Note on Geometric Interpolation of Images

Chee Yap and Ting-Jen Yen
Courant Institute, NYU
{yap,yentj}cs.nyu.edu

September 5, 2000

Abstract

This is a note on constructing continuous images from discrete ones. We look at a class of geometric constructions that includes bilinear interpolation as a special case. An implementation and some experiments are described.

1 Geometric Scheme for Interpolating Images

In many applications, we need continuous images or at least an image with arbitrary fine resolution. In this note, a **continuous image** mean a function of the form $J : \mathbb{R}^2 \to [0,1]$. But in practice, images are discrete, typically with domain given by the integer grid $\mathbb{Z}^2$. There are many standard ways to construct a continuous image from a discrete image. Two of the most common methods are bilinear interpolation and convolution with a Gaussian kernel. In this note, we consider a class of "geometric" constructions. An implementation and some simple experiments are described.

Let I be a discrete (greyscale) image,

$$I : \mathbb{Z}^2 \to [0,1].$$

We want to define a transformation that maps I to a continuous image $J : \mathbb{R}^2 \to [0,1]$ with the following properties:
(1) If I is a constant function $I = c$, then J is the constant function $J = c$.
(2) J is a local function of I. That means that the value of $J(x,y)$ depends on the values of I within some bounded (Euclidean) distance from (x,y).
(3) J is a continuous[1] function.

If we omit property (3), then we can define a continuous image J_0 as follows:

$$J_0(x,y) = I(\lfloor x + (1/2) \rfloor, \lfloor y + (1/2) \rfloor).$$

[1]We are using "continuous" here in the standard mathematical sense; on the other hand, "continuous image J" means that the domain of J is the entire plane.

This is the standard interpretation of discrete images used by graphics display systems. Property (2) excludes the use of convolution using the Gaussian kernel. Perhaps the simplest solution that satisfies all 3 properties is J_1, obtained through bilinear interpolation. That is, for all $(i,j) \in \mathbb{Z}^2$, the function J_1 restricted to the unit square $(i,j) + [0,1]^2$ is given by the unique bilinear function whose value is determined by I at the four points $(i,j),(i,j+1),(i+1,j),(i+1,j+1)$ where $i = \lfloor x \rfloor, j = \lfloor y \rfloor$, such that J agrees with I on all the grid points of $\mathbb{Z}^2$.

We can view J_1 as the convolution of J_0 with the "box function", $B(x,y) = 1$ if $|x| \leq 1/2$ and $|y| \leq 1/2$, and $B(x,y) = 0$ otherwise. Let us generalize this view point. Let $S \subseteq \mathbb{R}^2$ be a set. We say S is a **centrally symmetric** if it satisfies the properties that $p \in S$ implies $-p \in S$. We call S a **body** if it is convex and centrally symmetric. Define the function $B_S(x,y) = 1/|S|$ if $(x,y) \in S$ and $B_S(x,y) = 0$ otherwise. Now, we define the continuous image J_S to be the convolution of J_0 with B_S. We describe this class of transformations

$$I \mapsto J_S$$

as "geometric" because the transformation amounts to specifying a geometric shape S.

In case S is a disc, we obtain a scheme that has no preferred direction. This function smooths the "jaggy edges" seen in J_0. Another interesting case is where S is the diamond shape: this has two preferred directions, but it may be a good way to counteract the preferred directions of the function J_0.

2 Implementation and Experiments.

We implemented the case where $S = S_1$ is the disc with unit area (with radius $1/\sqrt{\pi} = 0.564189...$). As illustrated in figure 1, the color at each grid point $(i,j) \in \mathbb{Z}^2$ will have a small contribution from its four neighbors $(i \pm 1, j \pm 1)$. The weight of the contribution from each of the neighbors is just over 2 percent (the area of overlap between S_1 placed at (i,j) with each neighbor is 0.022...).

This algorithm for S_1 is relatively simple and fast because of the small size of S_1 – this cuts down on the number of possible cases. In implementation, we can exploit symmetry to further reduce the number of cases: we code up the transformation for only one of the four quadrants of S_1, and exploit it 4 times. An alternative way to decomposing the problem, to split up each pixel into 4 quadrants leads to a slightly more involved case analysis. The source code for our implementation is found in this directory. Our algorithm for S_1 may be a useful alternative to bilinear interpolation.

Extension by Iteration. A more general algorithm, in which S can be an arbitrary size disc, can similarly be constructed. Such an algorithm would be useful for larger scale blurring. However, such an extension would significantly increase the complexity

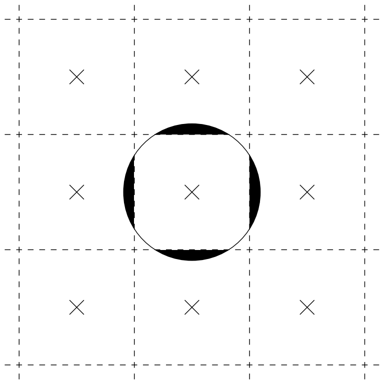

Figure 1: The unit disc S_1 centered at a grid point

of the implementation. We explore an alternative method based on iterating our special case.

Our implementation takes an $n \times m$ image and an integer parameter $r \geq 1$, and output an $2^r n \times 2^r m$ image. This output is the discretized version of the mathematically continuous image. To iterate, we first reduce the output from $2^r n \times 2^r m$ back to a $n \times n$ image by replacing each $2^r \times 2^r$ block of pixels by a single pixel with its average value, and then apply the algorithm. This can be repeated several times.

The images that accompany this note show the result of iteration: the original 62×55 image (a small part of a map) is found in the file `map0.tiff`. Using $r = 3$, the result of 3 iterations of our transformations are the 496×440 images found in `map1.tiff`, `map2.tiff`, and `map3.tiff`. Observe that in the last iterate `map3.tiff`, the graininess of the green texture has been removed and the jagged form of edges has also disappeared. The resolution of most of the lines is pretty much intact. The red letterings, having started with a poor initial quality, suffered more from this blurring.

As empirically observed, the rate of diffusion is extremely slow in this iteration process. This property is useful in preprocessing images.

There are obviously many unanswered questions. Are there nice mathematical properties of these class of geometric transformations? The iteration process should be better understood. How does this compare to Gaussian blurring?