3.9 Exercises 173

5. Interpolate adjacent CDFs for ﬁnal lookup.

**Ex 3.8: Padding for neighborhood operations** Write down the formulas for computing the padded pixel values *f*˜(*i, j*) as a function of the original pixel values *f* (*k, l*) and the image width and height (*M, N* ) for *each* of the padding modes shown in Figure 3.13. For example, for replication (clamping),

*f*˜(*i, j*) = *f* (*k, l*)*, k* = max(0*,* min(*M −* 1*, i*))*,*

*l* = max(0*,* min(*N −* 1*, j*))*,*

(Hint: you may want to use the min, max, mod, and absolute value operators in addition to the regular arithmetic operators.)

* Describe in more detail the advantages and disadvantages of these various modes.
* (Optional) Check what your graphics card does by drawing a texture-mapped rectangle where the texture coordinates lie beyond the [0*.*0*,* 1*.*0] range and using different texture clamping modes.

**Ex 3.9: Separable ﬁlters** Implement convolution with a separable kernel. The input should be a grayscale or color image along with the horizontal and vertical kernels. Make sure you support the padding mechanisms developed in the previous exercise. You will need this functionality for some of the later exercises. If you already have access to separable ﬁltering in an image processing package you are using (such as IPL), skip this exercise.

* (Optional) Use Pietro Perona’s (1995) technique to approximate convolution as a sum of a number of separable kernels. Let the user specify the number of kernels and report back some sensible metric of the approximation ﬁdelity.

**Ex 3.10: Discrete Gaussian ﬁlters** Discuss the following issues with implementing a dis- crete Gaussian ﬁlter:

* If you just sample the equation of a continuous Gaussian ﬁlter at discrete locations, will you get the desired properties, e.g., will the coefﬁcients sum up to 0? Similarly, if you sample a derivative of a Gaussian, do the samples sum up to 0 or have vanishing higher-order moments?
* Would it be preferable to take the original signal, interpolate it with a sinc, blur with a continuous Gaussian, then pre-ﬁlter with a sinc before re-sampling? Is there a simpler way to do this in the frequency domain?
* Would it make more sense to produce a Gaussian frequency response in the Fourier domain and to then take an inverse FFT to obtain a discrete ﬁlter?
* How does truncation of the ﬁlter change its frequency response? Does it introduce any additional artifacts?
* Are the resulting two-dimensional ﬁlters as rotationally invariant as their continuous analogs? Is there some way to improve this? In fact, can any 2D discrete (separable or non-separable) ﬁlter be truly rotationally invariant?