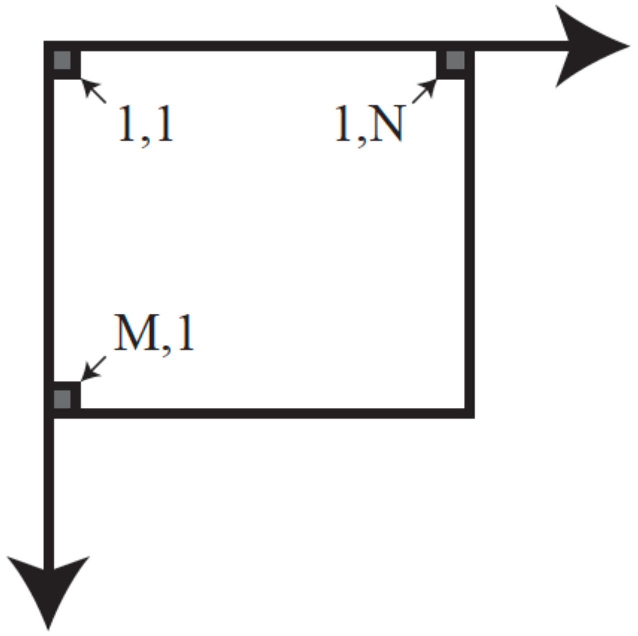
Upsampling images

D.A. Forsyth,

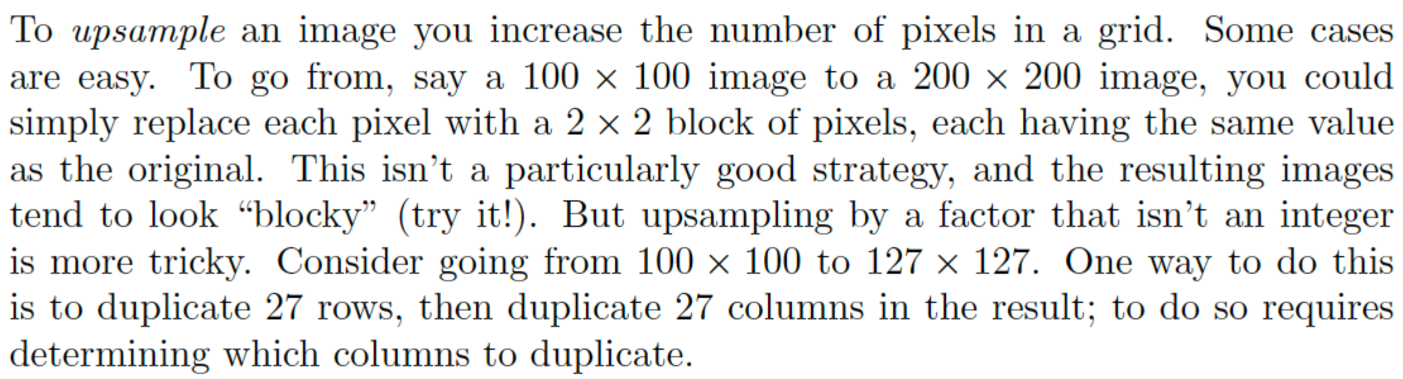
University of Illinois at Urbana Champaign

Key idea

- Sampled images have values on an integer grid
- We know
  - I_{1, 1}, I_{1, 2} and so on
  - First coordinate runs 1...M, second 1..N



The simplest strategies …



To *upsample* an image you increase the number of pixels in a grid. Some cases are easy. To go from, say a 100×100 image to a 200×200 image, you could simply replace each pixel with a 2×2 block of pixels, each having the same value as the original. This isn't a particularly good strategy, and the resulting images tend to look "blocky" (try it!). But upsampling by a factor that isn't an integer is more tricky. Consider going from 100×100 to 127×127. One way to do this is to duplicate 27 rows, then duplicate 27 columns in the result; to do so requires determining which columns to duplicate.

- Big issue:
  - what is the value of a pixel at a non-integer location?

Forward warping

- Scan the source image S and place each pixel in appropriate location in the target image T

- BUT
  - if you're upsampling, there will be holes!
    - actually, mostly holes
- Never do this
  - or hardly ever; there may be some cases…

Inverse warping

Alternatively, you might consider scanning the source (smaller - $\mathcal{S}$) image and, for each pixel, determining where it goes in the target (larger - $\mathcal{T}$) image. But there are more pixels in the target than in the source, so this approach must lead to holes in the predicted image. The correct alternative is to scan the target image and, for each pixel, determine what value it should receive. This is known as *inverse warping*. In the example, the i, j'th location of $\mathcal{T}$ must get the value of the $i/1.27$, $j/1.27$'th location of $\mathcal{S}$. In fact, most values requred are at locations that are not integer values.

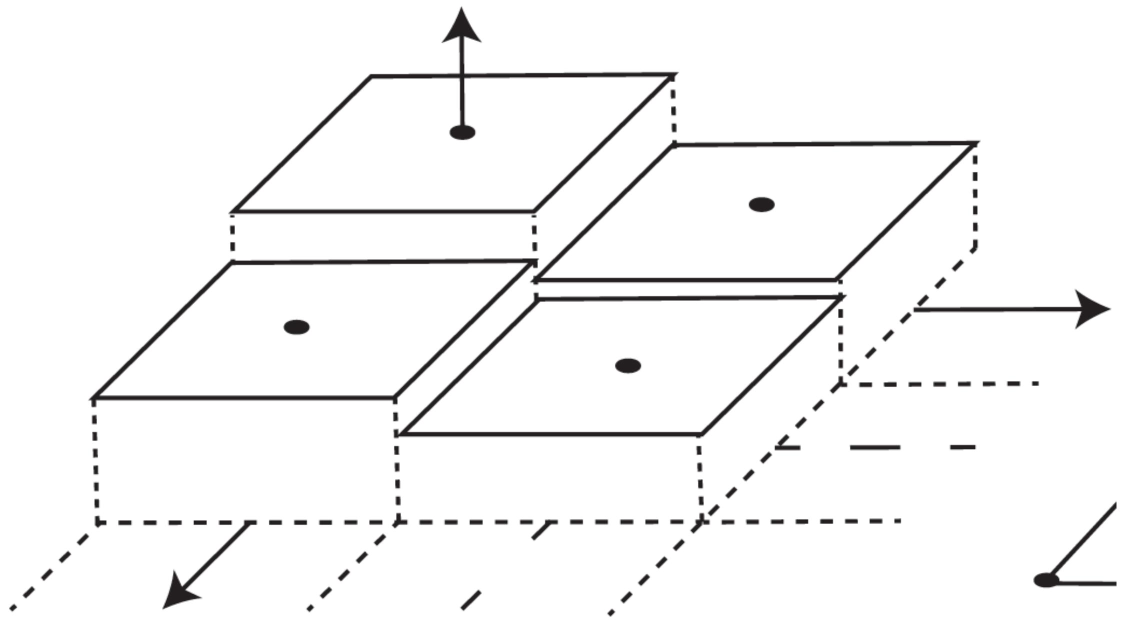
Interpolation

An *interpolate* is a function that (a) must have the same value as the original image at the original integer grid points (b) can be evaluated at any point rather than just the integer grid points. Write $\mathcal{I}(x, y)$ for an interpolate of an image $\mathcal{I}$.

The simplest interpolate is *nearest neighbors* – take the value at the integer point closest to location whose value you want. Break ties by rounding up, so you would use the value at $2, 2$ if you wanted the value at $1.5, 1.5$. As Figure 3.1 shows, this strategy has problems – the upsampled image looks blocky.

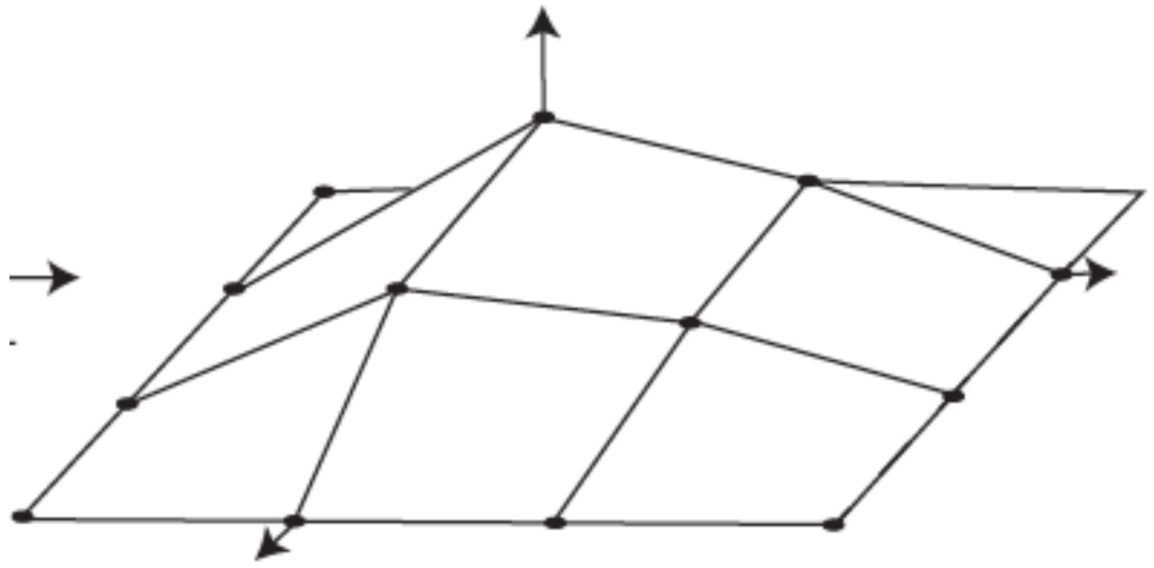
Nearest neighbors interpolation

- Source image goes from samples at grid points to function that looks like



An alternate – bilinear interpolation

- Source image goes from samples at grid points to function that looks like



Interpolation mechanics

There are many different ways to interpolate. Write $b(u, v)$ for a function that is one at the origin (so $b(0, 0) = 1$) and is zero at every other integer grid point. There are many such functions. For the moment, choose one. Then

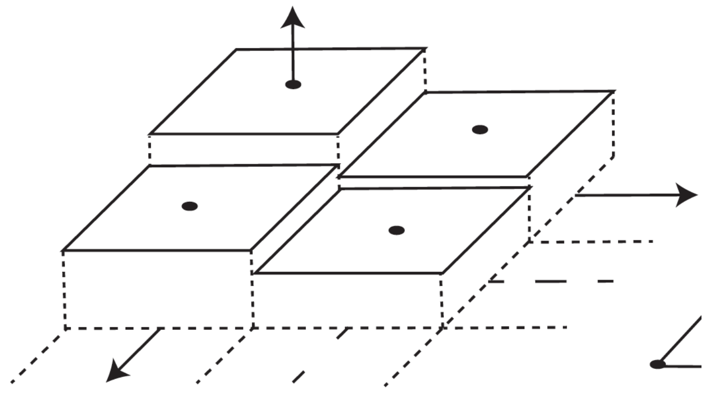
$$\mathcal{I}(x, y) = \sum_{ij} \mathcal{I}_{ij} b(x - i, y - j)$$

will be an interpolate (check you know why).

Nearest neighbors interpolation

For nearest neighbors, define

$$b_{nn}(u,v) = \begin{cases} 1 & \text{for } -1/2 \le u < 1/2 \text{ and } -1/2 \le u < 1/2 \\ 0 & \text{otherwise} \end{cases}$$

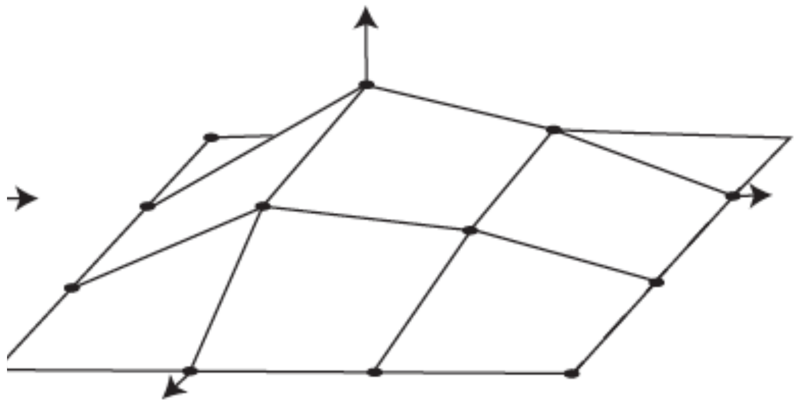


An alternate – bilinear interpolation

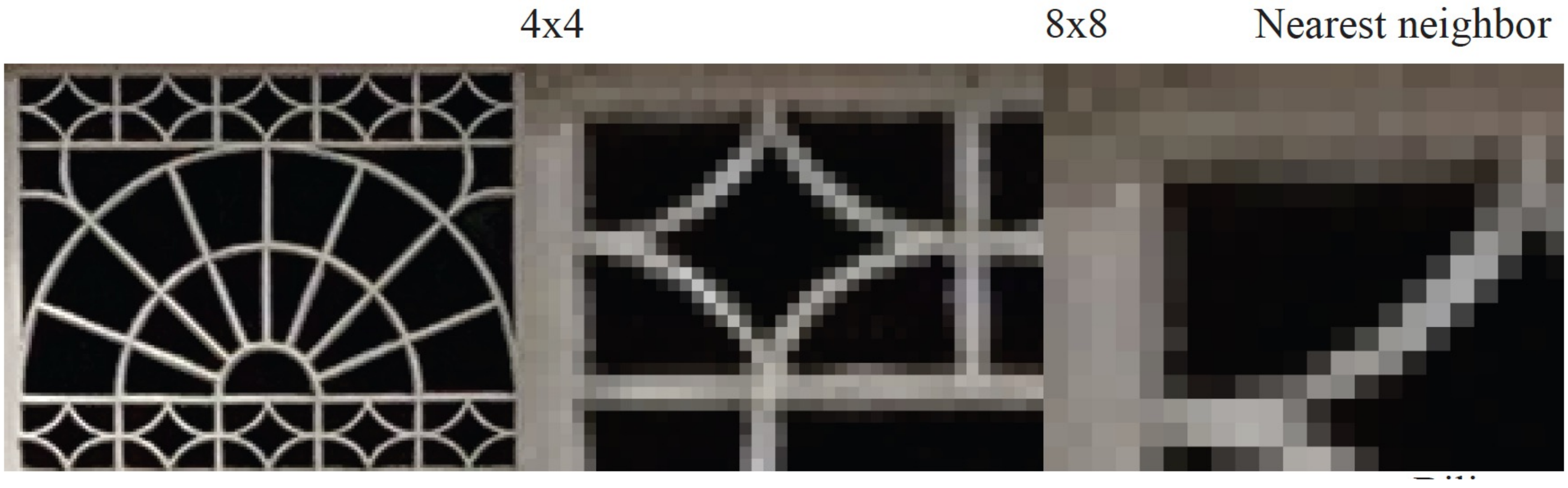
This fitted function looks like a collection of boxes, and is not continuous (Figure 3.1; **exercises** ()). Most widely used is *bilinear interpolation*. For this, construct a function

$$b_{bi}(u,v) = \begin{cases} (1-u)(1-v) & \text{for } 0 < u \leq 1 \text{ and } 0 < v \leq 1 \\ (1+u)(1-v) & \text{for } -1 \leq u \leq 0 \text{ and } 0 < v \leq 1 \\ (1+u)(1+v) & \text{for } -1 \leq u \leq 0 \text{ and } -1 \leq v \leq 0 \\ (1-u)(1+v) & \text{for } 0 < u \leq 1 \text{ and } -1 \leq v \leq 0 \\ 0 & \text{otherwise} \end{cases}$$

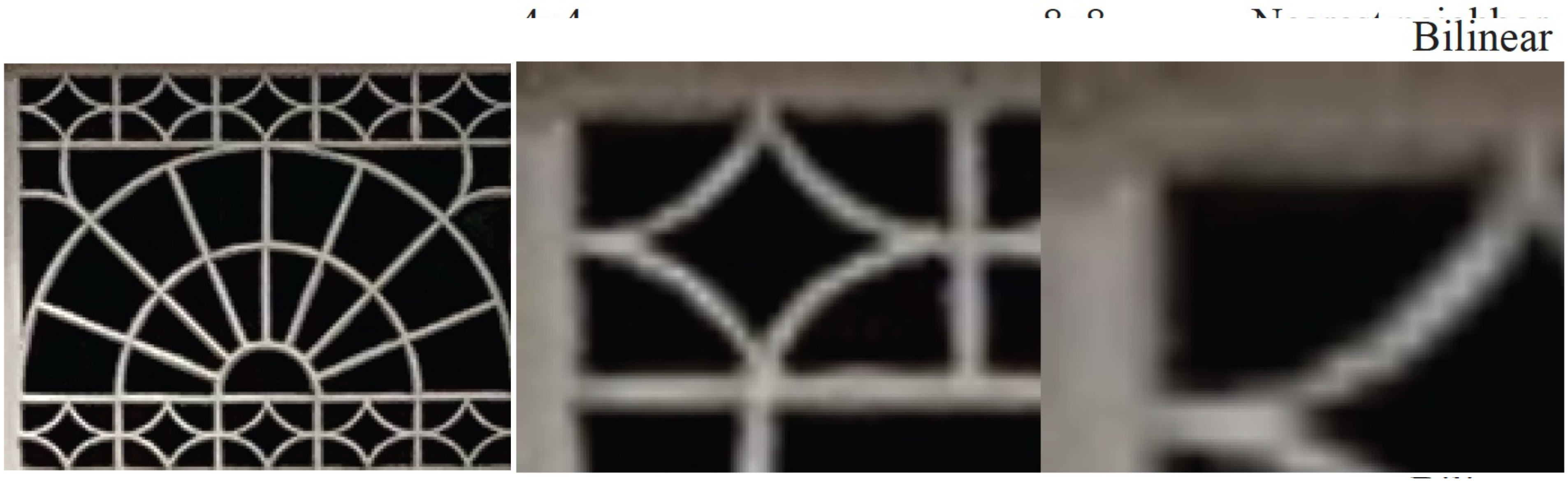
which is continuous, and again has the convenient property that $b_{bi}(0,0) = 1$, but $b_{bi} = 0$ for every other grid point (it looks a bit like a hat, Figure 3.1). The



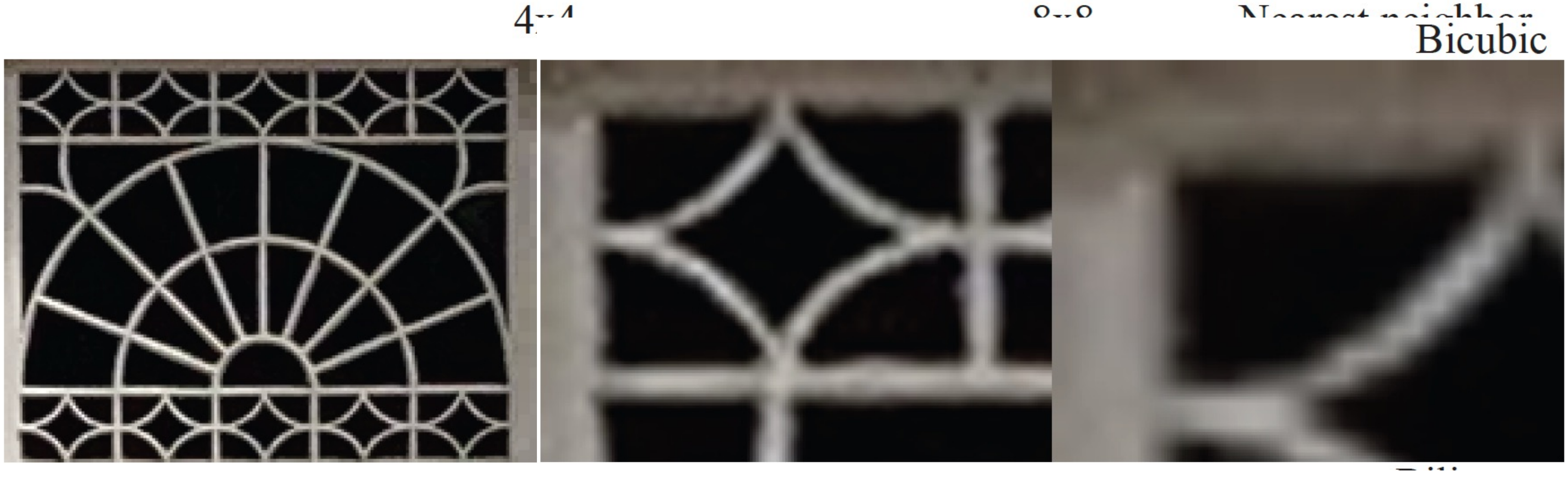
Choice of interpolate makes a difference



Choice of interpolate makes a difference



Choice of interpolate makes a difference



Things to think about...

3.1. Is a nearest neighbor interpolate continuous?

3.2. Is a bilinear interpolate continuous?

3.3. Define $b_{bi}(u,v) = f(u)f(v)$ where

$$f(u) = \begin{cases} \frac{(u-1)(u-2)(u+1)(u+2)}{4} & \text{for } -2 \leq u \leq 2 \\ 0 & \text{otherwise} \end{cases}.$$

Check that this an interpolate. Is this a useful interpolate? Why not?

3.4. Define $b_{sinc}(u,v) = f(u)f(v)$ where

$$f(u) = \frac{\sin \pi u}{u}.$$

Check that this an interpolate. Why is this not a useful interpolate for reconstructing images? Remember that

$$\lim_{x \to 0} \frac{\sin x}{x} = 0.$$