

Full-name: Phan Thi My Dung

Lecturer: Lee Chil Woo

Class: Advance in Pattern Recognition

BILINEAR INTERPOLATION

1. What is bilinear interpolation?

Bilinear Interpolation is a resampling method that uses the distance-weighted average of the four nearest pixel values to estimate a new pixel value.

The four cell centers from the input raster are closest to the cell center for the output processing cell will be weighted and based on the distance and then averaged.

Algorithm

Suppose that we want to find the value of the unknown function f at the point $P = (x, y)$. 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 $Q_{22} = (x_2, y_2)$.

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

$$f(R_1) \approx \frac{x_2 - x}{x_2 - x_1} f(Q_{11}) + \frac{x - x_1}{x_2 - x_1} f(Q_{21})$$

where $R_1 = (x, y_1)$

$$f(R_2) \approx \frac{x_2 - x}{x_2 - x_1} f(Q_{12}) + \frac{x - x_1}{x_2 - x_1} f(Q_{22})$$

where $R_2 = (x, y_2)$

We proceed by interpolation in the y – direction

$$f(P) \approx \frac{y_2 - y}{y_2 - y_1} f(R_1) + \frac{y - y_1}{y_2 - y_1} f(R_2)$$

This gives us the desired estimate of $f(x, y)$

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

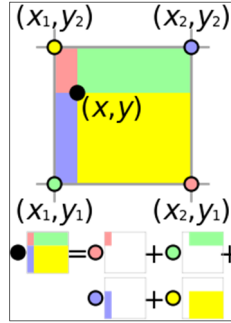


Figure 1: In this example, the value at the black pint is the sum of the value at each other's multiplied by the area of the rectangle of the same color, divided by the total area of all 4 rectangles

Example: 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

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

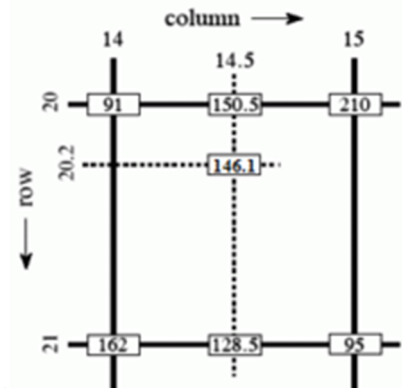


Figure 2: Example of bilinear interpolation in grayscale values.

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. Bilinear interpolation tends, however, to produce a greater number of interpolation artifacts (such as aliasing, blurring, and edge halos) than more computationally demanding techniques such as bicubic interpolation.

2. Application in Computer Vision and Image Processing

- In computer vision and image processing, bilinear interpolation is one of the basic resampling techniques.
- In texture mapping, it is also known as bilinear filtering or bilinear texture mapping and it can be used to produce a reasonably realistic image. 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, alpha, 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.
- In the Bi-linear interpolation, intensity of the zoomed image pixel 'P' is defined by the weighted sum of the mapped 4 neighboring pixels. If the zooming factor is 's', then the mapped pixel point in the original image is given by 'r' and 'c' as follows.

$$r = \text{floor}\left(\frac{x}{s}\right) \quad \text{and} \quad c = \text{floor}\left(\frac{y}{s}\right)$$

And the distance from the point of interest to the mapped pixels can be obtain as follows,

$$x_d, y_d = \{ \text{rem}\left(\frac{x}{s}, 1\right), \text{rem}\left(\frac{y}{s}, 1\right) \}$$

So the neighbouring 4 pixels can be defined as,







$$P_{topLft}, P_{topRht}, P_{botLft}, P_{botRht} = \{ f(r, c), f(r + 1, c), f(r, c + 1), f(r + 1, c + 1), \}$$

By using these values we can approximate the intensity of the pixel in interest as follows.

$$v(x, y) = (1 - y_d)(x_d \cdot P_{topRht} + (1 - x_d) \cdot P_{topLft}) + y_d(x_d \cdot P_{botRht} + (1 - x_d) \cdot P_{botLft})$$

Source code:

```
function image_zoom = blnrm2(image, zoom)
[r c d] = size(image);
rn = floor(zoom*r);
cn = floor(zoom*c);
s = zoom;
im_zoom = zeros(rn,cn,d);
for i = 1:rn;
    x1 = cast(floor(i/s), 'uint32');
    x2 = cast(ceil(i/s), 'uint32');
    if x1 == 0
        x1 = 1;
    end
    x = rem(i/s,1);
    for j = 1:cn;
        y1 = cast(floor(j/s), 'uint32');
        y2 = cast(ceil(j/s), 'uint32');
        if y1 == 0
            y1 = 1;
        end
        ctl = image(x1,y1,:);
        cbl = image(x2,y1,:);
        ctr = image(x1,y2,:);
        cbr = image(x2,y2,:);
        y = rem(j/s,1);
        tr = (ctr*y)+(ctl*(1-y));
        br = (cbr*y)+(cbl*(1-y));
        im_zoom(i,j,:) = (br*x)+(tr*(1-x));
    end
end
image_zoom = cast(im_zoom, 'uint8');
end
```

Zoom rate	Zoomed Image	
0.5		
Original Image		
1.5		
2.0		
2.5		
3.0		

An example of scaling image with different rate scale

References:

http://en.wikipedia.org/wiki/Bilinear_interpolation

http://pippin.gimp.org/image_processing/chap_resampling.html

http://thilinasameera.wordpress.com/2010/12/24/digital-image-zooming-sample-codes-on-matlab/#_Bilinear_Interpolation