1

Introduction

What is Texture?

Texture is the variation of data at scales smaller than the scales of interest. For example, in Figure 1.1 we show the image of a person wearing a Hawaiian shirt. If we are interested in identifying the person, the pattern on the shirt is considered as texture. If we are interested in identifying a flower or a bird on the shirt, each flower or bird of the pattern is a non-textured object, at the scale of this image, as we can hardly see any detail inside it.

Why are we interested in texture?

We are interested in texture for two reasons.

- Texture may be a nuisance in an automatic vision system. For example, if we were to recognise an object from its shape, texture would create extra lines in the edge map of the object and the shape recognition algorithm would be confused. This is demonstrated in Figure 1.2.
- Texture may be an important cue in object recognition as it tells us something about the material from which the object is made. For example, in the image of Figure 1.3 we may discriminate the city from the woods and the fields using the type of variation the image shows at scales smaller than the objects we are talking about.

How do we cope with texture when texture is a nuisance?

There are algorithms that can isolate the texture regions irrespective of the type of texture. The texture regions then may be identified and treated separately from the rest of the image. Since texture manifests itself as image intensity variation, it gives rise to many edges in the edge map of the image. A trivial algorithm that can isolate texture regions is to consider a scanning window and count inside the window the number of edgels. Any window with number of edgels higher than a certain threshold is considered as belonging to a textured region. Figure 1.4 demonstrates the result of this algorithm applied to the image of Figure 1.3. In Box 1.1 a more sophisticated algorithm for the same purpose is presented.

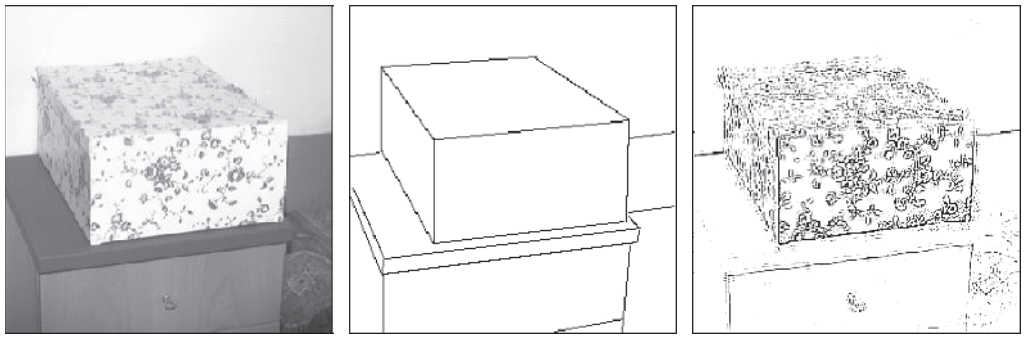
This algorithm is as follows.

Step 1: Perform edge detection in the image to produce edge map A, where all edgels are marked with value 1 and the non-edgels with value 0.

Image Processing: Dealing with Texture, Second Edition. Maria Petrou and Sei-ichiro Kamata.

Companion Website: www.wiley.com/go/kamataText2

Figure 1.1 Costas in bloom. Source: Maria Petrou.



(a) (b) (c)

Figure 1.2 (a) An original image. Source: Maria Petrou. (b) Manually extracted edge map. (c) The automatic edge extraction algorithm is confused by the presence of texture on the box.

Step 2: Create an output array B, the same size as the edge map A, and set all its elements equal to 1 or 255 (all white).

Step 3: Consider an $M \times M$ window, where M could be, for example, 15, 21, etc., depending on the resolution you desire. Select also a threshold T, such that $0 << T < M^2$.

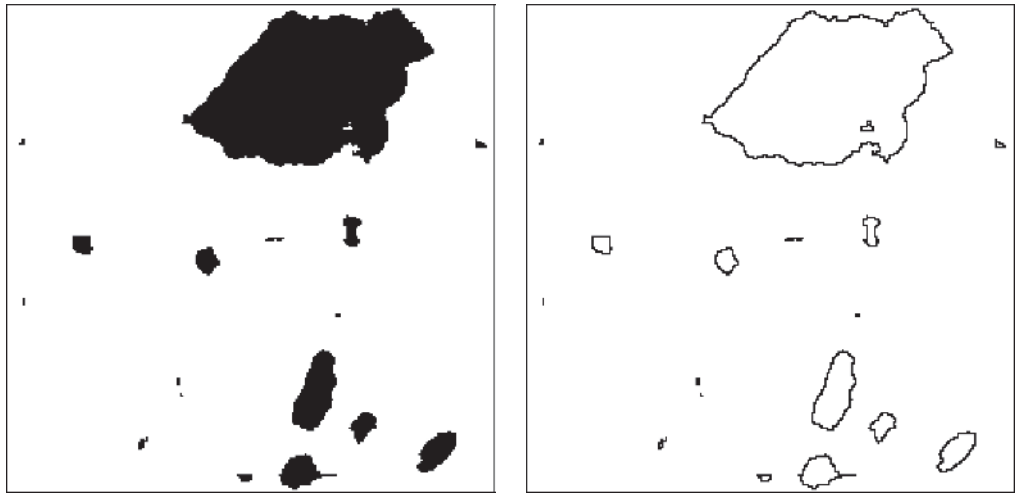
Step 4: Scan A with this window and each time sum up the values of edge map A covered by the window. Assign the output to the central pixel of the window in the output array B. Note that B may be obtained by convolving edge map A with a flat window of size $M \times M$, i.e. a window with all its weights equal to 1.

Step 5: Threshold B, so that if one of its elements has value below T becomes white, while if it has a value above or equal to T it becomes black (value 0). Call this output array C. This way you will obtain an output like that of Figure 1.4a.

Step 6: Consider each black pixel in C, and examine its 3×3 neighbourhood. If even one of those neighbours has value white, keep this as black pixel. If all its neighbours are black, convert it

(a) (b)

Figure 1.3 (a) Blewbury from an aeroplane (size 256 × 256). Source: Maria Petrou. (b) Edge map where the urban area has been annotated. Other texture patches correspond to woods.



(a) (b)

Figure 1.4 (a) When a scanning window of size 17 × 17 was placed in Figure 1.3b with its centre at a pixel marked here as black, the number of edgels inside the window was more than 100. (b) Boundaries of the black regions in (a).

into white. Note that this may be achieved by convolving B with a flat (all elements equal to 1) 3×3 window. If the result of the convolution for a pixel is more than 0, the value of the pixel in the output array should be 0; otherwise it should be 1. (This assumes that the white pixels have value 1.) This way, you will produce a result corresponding to Figure 1.4b.

How does texture give us information about the material of the imaged object?

There are two ways by which texture may arise in **optical images**:

- Texture may be the result of variation of the **albedo** of the imaged surface. For example, the image in Figure 1.2a shows the surface of a box on which a coloured pattern is printed. The change of colours creates variation in the brightness of the image at scales smaller than the size of the box, which on this occasion is the object of interest.
- Texture may be the result of variation of the **shape** of the imaged surface. If a surface is rough, even if it is uniformly coloured, its image will be textured due to the interplay of shadows and illuminated parts. The textures in the image of Figure 1.3 are the result of the roughness of the surfaces of the town and the woods when seen from above.

Are there non-optical images?

Yes, there are. They are the images created by devices that measure something other than the intensity of reflected light. For example, the grey value in **magnetic resonance images (MRI)**, used in medicine, may indicate the proton density of the imaged tissue. Another example concerns images called **seismic sections**, which are created by plotting data collected by seismographs. In this case, the grey value represents the intensity of the reflected seismic wave by the different rocks that make up the crust of the Earth.

What is the meaning of texture in non-optical images?

Texture in non-optical images indicates variation of a certain physical quantity at scales smaller than the scales of interest. For example, in a proton weighted MRI image of the brain, the presence of texture indicates variation of the proton density from one location to the next. Such variation will manifest itself in the image in the same way as variation of the albedo or surface roughness does in an optical image. From the image processing point of view, texture is treated in the same way in optical and non-optical images. So, in this book we shall talk about images irrespective of the sensor with which they have been captured.

What is the albedo of a surface?

The albedo of a surface is a function that characterises the material from which the surface is made. It gives the fraction of incident light this material reflects at each wavelength. When we say that a surface is multicoloured, effectively we say that it has an albedo that varies from location to location.

Can a surface with variable albedo appear non-textured?

Yes, if it is imaged by a single band sensor and the intensity of reflected light by the different coloured patches within the sensor sensitivity band yields constant accumulated recordings.

Can a rough surface of uniform albedo appear non-textured?

Yes, if the roughness of the surface and the incident light are such that no part of the surface is in shadow, all parts of the surface receive the same amount of light and the surface is mat (i.e. the material it is made from reflects equally in all directions, irrespective of the direction from which it is viewed). An example is a rough mat surface seen under totally diffuse ambient light.

Figure 1.5 A surface imaged from two different distances may create two very different image textures. Source: Maria Petrou.

What are the problems of texture that image processing is trying to solve?

Image processing is trying to solve three major problems concerned with texture.

- Identify what a surface represents by analysing the texture in an image of a surface sample: this is the problem of **texture classification**.
- Divide an image into regions of different textures: this is the problem of **texture segmentation**.
- Decide whether a texture is as it is expected to be or contains faults: this is the problem of **texture defect detection**.
- Create a new texture or repair a damaged texture.

What are the challenges in trying to solve the above problems?

The most important challenge is trying to infer information concerning the surface (i.e. its roughness (surface relief) and its albedo) by studying the texture of the image of the surface. The problem is that the same surface may appear totally differently textured under different imaging conditions. Figure 1.5 shows the same surface imaged from different distances. Its appearance changes dramatically and pure image processing without any extra information would never recognise these two images as depicting the same object. Another example is shown in Figure 1.6 where the same surface is imaged from the same distance, but with the illuminating source being at two different positions. Again, pure image processing would never recognise these two images as depicting the same object. These two examples illustrate two important properties of texture:

- Texture is scale-dependent
- Image texture depends on the imaging geometry.

How may the limitations of image processing be overcome for recognising textures?

Image processing that makes use of additional information, e.g. information concerning the imaging geometry, may be able to deal with both the scale dependence of texture as well as its dependence on the imaging geometry. An alternative way is to use computer vision techniques that allow one to recover the missing information suppressed by the imaging process. For example, a rough surface exists in 3D. However, when it is imaged on a 2D medium, the information concerning the third dimension is lost. This information is independent of the imaging process. If one could recover it from the available image, then one would have had a scale-invariant description of the

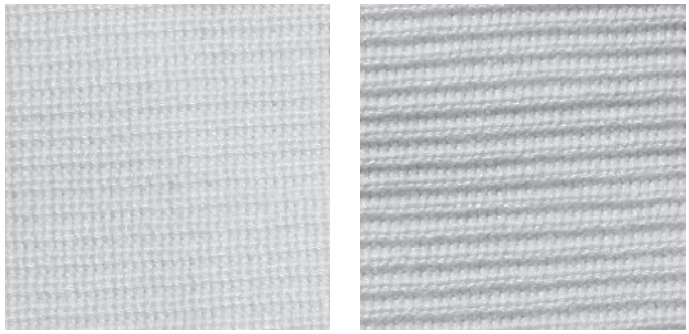


Figure 1.6 A surface imaged under different illumination directions may give rise to two very different image textures. Source: Maria Petrou.

roughness of the surface. The recovery of such information is possible if one uses extra information, or more than one image.

What is this book about?

This book is about the methods used to deal mainly with the first two problems related to texture, namely texture classification and texture segmentation. However, we shall also consider the problem of **in-painting**, i.e. the problem of repairing damaged texture. The various methods will be presented from the point of view of the data rather than the approach used; we shall start from the simplest image textures, i.e. 2D binary images, and proceed to composite grey texture images.

Box 1.1 An algorithm for the isolation of textured regions

Texture in an image implies intensity variation at scales smaller than the image size. It is well known that intensity variation gives rise to edgels in the edge map of the image. A high density of edgels, therefore, characterises textured regions.

Let us imagine that each edgel is a person standing in a flat landscape. The person sees the world around them through the other individuals who might surround them and obscure their view. Suppose that we define the field of view of each individual as the maximum angle through which they can see the world through their neighbours who stand within a certain distance from them. Individuals who are standing in the middle of a textured region will have their views severely restricted and therefore their fields of view will not have large values. On the other hand, individuals standing at the borders of textured regions will be able to see well towards the outside of the textured region and they will enjoy high values of field of view.

The trouble is that even edge pixels that simply separate uniformly coloured regions will enjoy equally high values of the field of view. So, we need to devise an algorithm that will distinguish these two types of edgel: those that delineate textured regions and those that simply form edges between different uniformly shaded regions.

The following are the steps of such an algorithm demonstrated with the help of the image of Figure 1.7a.

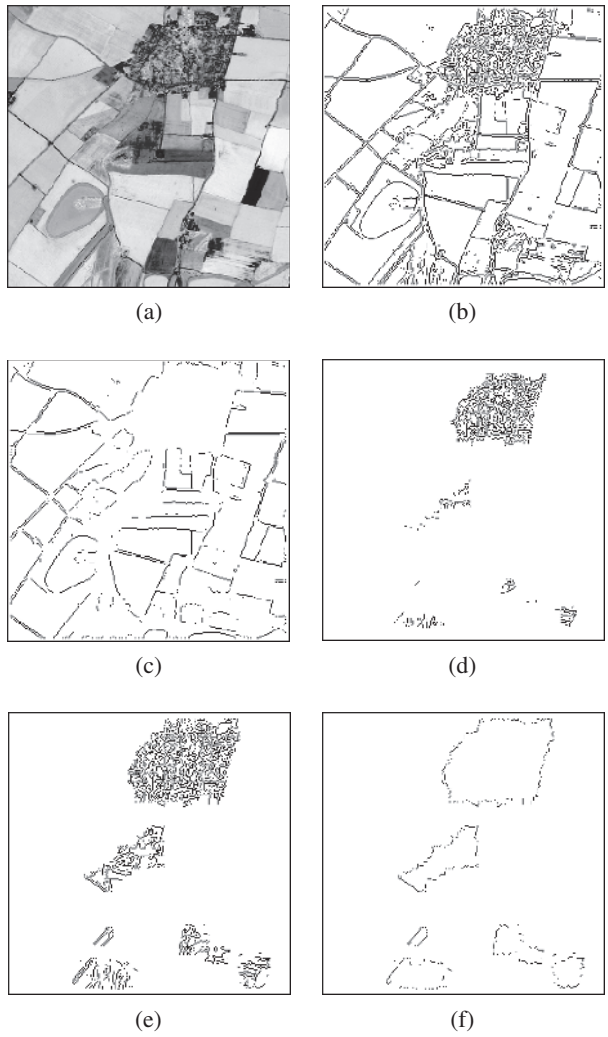


Figure 1.7 (a) Original image of size 256×256 showing a town. Source: Maria Petrou. (b) The edge map of the image obtained by using the Sobel edge detector. (c) The edgels that have field of view larger than 90° within a local neighbourhood of size 9×9 (i.e. $n = 9$). (d) The edge map of panel (b) without the edgels of panel (c) and any edgel in a neighbourhood of size 7×7 around them (i.e. $m = 7$). (e) The edgels of panel (d) plus all edgels from panel (b) that are in a neighbourhood of size 7×7 around them (i.e. $\delta m = 0$). (f) The final result keeping only the edgels of panel (e) that have field of view larger than 90° within a neighbourhood of 9×9.

Step 1: Identify the edgels in the image using an edge filter that does very little or no smoothing (e.g. Sobel filters). Call the output edge map A. An example is shown in Figure 1.7b.

Step 2: Consider a neighbourhood of size $n \times n$ around each edgel. Sort the neighbours of the edgel in a clockwise (or anti-clockwise) order around it. Find the angle between any two successive neighbours with the vertex of the angle being the edgel of interest. The largest of these angles is the field of view of the edgel of interest. This process is demonstrated in Figure 1.8.

(Continued)

Box 1.1 (Continued)

Step 3: In A keep only the edgels that have a field of view larger than d degrees. Call this output B. Such a result is shown in Figure 1.7c.

Step 4: Around each edgel you have kept in B, consider a window of size $m \times m$. Delete all edgels in A that are inside this window. Call the result C. An example is shown in Figure 1.7d. Note that in this way you have eliminated all those edgels with high viewing angles that do not delineate texture regions. However, at the same time the texture regions have shrunk.

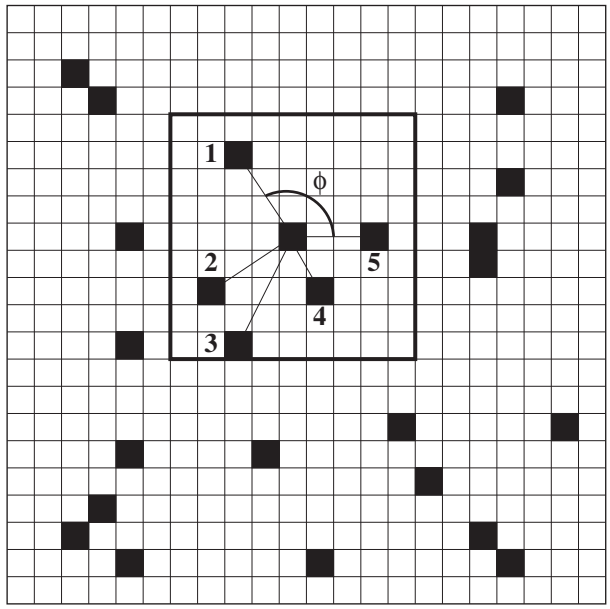


Figure 1.8 Each tile in this figure represents a pixel. The black tiles represent edgels. The windowed region with the thick black line represents the neighbourhood of size $n \times n$ we consider around the edgel in the middle. In this case $n = 9$. The neighbours inside this window are sorted in an anti-clockwise direction. The numbers next to them indicate their rank. We measure the angle between any two successive rays from the central edgel. The largest of these angles, marked here with ϕ, is the field of view of the central edgel.

Step 5: Around each edgel you have kept in C, consider a window of size $(m + \delta m) \times (m + \delta m)$. Restore inside the window all edgels that are present in A. Call the result D. An example is shown in Figure 1.7e. Note that now you have only the edgels that constitute the textured region. Constant δm makes sure that the window used in this step is slightly larger than the window used in the previous step, because we are dealing with discrete points which are at a certain average distance from each other.

Step 6: For each edgel in D calculate the field of view as you did before. Keep only the edgels that have a field of view larger than a threshold d. This is the final result. For the example the final result is shown in Figure 1.7f.

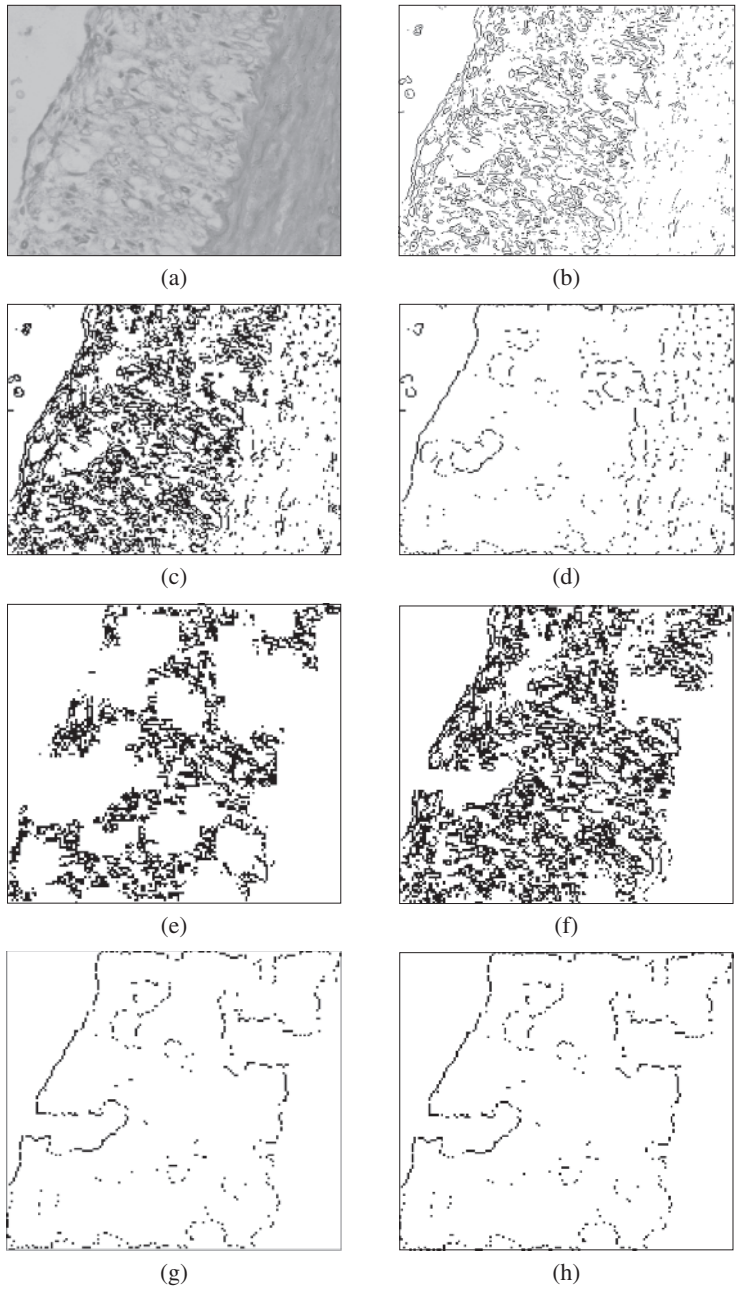


Figure 1.9 (a) An original image of size 382×287, showing the cross section of the aorta of a rabbit that had too much cholesterol. The edgels in the texture region are sparse. Source: Maria Petrou. (b) Edge map produced using a Sobel edge detector. (c) The edge map made denser by replacing every 4 pixels by 1 and marking the new pixel as an edgel, even if only a single pixel in the original 2×2 configuration was an edgel. (d) The edgels that have a field of view larger than 90° within a local neighbourhood of size 5×5 in panel (c). (e) The edgels of panel (c) after the edgels in (d) and their neighbouring edgels inside a window of size 4×4 are removed. (f) The edgels of panel (e) plus all their neighbouring edgels inside a window of size 6×6 in panel (c) restored. (g) The edgels in (f) which have a field of view larger than 90°. (h) The final result in the original size.

(Continued)

Box 1.1 (Continued)

This algorithm works well for texture regions with dense edgels. If the edgels are not very dense, but you still want to identify the textured regions, you may preprocess the edge map using the following step.

Step 1.5: Make the edge map denser by replacing every $l \times l$ tile of pixels by a single pixel. Even if only one of the pixels in the tile is an edgel, make the replacement pixel an edgel too. This way the edge map will shrink but the number of edgels will be preserved and the map will be made denser.

The result may be brought to the full size by the following step.

Step 7: Replace every pixel with a tile of size $l \times l$. All pixels in the tile will be marked either as edgels or background according to how the original pixel was marked. This will yield a thick boundary for the textured region. If this is a problem the boundary may be thinned while preserving its continuity.
Steps 1.5 and 7 are demonstrated in Figure 1.9.