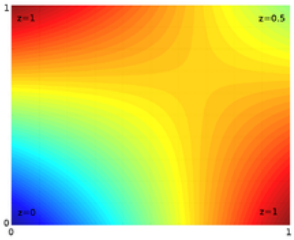
Bilinear interpolation

In mathematics, **bilinear interpolation** is an extension of linear interpolation for interpolating functions of two variables (e.g., *x* and *y*) on a rectilinear 2D grid.

Bilinear interpolation is performed using linear interpolation first in one direction, and then again in the other direction. Although each step is linear in the sampled values and in the position, the interpolation as a whole is not linear but rather quadratic in the sample location.

Bilinear interpolation is one of the basic resampling techniques in computer vision and image processing, where it is also called **bilinear filtering** or **bilinear texture mapping**.

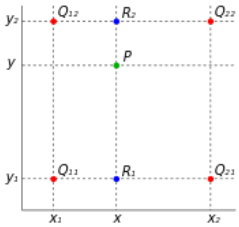


Example of bilinear interpolation on the unit square with the *z* values 0, 1, 1 and 0.5 as indicated. Interpolated values in between represented by color.

Contents

Computation

Suppose that we want to find the value of the unknown function *f* at the point (*x*, *y*). It is assumed that we know the value of *f* at the four points $Q_{11} = (x_1, y_1)$, $Q_{12} = (x_1, y_2)$, $Q_{21} = (x_2, y_1)$, and $Q_{22} = (x_2, y_2)$.



The four red dots show the data points and the green dot is the point at which we want to interpolate.

Repeated linear interpolation

We first do linear interpolation in the *x*-direction. This yields

$$f(x, y_1) = \frac{x_2 - x}{x_2 - x_1} f(Q_{11}) + \frac{x - x_1}{x_2 - x_1} f(Q_{21}),$$
$$f(x, y_2) = \frac{x_2 - x}{x_2 - x_1} f(Q_{12}) + \frac{x - x_1}{x_2 - x_1} f(Q_{22}).$$

We proceed by interpolating in the *y*-direction to obtain the desired estimate:

$$\begin{aligned} f(x, y) &= \frac{y_2 - y}{y_2 - y_1} f(x, y_1) + \frac{y - y_1}{y_2 - y_1} f(x, y_2) \\ &= \frac{y_2 - y}{y_2 - y_1} \left(\frac{x_2 - x}{x_2 - x_1} f(Q_{11}) + \frac{x - x_1}{x_2 - x_1} f(Q_{21}) \right) + \frac{y - y_1}{y_2 - y_1} \left(\frac{x_2 - x}{x_2 - x_1} f(Q_{12}) + \frac{x - x_1}{x_2 - x_1} f(Q_{22}) \right) \\ &= \frac{1}{(x_2 - x_1)(y_2 - y_1)} \left(f(Q_{11})(x_2 - x)(y_2 - y) + f(Q_{21})(x - x_1)(y_2 - y) + f(Q_{12})(x_2 - x)(y - y_1) + f(Q_{22})(x - x_1)(y - y_1) \right) \\ &= \frac{1}{(x_2 - x_1)(y_2 - y_1)} \begin{bmatrix} x_2 - x & x - x_1 \end{bmatrix} \begin{bmatrix} f(Q_{11}) & f(Q_{12}) \\ f(Q_{21}) & f(Q_{22}) \end{bmatrix} \begin{bmatrix} y_2 - y \\ y - y_1 \end{bmatrix}. \end{aligned}$$

Note that we will arrive at the same result if the interpolation is done first along the *y* direction and then along the *x* direction.[1]

Polynomial fit

An alternative way is to write the solution to the interpolation problem as a multilinear polynomial

$$f(x, y) \approx a_{00} + a_{10} x + a_{01} y + a_{11} xy,$$

where the coefficients are found by solving the linear system

$$\begin{bmatrix} 1 & x_1 & y_1 & x_1y_1 \\ 1 & x_1 & y_2 & x_1y_2 \\ 1 & x_2 & y_1 & x_2y_1 \\ 1 & x_2 & y_2 & x_2y_2 \end{bmatrix} \begin{bmatrix} a_{00} \\ a_{10} \\ a_{01} \\ a_{11} \end{bmatrix} = \begin{bmatrix} f(Q_{11}) \\ f(Q_{12}) \\ f(Q_{21}) \\ f(Q_{22}) \end{bmatrix},$$

yielding the result

$$\begin{bmatrix} a_{00} \\ a_{10} \\ a_{01} \\ a_{11} \end{bmatrix} = \frac{1}{(x_2-x_1)(y_2-y_1)} \begin{bmatrix} x_2y_2 & -x_2y_1 & -x_1y_2 & x_1y_1 \\ -y_2 & y_1 & y_2 & -y_1 \\ -x_2 & x_2 & x_1 & -x_1 \\ 1 & -1 & -1 & 1 \end{bmatrix} \begin{bmatrix} f(Q_{11}) \\ f(Q_{12}) \\ f(Q_{21}) \\ f(Q_{22}) \end{bmatrix}.$$

Weighted mean

The solution can also be written as a weighted mean of the $f(Q)$:

$$f(x,y) \approx w_{11}f(Q_{11}) + w_{12}f(Q_{12}) + w_{21}f(Q_{21}) + w_{22}f(Q_{22}),$$

where the weights sum to 1 and satisfy the transposed linear system

$$\begin{bmatrix} 1 & 1 & 1 & 1 \\ x_1 & x_1 & x_2 & x_2 \\ y_1 & y_2 & y_1 & y_2 \\ x_1y_1 & x_1y_2 & x_2y_1 & x_2y_2 \end{bmatrix} \begin{bmatrix} w_{11} \\ w_{12} \\ w_{21} \\ w_{22} \end{bmatrix} = \begin{bmatrix} 1 \\ x \\ y \\ xy \end{bmatrix},$$

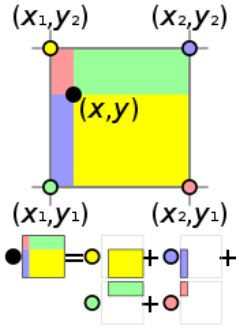
yielding the result

$$\begin{bmatrix} w_{11} \\ w_{21} \\ w_{12} \\ w_{22} \end{bmatrix} = \frac{1}{(x_2-x_1)(y_2-y_1)} \begin{bmatrix} x_2y_2 & -y_2 & -x_2 & 1 \\ -x_2y_1 & y_1 & x_2 & -1 \\ -x_1y_2 & y_2 & x_1 & -1 \\ x_1y_1 & -y_1 & -x_1 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ x \\ y \\ xy \end{bmatrix},$$

which simplifies to

$$\begin{aligned} w_{11} &= (x_2-x)(y_2-y)/(x_2-x_1)(y_2-y_1), \\ w_{12} &= (x_2-x)(y-y_1)/(x_2-x_1)(y_2-y_1), \\ w_{21} &= (x-x_1)(y_2-y)/(x_2-x_1)(y_2-y_1), \\ w_{22} &= (x-x_1)(y-y_1)/(x_2-x_1)(y_2-y_1), \end{aligned}$$

in agreement with the result obtained by repeated linear interpolation. The set of weights can also be interpreted as a set of generalized barycentric coordinates for a rectangle.



A geometric visualisation of bilinear interpolation. The product of the value at the desired point (black) and the entire area is equal to the sum of the products of the value at each corner and the partial area diagonally opposite the corner (corresponding colours).

Alternative matrix form

Combining the above, we have

$$f(x,y) \approx \frac{1}{(x_2-x_1)(y_2-y_1)} \begin{bmatrix} f(Q_{11}) & f(Q_{12}) & f(Q_{21}) & f(Q_{22}) \end{bmatrix} \begin{bmatrix} x_2y_2 & -y_2 & -x_2 & 1 \\ -x_2y_1 & y_1 & x_2 & -1 \\ -x_1y_2 & y_2 & x_1 & -1 \\ x_1y_1 & -y_1 & -x_1 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ x \\ y \\ xy \end{bmatrix}.$$

On the unit square

If we choose a coordinate system in which the four points where f is known are (0, 0), (1, 0), (0, 1), and (1, 1), then the interpolation formula simplifies to

$$f(x,y) \approx f(0,0)(1-x)(1-y) + f(1,0)x(1-y) + f(0,1)(1-x)y + f(1,1)xy,$$

or equivalently, in matrix operations:

$$f(x,y) \approx \begin{bmatrix} 1-x & x \end{bmatrix} \begin{bmatrix} f(0,0) & f(0,1) \\ f(1,0) & f(1,1) \end{bmatrix} \begin{bmatrix} 1-y \\ y \end{bmatrix}.$$

Here we also recognize the weights:

$$\begin{aligned} w_{11} &= (1-x)(1-y), \\ w_{12} &= (1-x)y, \\ w_{21} &= x(1-y), \\ w_{22} &= xy. \end{aligned}$$

Alternatively, the interpolant on the unit square can be written as

$$f(x,y) = a_{00} + a_{10}x + a_{01}y + a_{11}xy,$$

where

$$\begin{aligned}
a_{00} &= f(0,0),\\
a_{10} &= f(1,0) - f(0,0),\\
a_{01} &= f(0,1) - f(0,0),\\
a_{11} &= f(1,1) - f(1,0) - f(0,1) + f(0,0).
\end{aligned}$$

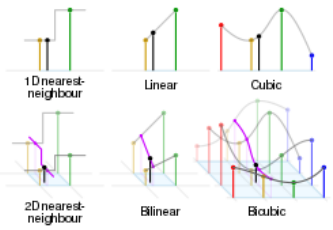
In both cases, the number of constants (four) correspond to the number of data points where f is given.

Properties

As the name suggests, the bilinear interpolant is *not* linear; but it is linear (i.e. affine) along lines parallel to either the x or the y direction, equivalently if x or y is held constant. Along any other straight line, the interpolant is quadratic. Even though the interpolation is *not* linear in the position (x and y), at a fixed point it *is* linear in the interpolation values, as can be seen in the (matrix) equations above.

The result of bilinear interpolation is independent of which axis is interpolated first and which second. If we had first performed the linear interpolation in the y direction and then in the x direction, the resulting approximation would be the same.

The interpolant is a multilinear polynomial in the coordinates, and a harmonic function.



Comparison of bilinear interpolation with some 1- and 2-dimensional interpolations. Black and red/yellow/green/blue dots correspond to the interpolated point and neighbouring samples, respectively. Their heights above the ground correspond to their values.

Inverse and generalization

In general, the interpolant will assume any value (in the convex hull of the vertex values) at an infinite number of points (forming branches of hyperbolas[2]), so the interpolation is not invertible.

However, when bilinear interpolation is applied to two functions simultaneously, such as when interpolating a vector field, then the interpolation is invertible (under certain conditions). In particular, this inverse can be used to find the "unit square coordinates" of a point inside any convex quadrilateral (by considering the coordinates of the quadrilateral as a vector field which is bilinearly interpolated on the unit square). Using this procedure bilinear interpolation can be extended to any convex quadrilateral.[3] The resulting map between quadrilaterals is known as a *bilinear transformation*, *bilinear warp* or *bilinear distortion*.

Alternatively, a projective mapping between a quadrilateral and the unit square may be used, but the resulting interpolant will not be bilinear.

In the special case when the quadrilateral is a paralellogram, a linear mapping to the unit square exists and the generalization follows easily.

The obvious extension of bilinear interpolation to three dimensions is called trilinear interpolation.

Inverse computation

Let F be a vector field that is bilinearly interpolated on the unit square parameterized by $\mu, \lambda \in [0,1]$. Inverting the interpolation requires solving a system of two bilinear polynomial equations:

$$A + B\lambda + C\mu + D\lambda\mu = 0$$

where

$$\begin{aligned}
A &= F_{00} - F\\
B &= F_{10} - F_{00}\\
C &= F_{01} - F_{00}\\
D &= F_{11} - F_{01} - F_{10} + F_{00}
\end{aligned}$$

Taking a 2-d cross product (see Grassman product) of the system with a carefully chosen vectors allows us to eliminate terms:

$$\begin{aligned}
(A + B\lambda + C\mu) \times D &= 0\\
(A + B\lambda) \times (C + D\lambda) &= 0\\
(A + C\mu) \times (B + D\mu) &= 0
\end{aligned}$$

which expands to

$$\begin{aligned}
c + e\lambda + f\mu &= 0\\
b + (c + d)\lambda + e\lambda^2 &= 0\\
a + (c - d)\mu + f\mu^2 &= 0
\end{aligned}$$

where

$$\begin{aligned}
a &= A \times B\\
b &= A \times C \qquad d = B \times C\\
c &= A \times D \qquad e = B \times D \qquad f = C \times D
\end{aligned}$$

The quadratic equations can be solved using the quadratic formula. We have the equivalent determinants

$$\mathbb{D} = (c + d)^2 - 4eb = (c - d)^2 - 4fa$$

and the solutions

$$\lambda = \frac{-c - d \pm \sqrt{\mathbb{D}}}{2e} \qquad \mu = \frac{-c + d \mp \sqrt{\mathbb{D}}}{2f}$$

(opposite signs are enforced by the linear relation). The cases when $e = 0$ or $f = 0$ must be handled separately. Given the right conditions, one of the two solutions should be in the unit square.

Application in image processing

In computer vision and image processing, bilinear interpolation is used to resample images and textures. An algorithm is used to map a screen pixel location to a corresponding point on the texture map. A weighted average of the attributes (color, transparency, etc.) of the four surrounding texels is computed and applied to the screen pixel. This process is repeated for each pixel forming the object being textured.[4]

When an image needs to be scaled up, each pixel of the original image needs to be moved in a certain direction based on the scale constant. However, when scaling up an image by a non-integral scale factor, there are pixels (i.e., *holes*) that are not assigned appropriate pixel values. In this case, those *holes* should be assigned appropriate RGB or grayscale values so that the output image does not have non-valued pixels.

Bilinear interpolation can be used where perfect image transformation with pixel matching is impossible, so that one can calculate and assign appropriate intensity values to pixels. Unlike other interpolation techniques such as nearest-neighbor interpolation and bicubic interpolation, bilinear interpolation uses values of only the 4 nearest pixels, located in diagonal directions from a given pixel, in order to find the appropriate color intensity values of that pixel.

Bilinear interpolation considers the closest 2 × 2 neighborhood of known pixel values surrounding the unknown pixel's computed location. It then takes a weighted average of these 4 pixels to arrive at its final, interpolated value. [5][6]

Example

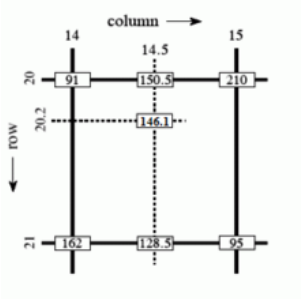
As seen in the example on the right, the intensity value at the pixel computed to be at row 20.2, column 14.5 can be calculated by first linearly interpolating between the values at column 14 and 15 on each rows 20 and 21, giving

$$I_{20,14.5} = \frac{15 - 14.5}{15 - 14} \cdot 91 + \frac{14.5 - 14}{15 - 14} \cdot 210 = 150.5,$$
$$I_{21,14.5} = \frac{15 - 14.5}{15 - 14} \cdot 162 + \frac{14.5 - 14}{15 - 14} \cdot 95 = 128.5,$$

and then interpolating linearly between these values, giving

$$I_{20.2,14.5} = \frac{21 - 20.2}{21 - 20} \cdot 150.5 + \frac{20.2 - 20}{21 - 20} \cdot 128.5 = 146.1.$$

This algorithm reduces some of the visual distortion caused by resizing an image to a non-integral zoom factor, as opposed to nearest-neighbor interpolation, which will make some pixels appear larger than others in the resized image.



Example of bilinear interpolation in grayscale values

See also

- Bicubic interpolation
- Trilinear interpolation
- Spline interpolation
- Lanczos resampling
- Stairstep interpolation
- Barycentric coordinates - for interpolating within a triangle or tetrahedron

References

1. Press, William H.; Teukolsky, Saul A.; Vetterling, William T.; Flannery, Brian P. (1992). *Numerical recipes in C: the art of scientific computing* (https://archive.org/details/numericalrecipes0865unse/page/123) (2nd ed.). New York, NY, USA: Cambridge University Press. pp. 123-128 (https://archive.org/details/numericalrecipes0865unse/page/123). ISBN 0-521-43108-5.
2. Monasse, Pascal (2019-08-10). "Extraction of the Level Lines of a Bilinear Image" (https://www.ipol.im/pub/art/2019/269/). *Image Processing On Line*. **9**: 205–219. doi:10.5201/ipol.2019.269 (https://doi.org/10.5201%2Fipol.2019.269). ISSN 2105-1232 (https://www.worldcat.org/issn/2105-1232).
3. Quilez, Inigo (2010). "Inverse bilinear interpolation" (https://www.iquilezles.org/www/articles/ibilinear/ibilinear.htm). *iquilezles.org*. Retrieved 2021-07-28.
4. Bilinear interpolation definition (popular article on www.pcmag.com (https://www.pcmag.com/encyclopedia_term/0,2542,t=bilinear+interpolation&i=38607,00.asp).
5. Khosravi, M. R. (2021-03-19). "BL-ALM: A Blind Scalable Edge-Guided Reconstruction Filter for Smart Environmental Monitoring Through Green IoMT-UAV Networks" (https://ieeexplore.ieee.org/document/9382001). *IEEE Transactions on Green Communications and Networking*. **5**: 727–736. doi:10.1109/TGCN.2021.3067555 (https://doi.org/10.1109%2FTGCN.2021.3067555).
6. "Web tutorial: Digital Image Interpolation" (http://www.cambridgeincolour.com/tutorials/image-interpolation.htm).