Chapter 3

Digital image acquisition

3.1 Digital sampling

In chapter 1 we already mentioned that when acquiring an image of a real scene it is discretized in two ways: sampling and quantization. Figure 3.1 shows sampling and quantization of a one-dimensional signal on a uniform grid\(^1\). The signal is sampled at ten positions \((x = 0, \ldots, 9)\), and each sampled value is then quantized to one of seven levels \((y = 0, \ldots, 6)\). Sampling and quantization of images is done in exactly the same way, except that sampling is now done in more than one dimension. An example is shown in figure 3.2. Here, we see a continuous signal of two variables (top left), and the corresponding image with signal strength converted to grey values (bottom left). On the right, the signal is shown after sampling on a discrete grid and quantization of the grey values onto five levels.

\(^{1}\)Note: even though this is a book about image processing, we will often use examples taken from one-dimensional signal processing wherever this simplifies matters.
By simply looking at figure 3.2, it is clear that the digital picture is not a very good representation of the original. Obviously the density of the sampling grid and the number of levels we choose in the quantization are important quality factors. These factors are called resolution: the spatial resolution equals the number of pixels used, and the intensity resolution equals the number of grey levels used. In digital images, both types of resolution are finite. The effect of lowering the spatial resolution can be seen in figure 3.3.

There is another kind of resolution called the optical resolution, which is the smallest spatial detail a visual system can see. Optical and spatial resolution are often used without their adjectives in texts, and what type of resolution is meant should be determined from the context.
Figure 3.3 Effect of lowering the spatial resolution. From top to bottom and from left to right: original image with a resolution of $256 \times 256$ pixels, then resolutions $128 \times 128, 64 \times 64, 32 \times 32, 16 \times 16,$ and $8 \times 8.$
3.1.1 The Nyquist frequency

If we sample an image, is it possible to make the image exactly the same as the original? And if so, how many samples are necessary? Before we can answer these questions, we need the concept of spatial frequency. The frequency of a sinusoid is defined as the number of cycles it makes per unit length. For example, the family of functions $f(x) = \sin(ax)$ has frequency $\frac{a}{2\pi}$. Figure 3.4 shows three sinusoids of different frequency and the corresponding image. There is a relationship between spatial frequency and the level of detail in images: a high level of detail corresponds to a high frequency, and a lack of details corresponds to low frequencies. A high level of detail is made up of relatively large contrast, i.e., the image intensity cycles fast from dark to light and vice versa, just like a high frequency sinusoid does. This relationship between frequency and image detail is at the heart of many image processing applications and will be explored further in the chapter on the Fourier transform.

We already noted that, in a digital image, the spatial and intensity resolution are finite. Hence the finest detail that can be represented is limited, and the range of spatial frequencies occurring in the image is also limited. This implies that the number of samples we must take of an image in order to exactly reproduce all the details is also finite. This observation is linked to spatial frequency by the sampling theorem:

To capture the highest frequencies (i.e., the smallest details) of a continuous image, the sampling rate must be $2\xi_n$ or faster, where $\xi_n$ equals the highest frequency occurring in the original image. $2\xi_n$ is called the Nyquist frequency.

---

$\xi_n$ is called the Nyquist frequency.

2 Many books use “band” terms when discussing frequencies. If the frequency has a limited range, it is called bandlimited. The range of frequency values itself is called the bandwidth.
If we sample an image below the Nyquist frequency, details will be lost. Oversampling, on the other hand, will not increase the level of detail: all of the details are already captured when sampling exactly at the Nyquist frequency.

Example

Consider the following line taken from an image:

The pattern has a frequency of $\frac{1}{2}$ per pixel – i.e., there are 2 pixels in a pattern cycle– so the Nyquist frequency is $2 \cdot \frac{1}{2} = 1$ per pixel. In the above, we have sampled the image line with this frequency, and, obviously, the result equals the original. If we lower the sampling frequency to $\frac{4}{5}$ per pixel – i.e., 4 samples in 5 pixels– details are lost:

If we lower the sampling frequency even further to $\frac{1}{3}$ per pixel, a pattern emerges that wasn’t even present in the original image:

3.1.2 Aliasing

The phenomenon occurring at the end of the example above is called aliasing: if we sample an image at a rate that is below half the Nyquist frequency, high frequency patterns in the original are translated to lower frequency patterns that are absent in the original image. In the example above, we see that a pattern with frequency $\frac{1}{2}$ per pixel is mapped to a pattern with frequency $\frac{1}{6}$ per pixel.
The examples above are extreme cases of aliasing that are unlikely to occur in practice. Aliasing pops up as an artifact with many imaging applications, however. Telltales are the occurrence of Moiré or repetitive patterns in images. A notorious example is the sampling of printed images, e.g., a picture from a newspaper, where the sampling frequency must be carefully adapted to the frequency used in the printing process, or low frequency artifacts will occur. Aliasing effects may not always be visible to the human eye, but still prove to be disastrous to a computer image processing application. It is therefore wise to check the original and sampling frequencies against possible aliasing.

### 3.1.3 Sampling and convolution

Until now we have assumed that we can sample an image with infinite precision. In practice this is an impossibility, since practical imaging devices cannot have an infinite precision. For example: if we use a camera to take samples of some infinitely small image location \((x, y)\), then the camera will gather its sample in a small neighborhood around \((x, y)\). There is in fact another reason why sampling with infinite precision is impossible: cameras need the energy from incoming light rays to detect these rays. Even if it were possible to point the camera with infinite precision at only the infinitely small point \((x, y)\), the energy of the incoming light rays would be reduced to zero, and we would detect nothing. There is a tradeoff here: on the one hand we would like our sampling to be as precise as possible, so a small sampling area, but on the other hand the sampling area cannot be too small, or it will not give off enough energy for proper detection.

**Example**

The diaphragm of a photocamera controls the size of the sampling area on the photosensitive film. The smaller the diaphragm, the smaller the sampling area, and the sharper the picture will be. However, if we make the diaphragm too small,
not enough light will be able to pass through the diaphragm to light the film properly. In practice, the photographer will choose the smallest diaphragm possible considering the ambient light.

To keep the diaphragm as small as possible, a photographer will often try to increase the energy that reaches the film by other means, for instance by extra lighting—using lamps or a flashlight— or increasing the exposure time of the film.

An important tool in describing the sampling process is mathematical convolution:

The convolution $f * g$ of two integrable functions $f, g : \mathbb{R} \rightarrow \mathbb{R}$, is defined as:

$$(f * g)(x) = \int_{-\infty}^{\infty} f(a)g(x - a) \, da.$$ 

If $f$ and $g$ are two-dimensional images, convolution is defined as

$$(f * g)(x, y) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} f(a, b)g(x - a, y - b) \, da \, db,$$

and so on for more dimensions. Convolution is a commutative operator, i.e., $f * g = g * f$.

**Example**

To understand convolution, we will work out a one-dimensional example. Suppose $f$ is some signal, for instance:

$$f(x) = \begin{cases} 
\text{some signal} 
\end{cases}$$
We define \( g(x) \) by

\[
g(x) = \begin{cases} 
  1/\sigma & \text{if } |x| \leq \sigma/2 \\
  0 & \text{if } |x| > \sigma/2 
\end{cases}
\]

In this example, the convolution equals:

\[
(g * f)(x) = \int_{-\infty}^{\infty} g(a) f(x - a) \, da = \frac{1}{\sigma} \int_{-\sigma/2}^{\sigma/2} f(x - a) \, da.
\] (3.1)

The effect of this integral is to take the average of all the values of \( f(x) \) in a neighborhood of size \( \sigma \) around \( x \). The effect of this “local averaging” is that the values of \( f \) get “smeared out” over a certain region. Figure 3.6 shows \( f * g \) for some values of \( \sigma \). The effect of the convolution is that \( f \) gets smoothed: small details of \( f \) vanish\(^3\). The larger we choose \( \sigma \) the more details of \( f \) vanish.

In cases like in the above example, where \( f \) is a signal or image, and \( g \) is a function with a narrow support (i.e., has significant values only in a small region), we call \( g \) the convolution kernel. Note that the area under the curve of \( g \) in this example is always 1.

If \( \sigma \) tends to zero, \( g \) in the example above transforms into a peculiar function that has value zero everywhere, except at \( x = 0 \), where there is a spike whose value tends to infinity. The area under the curve of \( g \) is still 1, however. This special function is known as the Dirac or delta function \( \delta(x) \):

\[
\delta(x) = \begin{cases} 
  \infty & \forall x \in \mathbb{R} \setminus 0 \\
  \int_{-\infty}^{\infty} \delta(x) \, dx = 1.
\end{cases}
\]

The delta function has a special property, called the sifting property:

\[
(\delta * f)(x) \overset{eq.3.1}{=} \lim_{\sigma \to 0} \frac{1}{\sigma} \int_{-\sigma/2}^{\sigma/2} f(x - a) \, da = f(x),
\]

\(^3\)Since small details correspond to high frequencies, we call this type of convolution a low pass filter, because it lets only low frequencies pass, and filters away high frequencies.
3.1 Digital sampling

Figure 3.5 Example of aliasing. The same two-dimensional function was sampled four times: using $400 \times 400$ points (top left), $100 \times 100$ points (top right), $50 \times 50$ points (bottom left), and $25 \times 25$ points (bottom right).

Figure 3.6 Examples of convolution of a signal with a rectangular convolution kernel (see text for details). Left: original signal, middle: convolution with $\sigma = 0.25$, right: convolution with $\sigma = 0.6$. 
so convolving with a delta function does not alter the function \( f \). This was to be expected, since we said convolving \( f \) with our choice of kernel \( g \) results in smoothing of \( f \). The larger we choose \( \sigma \), the more \( f \) gets smoothed. So by the same argument, the smaller we choose \( \sigma \), the more the convolution result resembles \( f \). If \( \sigma \) tends to zero, \( g \) transforms to the delta function, and \( f \) does not change at all.

We mentioned earlier that imaging devices do not have infinite precision, so sampling a signal at a certain location is done by taking a small neighborhood around that location into account. But this is exactly what convolving \( f \) with some convolution kernel \( g \) does! We now have all the ingredients we need to describe the sampling process in a mathematical way:

The value of a sample is the result of the convolution of the unknown signal or image \( f \) with some convolution kernel \( g \). The kernel \( g \) depends only on the imaging device. In the ideal case, the kernel \( g \) is infinitely narrow and equals the delta function \( \delta \).

In practice, we are usually able to determine the specific kernel \( g \) of imaging devices, so we are able to determine its behavior when sampling a variety of input images \( f \).

The convolution effect of the rectangular kernel \( g \) in our example is that the image \( f \) gets smoothed; each value \( f(x) \) was replaced by the local average of values in a neighborhood of size \( \sigma \). It is often easy to predict the convolution result from the shape of the kernel, see table 3.1 for examples.

The last two kernels in table 3.1 have a realistic appeal: like the rectangular kernel they compute local averages of \( f(x) \), but in a weighted way: values close to \( x \) have a higher weight than values further away. This is what we expect of imaging systems like cameras: the sampled value is an average of values in a small region, but the center of this region has a larger impact on the result than the edges have.

The last kernel in the table is called the Gaussian function, \( g(x) = \frac{1}{\sigma \sqrt{2\pi}} e^{-\frac{1}{2} \left(\frac{x}{\sigma}\right)^2} \), which is especially useful in modeling realistic kernels for many imaging devices. It also has a number of interesting theoretical properties that we will come back to later in this book.

### 3.1.4 Extension to two dimensions

In the previous section we have seen examples of one-dimensional kernels and their effect on one-dimensional signals. We will now show a few examples of two-dimensional kernels, \textit{i.e.}, kernels working on two-dimensional images. In the one-dimensional case we found that the area under the kernel equals one. This also holds in the two-dimensional case, or rather, the \textit{volume} under the kernel equals one. In general, integrating
the kernel over all of its variables should result in one. Often, two-dimensional kernels are symmetrical in their variables, i.e., swapping the variables will not alter the kernel formula. Many imaging systems have this property.

For example, the symmetrical two-dimensional version of the rectangular kernel (first kernel in table 3.1) would be

\[
g(x, y) = \begin{cases} 
\frac{1}{\sigma^2} & \text{if } |x| \leq \sigma/2 \text{ and } |y| \leq \sigma/2 \\
0 & \text{otherwise}
\end{cases}
\]

Further examples: the equivalent of the ideal sampling function \( \delta(x) \) in two dimensions is \( g(x, y) = \delta(x)\delta(y) \), and the equivalent of the Gaussian kernel \( \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2} \left( \frac{x^2}{\sigma^2} \right)} \) is \( g(x, y) = \frac{1}{\sigma^2\sqrt{2\pi}} e^{-\frac{1}{2} \left( \frac{x^2+y^2}{\sigma^2} \right)} \).

In all cases, it can be verified that the volume under the kernel equals one:

\[
\int_{-\infty}^{\infty} \int_{-\infty}^{\infty} g(x, y) \, dx \, dy = 1.
\]

Figure 3.7 shows an example of a two-dimensional Gaussian kernel.

![Example of a two-dimensional Gaussian kernel with \( \sigma = 0.7 \).](image)

### 3.1.5 The point spread function

To find out what the convolution kernel of an imaging system looks like, we often examine the point spread function (PSF) of the system\(^4\). The PSF is defined as the image produced by the system if we present it with a delta function (a “point”). In terms of

\(^4\)We can often assume the PSF and the convolution kernel of the system to be identical.
images, this means that we expose our camera to an image that is fully dark everywhere, except at one infinitely small spot, where the luminance is infinite. In practice, we cannot produce such an image, so we use the next best thing, which is an image that is dark everywhere, but has the highest luminance we can achieve at a very small spot. Figure 3.8 illustrates this.

![Figure 3.8 Establishing the PSF of an imaging system. Left column: the delta function or its practical approximation. Right column: the resultant PSF.](image)

The broader the PSF, the more the original image will be smoothed by the imaging system. Figure 3.9 shows an example of images that have been smoothed using a Gaussian convolution kernel.

### 3.2 Quantization

After sampling, the image is quantized using a finite number of grey values. With most imaging systems, the quantizer is a digital circuit that converts the sampled value to an $n$-bit word. Since an $n$-bit word can take $2^n$ different values, the number of possible grey values equals $2^n$. Figure 3.10 illustrates this. This figure also shows the effect of using different values of $n$ on the grey scale: the larger $n$ is, the smoother the grey value transitions are. Figure 3.11 shows the effects of reducing $n$ on a real image. In practice, $n$
3.3 Color models

Up until now, we associated a single value with each pixel, that represented the pixel’s grey value (luminance). But how can we represent color images? We have already seen in chapter 1 that we can reproduce any color using a mixture of the primary colors red, green, and blue. And indeed the most common way to represent a color image is to

Figure 3.9 Example of smoothing of an image with a Gaussian PSF. Original image with a resolution of $256 \times 256$ pixels (left), after convolution with a Gaussian with $\sigma = 1$ pixel (middle), and $\sigma = 4$ pixels (right).

varies from 1 to about 16. For reproducing grey valued images intended for humans, a value of $n = 8$ (256 grey values) is usually sufficient. Medical scanners like a CT scanner have 12 or more bits of quantization.

Although a large $n$ will give us a smooth grey scale, it is not always useful to make $n$ larger. For one thing, a larger $n$ means that quantizing a sample will take more time. In many cases, the effects of a slow quantization with a large $n$ are worse than using a fast quantization with a smaller $n$. Another reason is that increasing $n$ only makes sense if the original signal has enough detail to merit this. For example: suppose we have a grey valued image with 256 grey values (i.e., $n = 8$), then it would be pointless to scan this image with $n > 8$. In practice, there is always an upper bound for $n$ above which it is pointless to acquire an image.

The number of bits $n$ used in the quantization can only be used to determine the number of grey levels that can be encoded using those bits. The actual grey value also depends on the encoding scheme used. For example, if we use $n = 32$ (four bytes), we can use an “integer” scheme to encode values in the range $\{-2^{31}, \ldots, 2^{31} - 1\}$, an “unsigned integer” scheme to encode values in the range $\{0, \ldots, 2^{32} - 1\}$, or a “floating point” scheme to encode real numbers in an implementation dependent range.
associate *three* values with every pixel, namely the red (700 nm), green (546.1 nm), and blue component (435.8 nm) of the pixel’s color. This color model is called the *RGB* model for color representation.

**Example**

Below are the red, green, and blue color components (each scaled to be between 0 and 1) of a single $3 \times 5$ color image:

<table>
<thead>
<tr>
<th>R</th>
<th>G</th>
<th>B</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 0 0</td>
<td>0 1 0</td>
<td>0 0 1</td>
</tr>
<tr>
<td>0.1 0.5 0.9</td>
<td>0.1 0.5 0.9</td>
<td>0.1 0.5 0.9</td>
</tr>
<tr>
<td>0 1 1</td>
<td>0 1 1</td>
<td>0 1 1</td>
</tr>
<tr>
<td>0.1 0.5 1</td>
<td>0.1 0.5 1</td>
<td>0 0 0</td>
</tr>
<tr>
<td>0 0 0</td>
<td>0.1 0.5 1</td>
<td>0.1 0.5 1</td>
</tr>
</tbody>
</table>

And the composite color image will look like: (note that equal values of the R, G, and B components will result in a shade of grey)

<table>
<thead>
<tr>
<th>pure red</th>
<th>pure green</th>
<th>pure blue</th>
</tr>
</thead>
<tbody>
<tr>
<td>dark grey</td>
<td>medium grey</td>
<td>light grey</td>
</tr>
<tr>
<td>black</td>
<td>pure white</td>
<td>pure white</td>
</tr>
<tr>
<td>dark yellow</td>
<td>medium yellow</td>
<td>pure yellow</td>
</tr>
<tr>
<td>dark cyan</td>
<td>medium cyan</td>
<td>pure cyan</td>
</tr>
</tbody>
</table>
The colors red, green, and blue are called the primary colors of the RGB model. Their mixtures cyan (blue mixed with green), magenta (red and blue), and yellow (red and green) are called the secondary colors.

Intermezzo

The RGB model is an additive model: pure red, green, blue mixed together form white. Additive models are logical models to use in case of computer or TV screens, or any device that emits light. They cannot be used in cases where we use reflected light. For example: if we mix paint, the combination of red, green, and blue will be black instead of white. The reason for this is that, e.g., green paint looks green to us because it only reflects the green part of the spectrum, and does not reflect the red or blue parts. Similarly, red paint does not reflect the blue or green parts of the spectrum. Yellow paint reflects the green and red parts, but stops the blue part. We see that the model here is subtractive rather than additive: yellow paint subtracts the blue part from white light, and cyan paint subtracts the red part from white light, so if we mix yellow and cyan, we subtract both the blue and red parts, and end up with green paint. The most common subtractive model is the CMY model (cyan, magenta, and yellow). The RGB additive model and CMY subtractive model are shown in figure 3.12. Note that the primary colors in the RGB model are secondary in the CMY model and vice versa.

The CMY model is commonly used in printing: by mixing C, M, and Y inks any color can be created. In practice, however, creating black by mixing C, M, and Y is tricky (very hard to balance right), and expensive, so a fourth, black, ink is used. This four color printing model is called the CMYK model.

Besides color models whose components describe the value of a specific primary color, there are models that have three entirely different variables: the most common amongst these is the HSV, or hue, saturation, value model, see figure 3.13. The hue identifies the color itself. Hue is usually represented as an angle in a color circle, where \( H = 0^\circ \) corresponds to red, \( H = 120^\circ \) to green, \( H = 240^\circ \) to blue, and all other colors are embedded additively between them like a rainbow. The saturation identifies how pure the color component is: a fully saturated color \( (S = 1) \) is a pure color, and if the saturation is zero \( (S = 0) \), the color component is fully absent, and we just have a shade of grey: the dotted line \( S = 0 \) in the figure represents the “greys”. The value refers to the luminance value of the color.

The figure shows the peculiar double-cone structure of HSV-space: there is a single point “black” at the bottom, and “white” at the top. At a first glance, we may expect the \( H, S, \) and \( V \) coordinates to describe a cylindrical space, but it turns out that values
outside of the double cone are impossible, or simply map to the same color as a point on the edge of a cone. The bottom cone degenerates to a single point “black” simply because any point with a luminance value $V = 0$ is black, no matter what the color or saturation components are. The top cone degenerates to a single point “white” because to reach maximum luminance on, e.g., a computer screen we need all three color components (the red, green, and blue dots on the screen) to radiate at maximum luminance, hence the point with maximum luminance value $V$ is always white.

There are many other color models, most of which are simple variants of the RGB, CMY, or HSV models. The models described here are the most common in image processing.
3.3 Color models

<table>
<thead>
<tr>
<th>Convolution kernel $g$</th>
<th>Graph of $g$</th>
<th>Convolution effect on $f$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\delta(x)$</td>
<td></td>
<td>$f(x)$ unaltered (ideal sampling)</td>
</tr>
<tr>
<td>$\frac{1}{2}(\delta(x + 1) + \delta(x - 1))$</td>
<td></td>
<td>$f(x)$ replaced by the average of $f(x - 1)$ and $f(x + 1)$</td>
</tr>
</tbody>
</table>
| $\left\{ \begin{array}{ll}
\frac{1}{\sigma} & \text{if } |x| \leq \sigma/2 \\
0 & \text{if } |x| > \sigma/2
\end{array} \right.$ | | $f(x)$ replaced by local average of $f$ in a neighborhood of size $\sigma$ |
| $\frac{1}{2\pi}e^{-\frac{1}{2}(\frac{x}{\sigma})^2}$ | | $f(x)$ replaced by weighted average of $f$ in a neighborhood of size $\sigma$; values closer to $x$ have higher weight |

Table 3.1 Some examples of convolution kernels and their effects.
Figure 3.11 Effect of reducing the quantization level on a real image. From left to right and top to bottom: original image with $n = 8$ (256 grey levels), $n = 3$ (8 levels), $n = 2$ (4 levels), and $n = 1$ (2 levels).
3.3 Color models

Figure 3.12 The additive (emissive) RGB color model (left), and the subtractive (reflective) CMY color model (right).

Figure 3.13 The HSV (hue, saturation, value) color model. Black, white, the pure colors red (R), green (G), blue (B), and the line of greys are indicated. A point in HSV-space is identified by three cylindrical coordinates: height $V$, angle $H$, and radius $S$. 