
Lecture #4: Pixels and Filters

Brian Hicks, Alec Arshavsky, Sam Trautwein, Christine Phan, James Ortiz
Department of Computer Science
Stanford University
Stanford, CA 94305
`{bhicks2, arshava, rodion1, cxphan, jameso2}@cs.stanford.edu`

Contents:

1. Image Sampling and Quantization
2. Image Histograms
3. Images as Functions
4. Linear Systems
5. Convolution and Correlation

1 Image Sampling and Quantization

1.1 Image Types

Binary Images contain pixels that are either black (0) or white (1).

Grayscale Images have a wider range of intensity than black and white. Each pixel is a shade of gray with pixel values ranging between 0 (white) and 255 (black).

Color Images have multiple channels with colors (RGB, LAB, HSV). Each pixel in a channel are values 0-255 for intensity. A 3D *tensor* usually represents color images (Width x Length x 3), where the 3 channels can represent RGB, LAB, HSV, and so on.

1.2 Sampling and Resolution

Images are **samples**: they are not continuous, they are discrete pixels of a certain size and density. This can lead to errors (or graininess) because we can only measure with a certain resolution and must approximate.

Resolution is a sampling parameter, defined in dots per inch. The standard DPI value for screens is 72 DPI.



Figure 1: Illustrations of different pixel densities. Taken from the accompanying lecture slides. (Slide 14, slide credit Ulas Bagci)

Pixels are quantized (i.e, all pixels (or channels of a pixel) have one of a set numbers of values (usually [0, 255]). Quantization and sampling loses information due to a finite precision.

2 Image Histograms

Histograms measure the frequency of brightness within the image: how many times does a particular pixel value appear in an image.

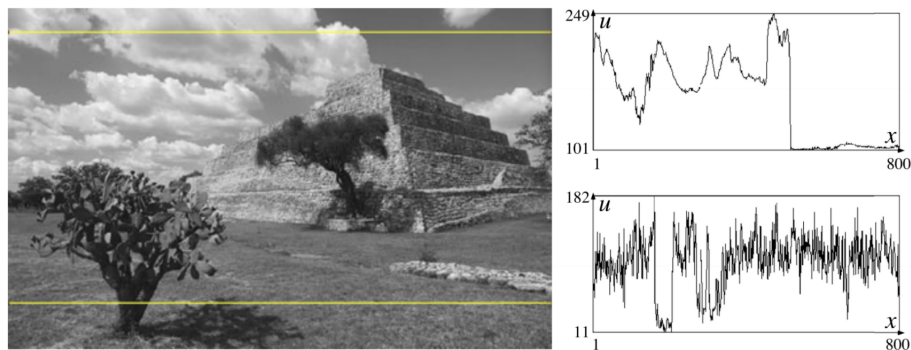


Figure 2: The image is sampled at two vertical positions, sampling a patch of sky and sampling a patch of grass. The corresponding histograms are shown to the right. Taken from the accompanying lecture slide (Slide 23, slide credit Dr. Mubarak Shah)

Histograms can help us detect particular features in images, for example:

- Sky: Smooth coloration denotes consistency in image, consistent with the image of a sky.
- Grass: A jagged histogram shows wide ranging variety in coloration, consistent with the shadows of a grass field.
- Faces: The color composition of a face will be displayed in a histogram.

An image histogram measures the frequency of certain grayscale intensities in an image. Histograms can be used to start to build up an idea of what things look like, which can be useful for classifiers.

3 Images as Functions

Most images that we deal with in computer vision are digital, which means that they are discrete representations of the scene shown. This discretization is achieved through the sampling of 2-dimensional space onto a regular grid, eventually producing a representation of the image as a matrix of integer values.

When dealing with images, we can imagine the image matrix as infinitely tall and wide. However, the displayed image is only a finite subset of this infinite matrix. When we deal conceive of images in this way, we can write them as coordinates in a matrix

$$\begin{bmatrix} \ddots & & \vdots & & \ddots \\ \dots & f[-1, 1] & f[0, 1] & f[1, 1] & \dots \\ & f[-1, 0] & f[0, 0] & f[1, 0] & \\ & f[-1, -1] & f[0, -1] & f[1, -1] & \dots \\ \ddots & & \vdots & & \ddots \end{bmatrix}$$

We can also treat an image as a function $f : \mathbb{R}^2 \rightarrow \mathbb{R}^N$. When we do so, $f[n, m]$ where $f[n, m]$ is the intensity of a pixel at position (m, n). Note that we use square brackets, rather than the typical parentheses, to denote discrete functions.

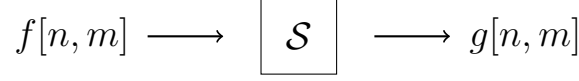


Figure 3: Graphical representation of a system's mapping of f to g

When we treat an image as a function, it is defined over a rectangle with finite range. For example, the following function f returns the (grayscale) intensity of a single pixel in an image located between a and b horizontally and c and d vertically.

$$f : [a, b] \times [c, d] \rightarrow [0, 255] \quad (\text{Grayscale Pixel Intensity})$$

The set of values $[a, b] \times [c, d]$ is known as the *domain support*, and contains all values that are valid inputs to the function f , while $[0, 255]$ (in this case) is the range, which defines the set of possible outputs.

We can also treat color images as a function mapping $\mathbb{R}^2 \rightarrow \mathbb{R}^3$. For example the RGB intensities of a given pixel can be written as the function g below

$$g[x, y] = \begin{bmatrix} r[x, y] \\ g[x, y] \\ b[x, y] \end{bmatrix} \quad (\text{Color Pixel Intensity})$$

where $r, g, b : [a, b] \times [c, d] \rightarrow [0, 255]$.

4 Linear Systems (Filters)

The term *filtering* refers to a process that forms a new image whose pixel values are transformations of the original pixel values. In general, our purpose in applying filters is to extract useful information (e.g., edge detection) or else change its visual properties (e.g., de-noising).

Filters are examples of *systems*, which are units that convert an input function $f[m, n]$ to an output (or response) function $g[m, n]$ where m, n are the independent variables. When dealing with images, m, n are the representation of a spatial position in the image.

Notationally, \mathcal{S} is referred to as the *system operator*, which maps a member of the set of possible outputs $g[m, n]$ to a member of the set of possible inputs $f[m, n]$. When using notation involving \mathcal{S} , we can write that

$$\begin{aligned} \mathcal{S}[g] &= f \\ \mathcal{S}\{f[m, n]\} &= g[m, n] \\ f[m, n] &\xrightarrow{\mathcal{S}} g[m, n] \end{aligned}$$

4.1 Examples of Filters

Moving Average

One intuitive example of a filter is the *moving average*. This filter sets the value of a pixel to be the average of the nine pixels in a 3×3 radius around it. Mathematically, we can represent this as

$$g[m, n] = \frac{1}{9} \sum_{i=-1}^1 \sum_{j=-1}^1 f[m-i, n-j] \quad (\text{Weighted Average})$$

This weighted average filter serves to smooth out the sharper edges of the image, creating a blurred or smoothed effect.

Image Segmentation

We can also use filters to perform rudimentary *image segmentation* based on a simple threshold system. In this case, the filter sets the value of a pixel either to an extremely high or an extremely low value, depending on whether or not it meets the threshold t . Mathematically, we write this as

$$g[m, n] = \begin{cases} 255 & f[m, n] \geq t \\ 0 & \text{otherwise} \end{cases} \quad (\text{Threshold})$$

Roughly speaking, this basic image segmentation filter breaks divides an image's pixels into binary classifications of bright regions and dark regions, depending on whether or not the $f[m, n] \geq t$.

4.2 Properties of Systems

When discussing specific systems, it is useful to be able to describe their properties. The following includes a list of properties that a system *may* possess, but not all systems will have all (or any) of these properties. In other words, these are potential characteristics of individual systems, not traits of a systems in general.

Amplitude Properties

- *Additivity*: A system is additive if it satisfies the equation

$$\mathcal{S}[f_i[m, n] + f_j[m, n]] = \mathcal{S}[f_i[m, n]] + \mathcal{S}[f_j[m, n]]$$

- *Homogeneity*: A system is homogeneous if it satisfies the equation

$$\mathcal{S}[\alpha f_i[n, m]] = \alpha \mathcal{S}[f_i[n, m]]$$

- *Superposition*: A system has the property of superposition if it satisfies the equation

$$\mathcal{S}[\alpha f_i[n, m]] + \beta f_j[n, m]] = \alpha \mathcal{S}[f_i[n, m]] + \beta \mathcal{S}[f_j[n, m]]$$

- *Stability*: A system is stable if it satisfies the inequality

$$|f[n, m]| \leq k \implies |g[n, m]| \leq ck$$

for some c .

- *Invertibility*: A system is invertible if it satisfies the equation

$$\mathcal{S}^{-1}[\mathcal{S}[f[n, m]]] = f[n, m]$$

Spatial Properties

- *Causality*: A system is causal if for $m < m_0$ and $n < n_0$

$$f[m, n] = 0 \implies g[m, n] = 0$$

- *Shift Invariance*: A system is shift invariant if

$$f[m - m_0, n - n_0] \xrightarrow{\mathcal{S}} g[m - m_0, n - n_0]$$

4.3 Linear Systems

A *linear system* is a system that satisfies the property of superposition. When we employ a linear system for filtering, we create a new image whose pixels are weighted sums of the original pixel values, using the same set of weights for each pixel. A *linear shift-invariant system* is a linear system that is also shift invariant.

Linear systems also have what is known as an *impulse response*. To determine the impulse response of a system \mathcal{S} , consider first $\delta_2[m, n]$. This is a function defined as follows

$$\delta_2[m, n] = \begin{cases} 1 & m = 0 \text{ and } n = 0 \\ 0 & \text{otherwise} \end{cases}$$

The impulse response r is then simply

$$r = \mathcal{S}[\delta_2]$$

A simple linear shift-invariant system is a system that shifts the pixels of an image, based on the shifting property of the delta function.

$$f[m, n] = \sum_{i=-\infty}^{\infty} \sum_{j=-\infty}^{\infty} f[i, j] \delta_2[m - i, n - j]$$

We can then use the superposition property to write *any* linear shift-invariant system as a weighted sum of such shifting system

$$\alpha_1 \sum_{i=-\infty}^{\infty} \sum_{j=-\infty}^{\infty} f[i, j] \delta_{2,1}[m-i, n-j] + \alpha_{2,2} \sum_{i=-\infty}^{\infty} \sum_{j=-\infty}^{\infty} f[i, j] \delta_{2,3}[m-i, n-j] + \dots$$

We then define the filter h of a linear shift-invariant system as

$$h[m, n] = \alpha_1 \delta_{2,1}[m-i, n-j] + \alpha_2 \delta_{2,2}[m-i, n-j] + \dots$$

Impulse response (for all linear systems) Delta[n,m] function: has value that is 1 specifically at one pixel, gives back a response, $h[n,m]$ A shifted delta function gives back a shifted response

Linear shift invariant systems (LSI) Example: a moving average filter is a summation of impulse responses

- Systems that satisfy the superposition property
- Have an *impulse response*: $\mathcal{S}[\delta_2[n, m]] = \delta_2[n, m]$
- Discrete convolution: $f[n, m] * h[n, m]$ (multiplication of shifted-version of impulse response by original function)

5 Convolution and Correlation

5.1 Convolution

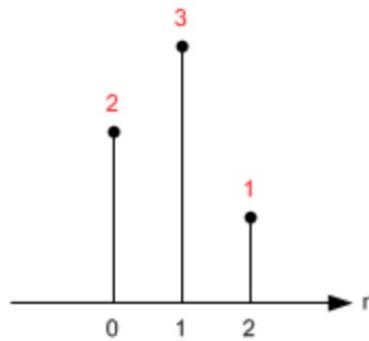
The easiest way to think of convolution is as a system that uses information from neighboring pixels to filter the target pixel. A good example of this is a moving average, or box filter that we talked about earlier.

Convolution is represented by $*$, for example:

$f[n, m] * h[n, m]$ represents a function being multiplied by a shifted impulse response

Convolution allows us to compute the output of passing any input signal through a system simply by considering the impulse response of the system. To understand what this means, we must first understand how to break up a signal into a set of impulse functions.

The impulse (delta) function, $\delta[n]$, is defined to be 1 at $n = 0$ and 0 elsewhere. As seen in the image below, any signal can be decomposed as the weighted sum of impulse functions.



$$x[n] = x[0]\delta[n] + x[1]\delta[n-1] + x[2]\delta[n-2]$$

Figure 4: An example decomposition of a signal into an impulse function. (Adapted from <http://www.songho.ca/dsp/convolution/convolution.html>)

More generally, an arbitrary signal x can be written as $x[n] = \sum_{k=-\infty}^{\infty} x[k]\delta[n-k]$.

The impulse response of a system, $h[n]$, is defined as the output resulting from passing an impulse function into a system. When a system is linear, scaling the impulse function causes the impulse

response to be scaled by the same amount. Moreover, when the system is shift-invariant, shifting the impulse function causes the impulse response to be shifted by the same amount. The image below illustrates these properties for a signal consisting of 3 components.

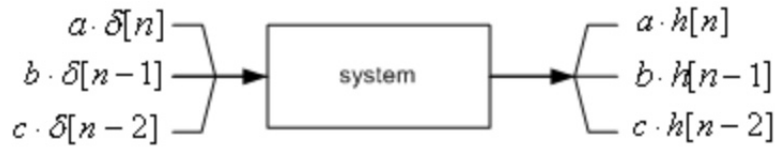


Figure 5: An impulse function is sent through a system to create an impulse response function. Adapted from <http://www.songho.ca/dsp/convolution/convolution.html>

More generally, for an arbitrary input signal $x[n] = \sum_{k=-\infty}^{\infty} x[k]\delta[n-k]$ passed into a linear, shift-invariant system, the output is $y[n] = \sum_{k=-\infty}^{\infty} x[k]h[n-k]$, i.e. the convolution of the signal x with the impulse response h .

Convolution can also be done in 2 dimensions. For example, when an arbitrary, 2D signal $x[n, m] = \sum_{i=-\infty}^{\infty} \sum_{j=-\infty}^{\infty} x[i, j]\delta[n-i, m-j]$ is passed into a linear, shift-invariant system, the output is $y[n, m] = \sum_{i=-\infty}^{\infty} \sum_{j=-\infty}^{\infty} x[i, j]h[n-i, m-j]$, i.e. the convolution of the signal x with the impulse response h in 2 dimensions.

5.2 Correlation

Cross correlation is the same as convolution, except that the filter kernel is not flipped. Two-dimensional cross correlation is represented as:

$$r[k, l] = \sum_{m=-\infty}^{\infty} \sum_{n=-\infty}^{\infty} f[m+k, n+l]g[m, n]$$

It can be used to find known features in images by using a kernel that contains target features.

Acknowledgments

We would like to acknowledge Professor Niebles and Krishna and the rest of the teach staff for compiling the invaluable accompanying lecture slides and delivering such an informative lecture.