $\Delta\underline{x}'$. If we now write

$$\delta\underline{x} = \begin{bmatrix} \Delta x \\ \Delta y \\ \Delta x' \\ \Delta y' \end{bmatrix}, \tag{12.9}$$

the constraint can be written as

$$J^T \delta\underline{x} = -\underline{x}'^T F\underline{x}, \tag{12.10}$$

where J^T is the Jacobian row matrix

$$J^T = \begin{bmatrix} (F^T\underline{x}')_1 & (F^T\underline{x}')_2 & (F\underline{x})_1 & (F\underline{x})_2 \end{bmatrix} \tag{12.11}$$

with $(F\underline{x})_j$ the j-th entry of the vector $F\underline{x}$ (and similarly for $(F^T\underline{x}')_j$).

We can now reformulate the problem as: solve

$$J^T \delta\underline{x} = -\underline{x}'^T F\underline{x}, \tag{12.12}$$

with $\|\delta\underline{x}\|$ as small as possible. But this is just the generalized solution of the system, i.e.

$$\delta\underline{x} = -J(J^T J)^{-1}\underline{x}'^T F\underline{x}. \tag{12.13}$$

Written out in full this gives

$$\begin{bmatrix} \widehat{x} \\ \widehat{y} \\ \widehat{x}' \\ \widehat{y}' \end{bmatrix} = \begin{bmatrix} x \\ y \\ x' \\ y' \end{bmatrix} - \frac{\underline{x}'^T F\underline{x}}{(F\underline{x})_1^2 + (F\underline{x})_2^2 + (F^T\underline{x}')_1^2 + (F^T\underline{x}')_2^2} \begin{bmatrix} (F^T\underline{x}')_1 \\ (F^T\underline{x}')_2 \\ (F\underline{x})_1 \\ (F\underline{x})_2 \end{bmatrix}. \tag{12.14}$$

Thus, given a pair of corresponding points $\underline{x}$ and $\underline{x}'$ in Euclidean coordinates, we first calculate the Sampson correction (12.14), and then do the triangulation (12.3).