SNS COLLEGE OF ENGINEERING

Kurumbapalayam (Po), Coimbatore – 641 107

An Autonomous Institution

Accredited by NBA – AICTE and Accredited by NAAC – UGC with ‘A’ Grade
Approved by AICTE, New Delhi & Affiliated to Anna University, Chennai

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

COURSE NAME : 19EC513 – IMAGE PROCESSING AND COMPUTER VISION

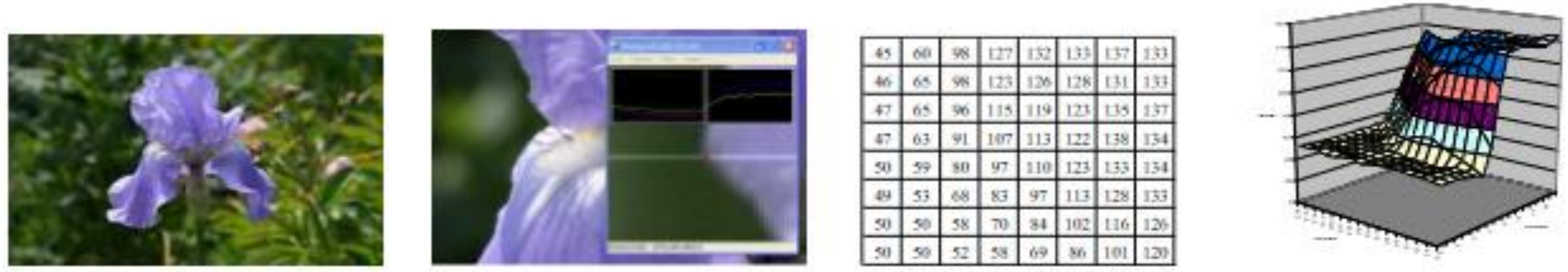
III YEAR / V SEMESTER

Unit V- Computer Vision

Topic : Point operator : Linear filtering

Point operators

The simplest kinds of image processing transforms are point operators, where each output pixel's value depends on only the corresponding input pixel value (plus, potentially, some globally collected information or parameters).



45	60	98	127	132	133	137	133
46	65	98	123	126	128	131	133
47	65	96	115	119	123	135	137
47	63	91	107	113	122	138	134
50	59	80	97	110	123	133	134
49	53	68	83	97	113	128	133
50	50	58	70	84	102	116	126
50	50	52	58	69	86	101	120

(a) (b) (c) (d)

Figure 3.3 *Visualizing image data: (a) original image; (b) cropped portion and scanline plot using an image inspection tool; (c) grid of numbers; (d) surface plot. For figures (c)–(d), the image was first converted to grayscale.*

Examples of such operators include brightness and contrast adjustments (Figure 3.2) as well as color correction and transformations. In the image processing literature, such operations are also known as point processes

We begin this section with a quick review of simple point operators, such as brightness scaling and image addition. Next, we discuss how colors in images can be manipulated.

We then present image compositing and matting operations, which play an important role in computational photography and computer graphics applications. Finally, we describe the more global process of histogram equalization. We close with an example application that manipulates tonal values (exposure and contrast) to improve image appearance

Pixel transforms

A general image processing operator is a function that takes one or more input images and produces an output image. In the continuous domain, this can be denoted as

$$g(\mathbf{x}) = h(f(\mathbf{x})) \quad \text{or} \quad g(\mathbf{x}) = h(f_0(\mathbf{x}), \ldots, f_n(\mathbf{x})),$$

where x is in the D-dimensional (usually D = 2 for images) domain of the input and output functions f and g, which operate over some range, which can either be scalar or vector-valued, e.g., for color images or 2D motion. For discrete (sampled) images, the domain consists of a finite number of pixel locations, x = (i, j), and we can write

$$g(i, j) = h(f(i, j))$$

Two commonly used point processes are multiplication and addition with a constant,

$$g(\mathbf{x}) = af(\mathbf{x}) + b. \tag{3.3}$$

The parameters $a > 0$ and b are often called the *gain* and *bias* parameters; sometimes these parameters are said to control *contrast* and *brightness*, respectively (Figures 3.2b–c).[2] The bias and gain parameters can also be spatially varying,

$$g(\mathbf{x}) = a(\mathbf{x})f(\mathbf{x}) + b(\mathbf{x}), \tag{3.4}$$

e.g., when simulating the *graded density filter* used by photographers to selectively darken the sky or when modeling vignetting in an optical system.

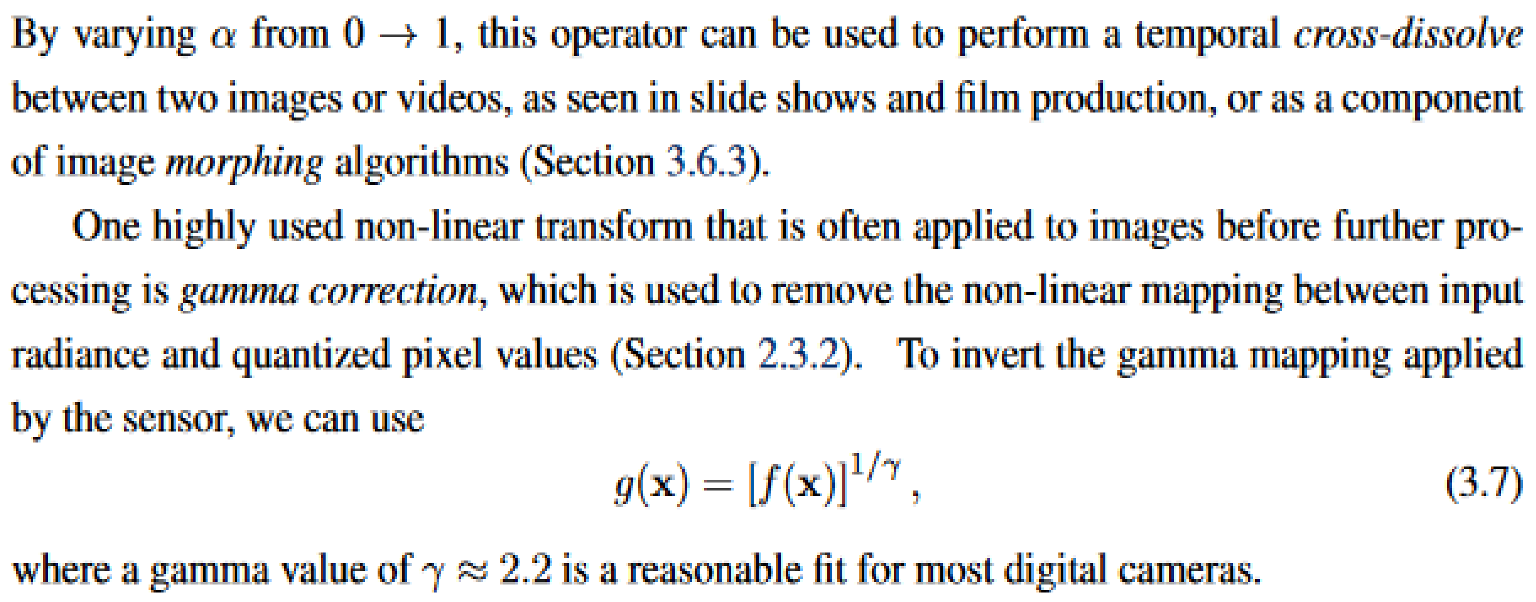
Multiplicative gain (both global and spatially varying) is a *linear* operation, as it obeys the *superposition principle*,

$$h(f_0 + f_1) = h(f_0) + h(f_1). \tag{3.5}$$

(We will have more to say about linear shift invariant operators in Section 3.2.) Operators such as image squaring (which is often used to get a local estimate of the *energy* in a band-pass filtered signal, see Section 3.5) are not linear.

Another commonly used *dyadic* (two-input) operator is the *linear blend* operator,

$$g(\mathbf{x}) = (1 - \alpha)f_0(\mathbf{x}) + \alpha f_1(\mathbf{x}). \tag{3.6}$$



By varying α from $0 \rightarrow 1$, this operator can be used to perform a temporal *cross-dissolve* between two images or videos, as seen in slide shows and film production, or as a component of image *morphing* algorithms (Section 3.6.3).

One highly used non-linear transform that is often applied to images before further processing is *gamma correction*, which is used to remove the non-linear mapping between input radiance and quantized pixel values (Section 2.3.2). To invert the gamma mapping applied by the sensor, we can use

$$g(\mathbf{x}) = [f(\mathbf{x})]^{1/\gamma}, \tag{3.7}$$

where a gamma value of $\gamma \approx 2.2$ is a reasonable fit for most digital cameras.

Color transforms

While color images can be treated as arbitrary vector-valued functions or collections of inde-
pendent bands, it usually makes sense to think about them as highly correlated signals with strong connections to the image formation process, sensor design, and human perception.

Consider, for example, brightening a picture by adding a constant value to all three channels. Can you tell if this achieves the desired effect of making the image look brighter? Can you see any undesirable side-effects or artifacts?

In fact, adding the same value to each color channel not only increases the apparent in-
tensity of each pixel, it can also affect the pixel's hue and saturation. How can we define and manipulate such quantities in order to achieve the desired perceptual effects?

Similarly, color balancing (e.g., to compensate for incandescent lighting) can be per-
formed either by multiplying each channel with a different scale factor or by the more com-
plex process of mapping to XYZ color space, changing the nominal white point, and mapping back to RGB, which can be written down using a linear 3 × 3 color twist transform matrix. Exercises 2.8 and 3.1 have you explore some of these issues.

Compositing and matting

In many photo editing and visual effects applications, it is often desirable to cut a foreground object out of one scene and put it on top of a different background (Figure 3.4). The process of extracting the object from the original image is often called matting (Smith and Blinn while the process of inserting it into another image (without visible artifacts) is called compositing

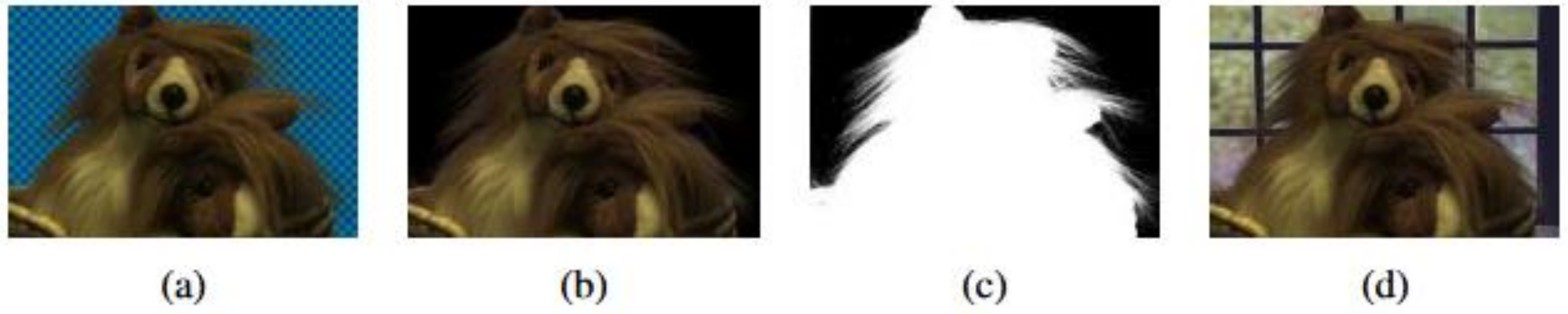


Figure 3.4 *Image matting and compositing (Chuang, Curless* et al. *2001) © 2001 IEEE: (a) source image; (b) extracted foreground object F; (c) alpha matte α shown in grayscale; (d) new composite C.*

Histogram equalization

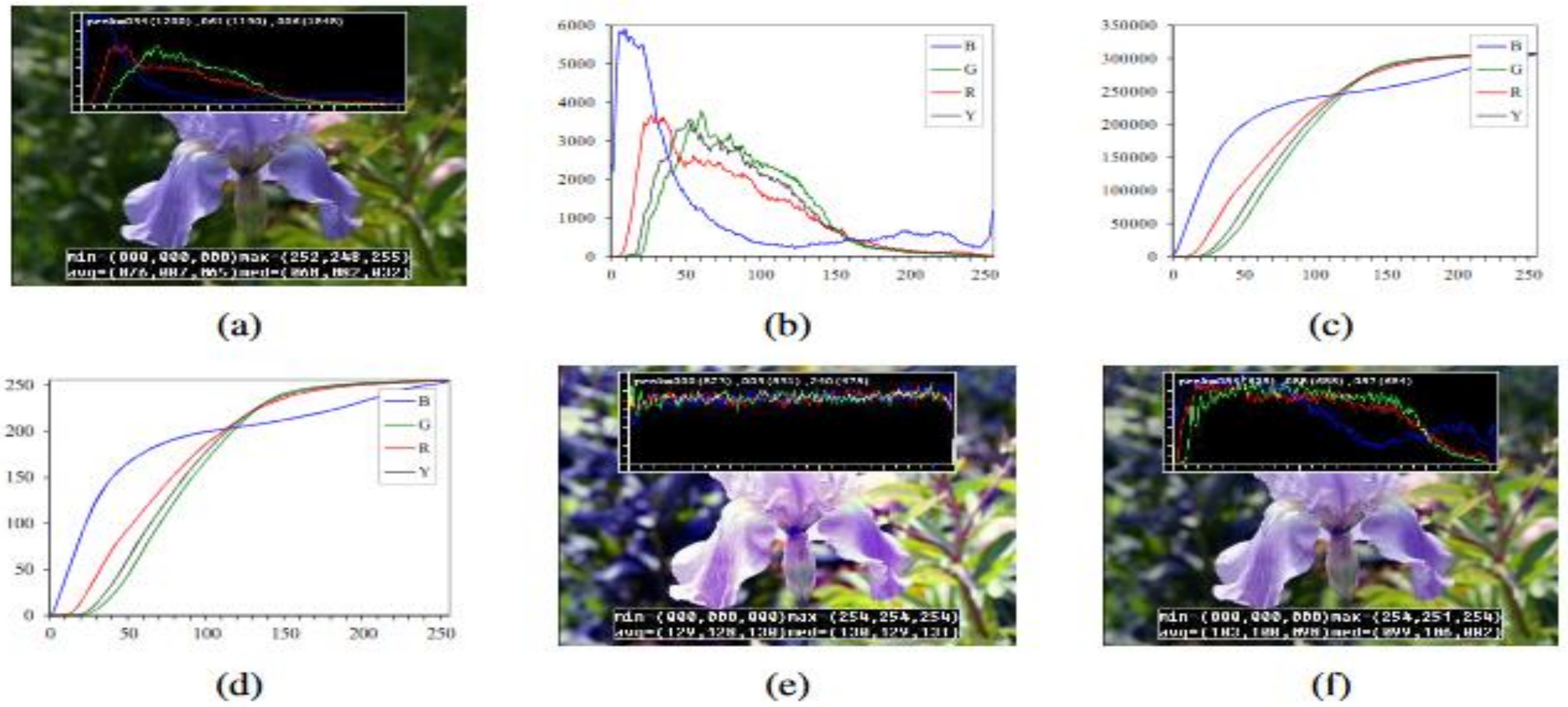


Figure 3.7 *Histogram analysis and equalization: (a) original image; (b) color channel and intensity (luminance) histograms; (c) cumulative distribution functions; (d) equalization (transfer) functions; (e) full histogram equalization; (f) partial histogram equalization.*

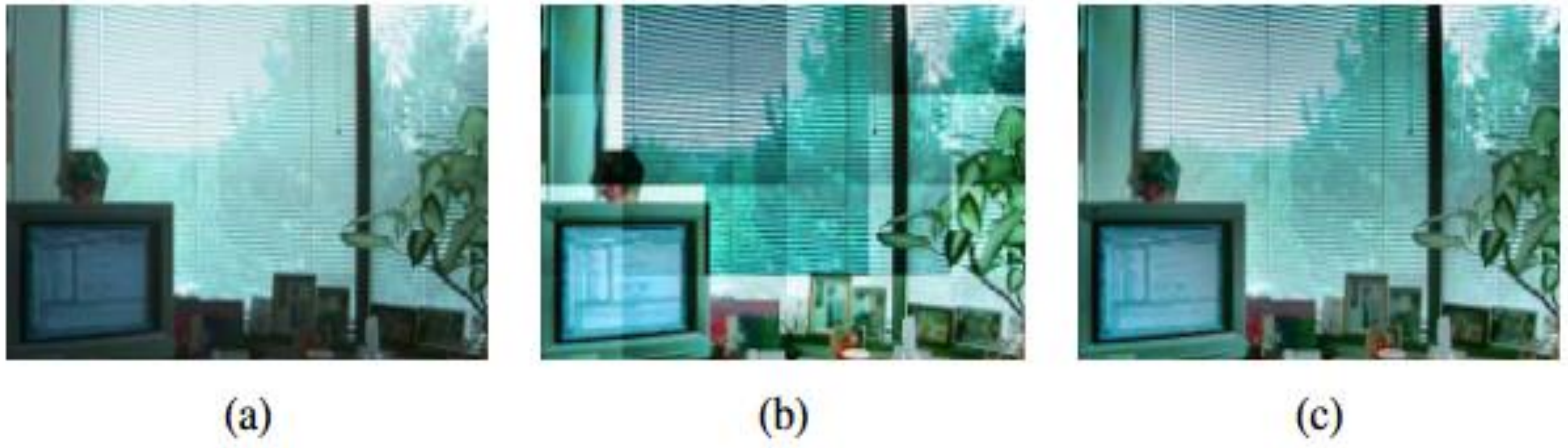
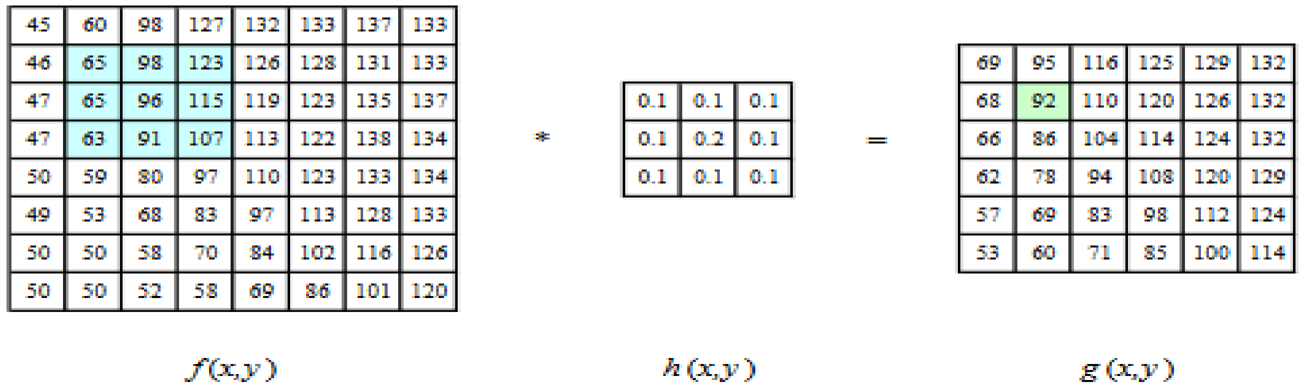


Figure 3.8 *Locally adaptive histogram equalization: (a) original image; (b) block histogram equalization; (c) full locally adaptive equalization.*

Linear filtering

Locally adaptive histogram equalization is an example of a neighborhood operator or local operator, which uses a collection of pixel values in the vicinity of a given pixel to determine its final output value (Figure 3.10). In addition to performing local tone adjustment, neighborhood operators can be used to filter images to add soft blur, sharpen details, accentuate edges, or remove noise (Figure 3.11b–d). In this section, we look at linear filtering operators, which involve fixed weighted combinations of pixels in small neighborhoods.



45	60	98	127	132	133	137	133
46	65	98	123	126	128	131	133
47	65	96	115	119	123	135	137
47	63	91	107	113	122	138	134
50	59	80	97	110	123	133	134
49	53	68	83	97	113	128	133
50	50	58	70	84	102	116	126
50	50	52	58	69	86	101	120

0.1	0.1	0.1
0.1	0.2	0.1
0.1	0.1	0.1

69	95	116	125	129	132
68	92	110	120	126	132
66	86	104	114	124	132
62	78	94	108	120	129
57	69	83	98	112	124
53	60	71	85	100	114

Figure 3.10 *Neighborhood filtering (convolution): The image on the left is convolved with the filter in the middle to yield the image on the right. The light blue pixels indicate the source neighborhood for the light green destination pixel.*

The most widely used type of neighborhood operator is a *linear filter*, where an output pixel's value is a weighted sum of pixel values within a small neighborhood $\mathcal{N}$ (Figure 3.10),

$$g(i,j) = \sum_{k,l} f(i+k, j+l)h(k,l). \tag{3.12}$$

The entries in the weight *kernel* or *mask* $h(k,l)$ are often called the *filter coefficients*. The above *correlation* operator can be more compactly notated as

$$g = f \otimes h. \tag{3.13}$$

A common variant on this formula is

$$g(i,j) = \sum_{k,l} f(i-k, j-l)h(k,l) = \sum_{k,l} f(k,l)h(i-k, j-l), \tag{3.14}$$

where the sign of the offsets in f has been reversed, This is called the *convolution* operator,

$$g = f * h, \tag{3.15}$$

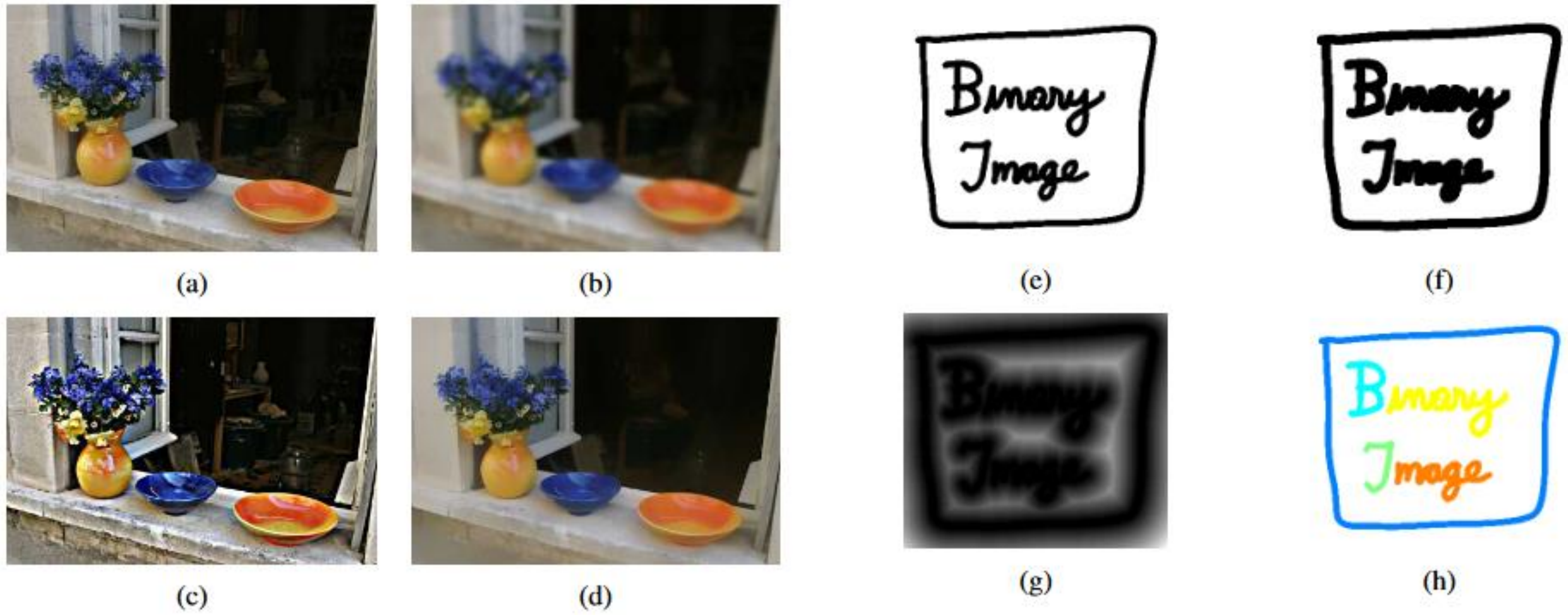


Figure 3.11 *Some neighborhood operations: (a) original image; (b) blurred; (c) sharpened; (d) smoothed with edge-preserving filter; (e) binary image; (f) dilated; (g) distance transform; (h) connected components. For the dilation and connected components, black (ink) pixels are assumed to be active, i.e., to have a value of 1 in Equations (3.44–3.48).*

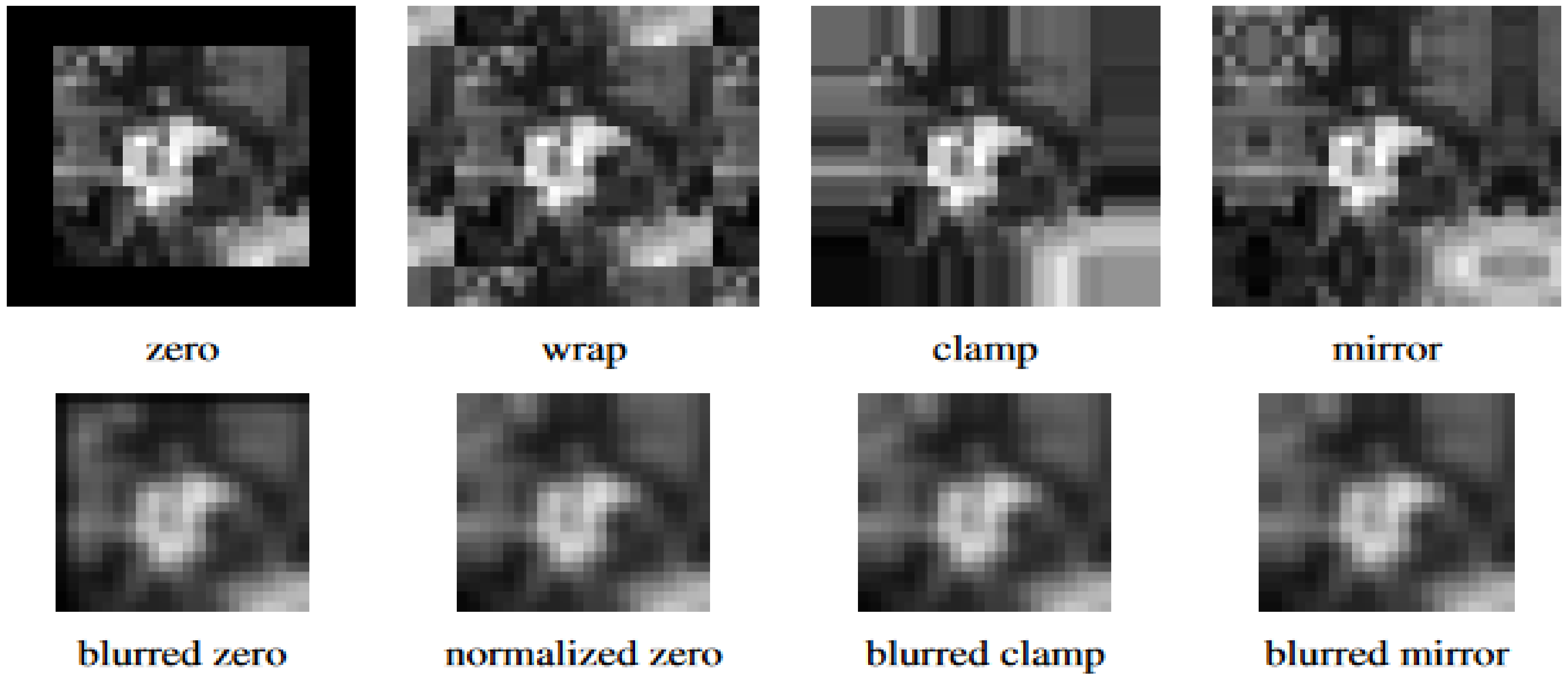
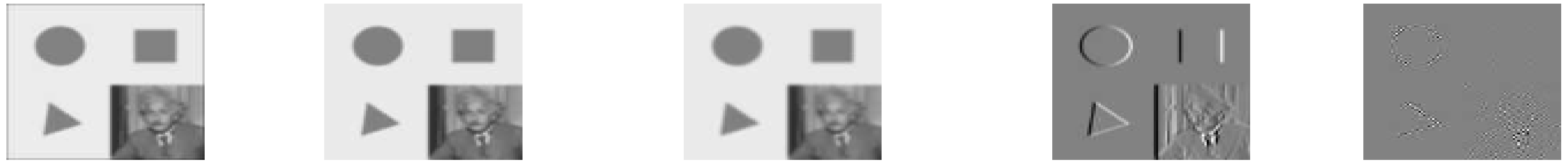


Figure 3.13 *Border padding (top row) and the results of blurring the padded image (bottom row). The normalized zero image is the result of dividing (normalizing) the blurred zero-padded RGBA image by its corresponding soft alpha value.*

- zero: set all pixels outside the source image to 0 (a good choice for alpha-matted cutout images);
- constant (border color): set all pixels outside the source image to a specified border value;
- clamp (replicate or clamp to edge): repeat edge pixels indefinitely;
- (cyclic) wrap (repeat or tile): loop “around” the image in a “toroidal” configuration;
- mirror: reflect pixels across the image edge;
- extend: extend the signal by subtracting the mirrored version of the signal from the edge pixel value

$$\frac{1}{K^2}\begin{array}{|c|c|c|c|}\hline 1 & 1 & \cdots & 1 \\ \hline 1 & 1 & \cdots & 1 \\ \hline \vdots & \vdots & 1 & \vdots \\ \hline 1 & 1 & \cdots & 1 \\ \hline \end{array} \quad \frac{1}{16}\begin{array}{|c|c|c|}\hline 1 & 2 & 1 \\ \hline 2 & 4 & 2 \\ \hline 1 & 2 & 1 \\ \hline \end{array} \quad \frac{1}{256}\begin{array}{|c|c|c|c|c|}\hline 1 & 4 & 6 & 4 & 1 \\ \hline 4 & 16 & 24 & 16 & 4 \\ \hline 6 & 24 & 36 & 24 & 6 \\ \hline 4 & 16 & 24 & 16 & 4 \\ \hline 1 & 4 & 6 & 4 & 1 \\ \hline \end{array} \quad \frac{1}{8}\begin{array}{|c|c|c|}\hline -1 & 0 & 1 \\ \hline -2 & 0 & 2 \\ \hline -1 & 0 & 1 \\ \hline \end{array} \quad \frac{1}{4}\begin{array}{|c|c|c|}\hline 1 & -2 & 1 \\ \hline -2 & 4 & -2 \\ \hline 1 & -2 & 1 \\ \hline \end{array}$$

$$\frac{1}{K}\begin{array}{|c|c|c|c|}\hline 1 & 1 & \cdots & 1 \\ \hline \end{array} \quad \frac{1}{4}\begin{array}{|c|c|c|}\hline 1 & 2 & 1 \\ \hline \end{array} \quad \frac{1}{16}\begin{array}{|c|c|c|c|c|}\hline 1 & 4 & 6 & 4 & 1 \\ \hline \end{array} \quad \frac{1}{2}\begin{array}{|c|c|c|}\hline -1 & 0 & 1 \\ \hline \end{array} \quad \frac{1}{2}\begin{array}{|c|c|c|}\hline 1 & -2 & 1 \\ \hline \end{array}$$



(a) box, $K = 5$ (b) bilinear (c) "Gaussian" (d) Sobel (e) corner

Figure 3.14 *Separable linear filters: For each image (a)–(e), we show the 2D filter kernel (top), the corresponding horizontal 1D kernel (middle), and the filtered image (bottom). The filtered Sobel and corner images are signed, scaled up by 2× and 4×, respectively, and added to a gray offset before display.*

THANK YOU !!!