

Re: Scale a vector

Source:

<http://www.tech-archive.net/Archive/DotNet/microsoft.public.dotnet.languages.csharp/2007-09/msg01885.html>

- *From:* "Matteo Migliore" <matteo.migliore.cut@xxxxxxxxxx>
 - *Date:* Mon, 17 Sep 2007 02:27:39 +0200
-

Peter Duniho wrote:

Matteo Migliore wrote:

[...]

But no, I need to resize a one-dimensional array.

For what it's worth, you still haven't provided any guidance regarding how you want the data in the array to be calculated. Is that because you yourself do not know?

I arrived to the conclusion that I can use the straight line equation to interpolate value in X1 to X2 and value X3 to X4 etc... If the size parameter is greater than the width of the original bitmap.

[...]

This function calculate the middle luminance for each column (X coordinate) in the image and return an array of middle luminances, so the array lenght is the width of the image.

Same thing for the vertical projection, but the calculate is on rows!

Ok, now I've to compare horizontal RGB projection for image A, B and C. The array lenght must be the same, so I've to scale RGB proj. of A to 100 elements,

RGB proj. of B to 100 elements and the same for C. 100 is random selected, it could be 640

(the most minum width for an image). If the image width is smaller than 100px the function must

Scale the array to 640.

I am not clear on what you mean by "middle". But, whether that's the median value of the column or row of pixels, the average, or literally the value of the middle pixel, I think the solution is basically the

Re: Scale a vector

same.

Sorry, I mean the median luminosity value for each column for horizontal RGB and for each row for vertical RGB.

What you are looking for is essentially a one-dimensional image scaling. So, a couple of thoughts come to mind:

1) You may find that it makes more sense to scale the input bitmaps first, and then do the comparison on data calculated from the scaled input. This is less efficient because you wind up scaling data that you don't necessarily use, but

a) it's not clear even now from your description that you really don't want to use the data (even though you may believe that you don't, it's possible that because you are scaling your data, you really do want the scaling to take into account all neighboring pixels, not just those in a specific direction), and

b) because it makes the implementation of your solution simpler, the reduced efficiency may be a worthwhile price for simpler code.

No I can't scale images because performance are the key, so I've to scale arrays and store them in a Dictionary to compare them.

So I can't resize images, that was my first idea :-).

2) If you don't want to scale the input first, I would say that instead of treating the data as an array, you should generate new bitmaps, as wide or high as the input bitmap, and one pixel in size in the other direction. This way, you can take advantage of the built-in .NET image processing functionality. How best to do this would depend on what is meant by "middle".

In option #2, if you are literally taking a the middle pixel from a column or row, then you should just copy a complete row or column (respectively) of pixels into a new bitmap and then scale that bitmap and extract the luminosity from the resulting bitmap. If you have some other meaning of "middle", you'll have to convert your luminosity data into a format that can be treated as a regular Bitmap instance (for example, a plain 24-bit-per-pixel RGB Bitmap, where each pixel is a gray value based on the luminosity value). Then you just scale that bitmap in a single dimension and extract back out the values (which should still be gray values, making it trivial to convert back to luminosity).

In either case, the scaling will be done with the Bitmap and Graphics

Re: Scale a vector

classes. You'll create two Bitmap instances: the input one and the output one. Then you'll use Graphics.FromImage() to get a Graphics instance for the output Bitmap instance. Then you'll use the Graphics.DrawImage() method to copy the data from the input Bitmap to the output Bitmap, providing source and destination rectangles that correspond to the full size of the input and output Bitmaps, respectively. The DrawImage() method will then apply some image-scaling algorithm to the data; which one is specifically used will depend on the InterpolationMode property of the Graphics instance used for the drawing.

No I can't :-). So I've to scale the arrays (I know, I'm repetitive :-D)

The straight line equation and interpolation is a good mode, but it not simple. But I think that is the only way :-).

I found this article on CodeProject:

http://www.codeproject.com/useritems/Douglas-Peucker_Algorithm.asp

This algorithm solve a problem similar to mine, but it only *reduce* points number and accept a tollerance parameter, not how many points to have :-).

Hope that helps.

Pete

I don't know how to thank you!!
(But my problem remain).

I'll upload my solution on CodePlex ;-)

It's an application to find duplicated images :-D
Matteo Migliore.

.