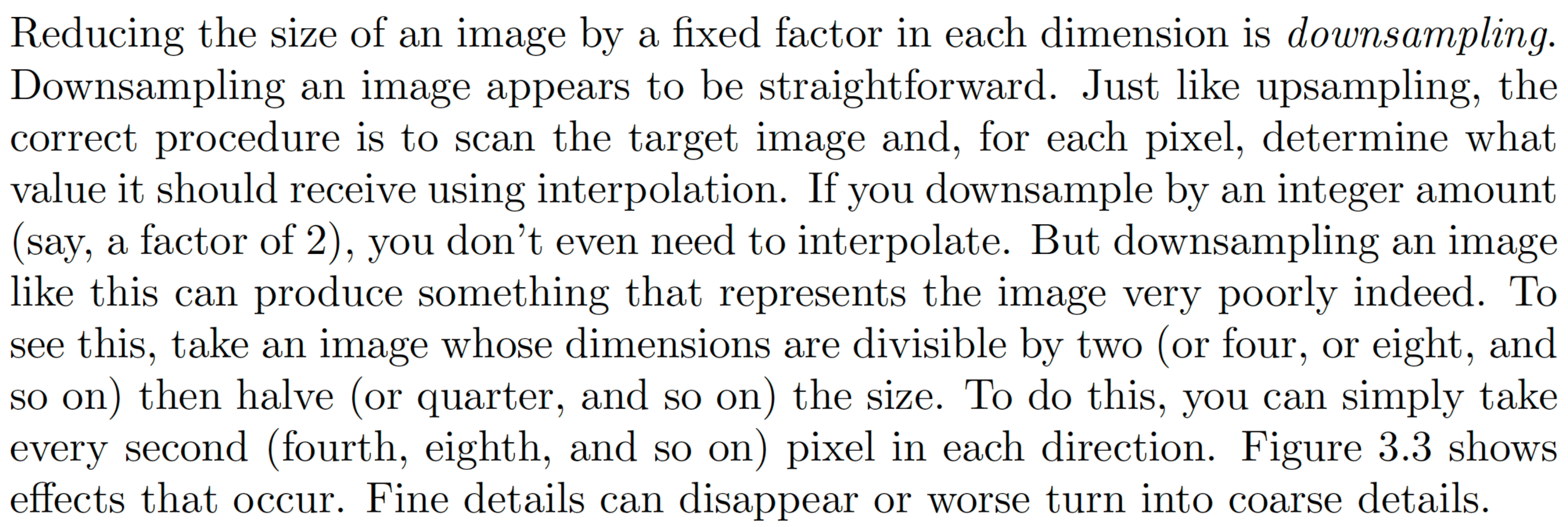
Smoothing and Downsampling Images

D.A. Forsyth,

University of Illinois at Urbana Champaign

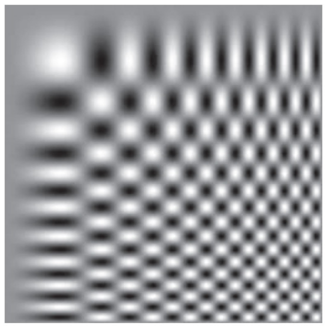
Doing the obvious

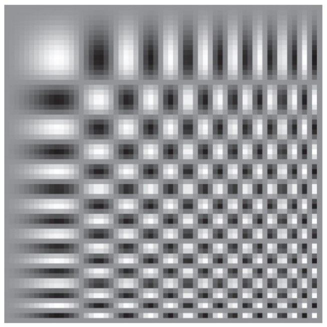
- The obvious:
  - Inverse warp, interpolating as required

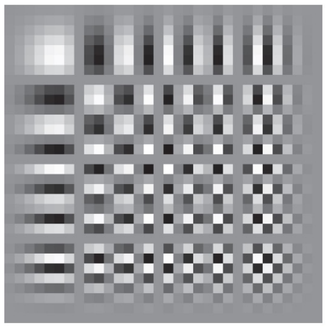


Reducing the size of an image by a fixed factor in each dimension is *downsampling*. Downsampling an image appears to be straightforward. Just like upsampling, the correct procedure is to scan the target image and, for each pixel, determine what value it should receive using interpolation. If you downsample by an integer amount (say, a factor of 2), you don't even need to interpolate. But downsampling an image like this can produce something that represents the image very poorly indeed. To see this, take an image whose dimensions are divisible by two (or four, or eight, and so on) then halve (or quarter, and so on) the size. To do this, you can simply take every second (fourth, eighth, and so on) pixel in each direction. Figure 3.3 shows effects that occur. Fine details can disappear or worse turn into coarse details.

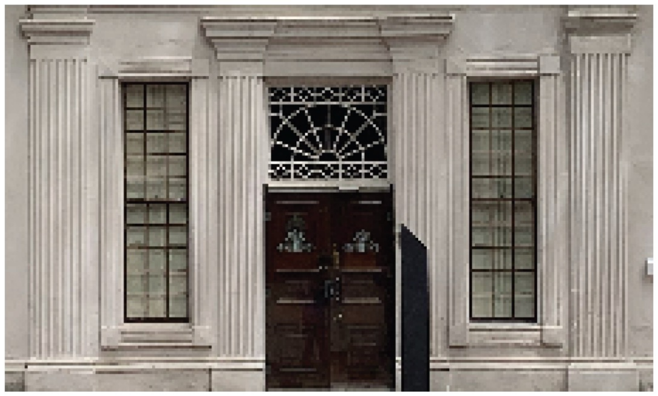
works badly!



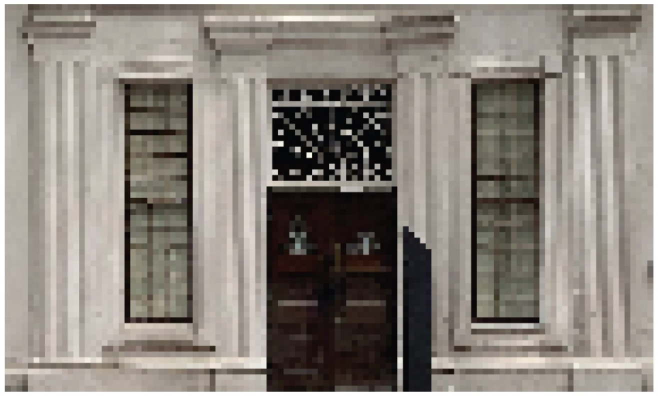




Image

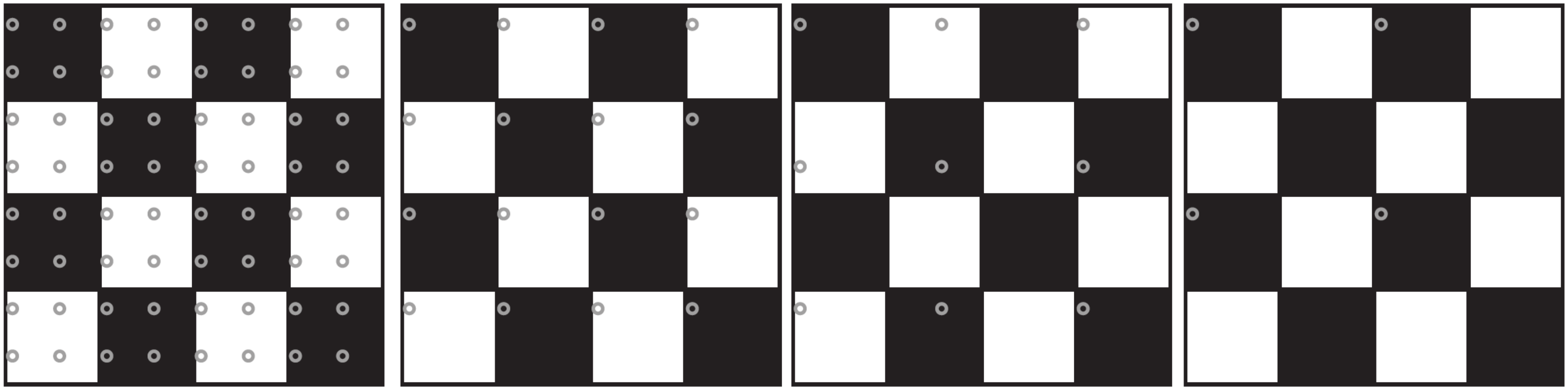


downsampled by 4

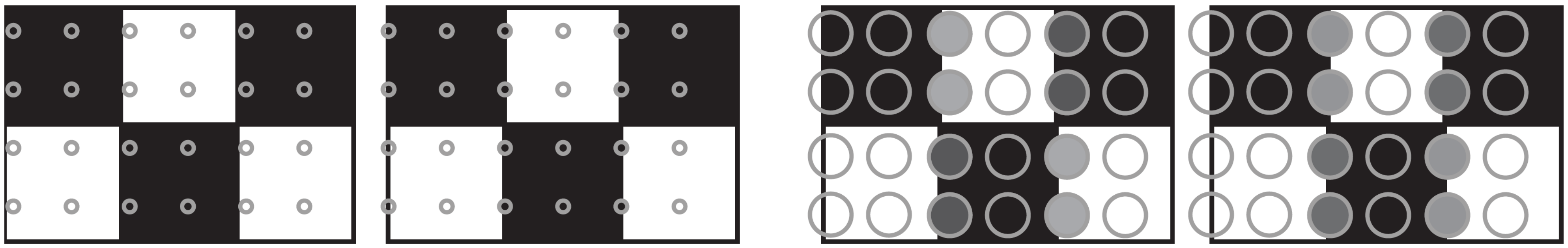


downsampled by 8

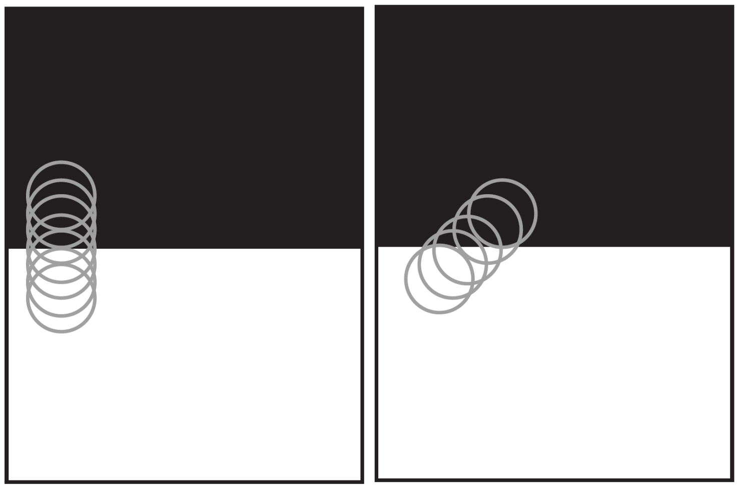
Underlying mechanism

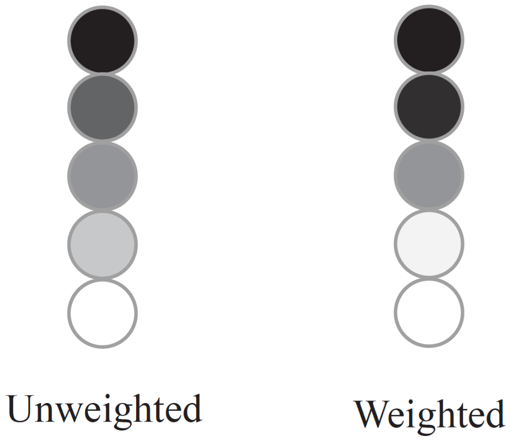


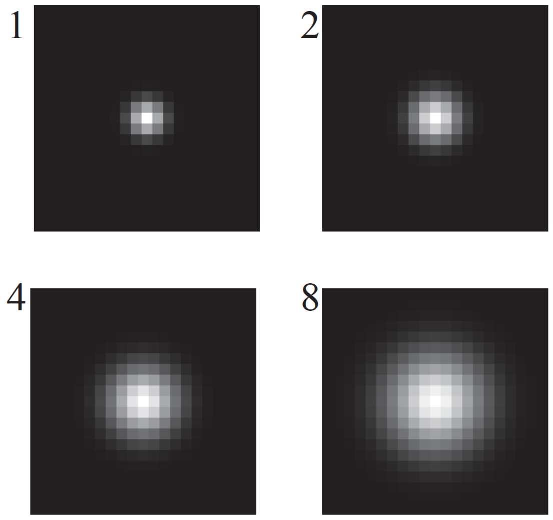
Pooling or averaging might help



Weighted average -> better representation



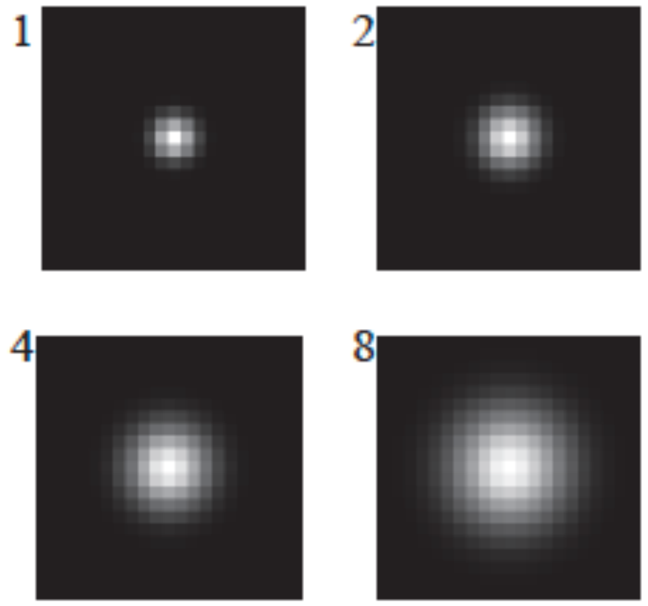




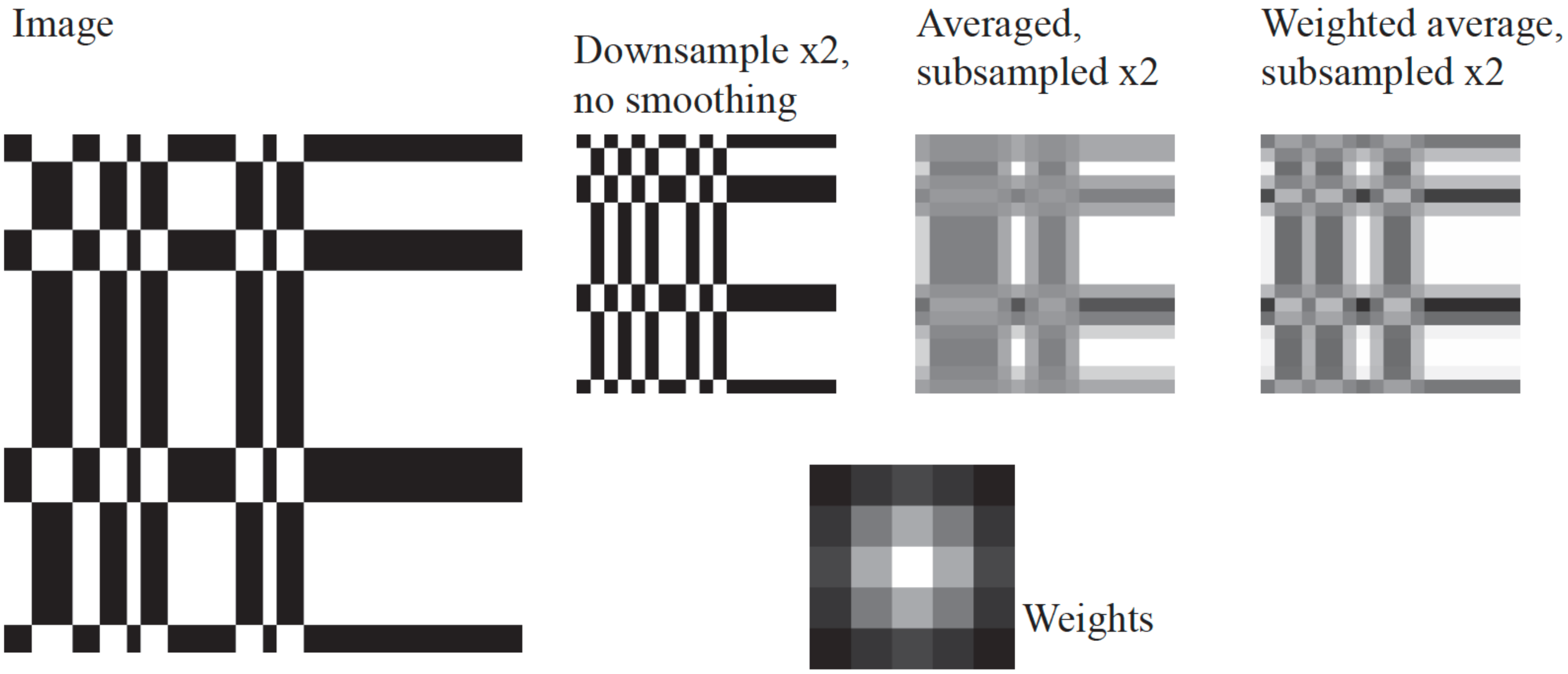
Gaussian weights

A traditional weighting scheme is given by a one parameter family of functions, derived from the normal distribution and widely called *gaussians*. The parameter σ is sometimes called the *scale* and more usually called the *sigma* of the weights. For downsampling by two, $\sigma = 1$ or $\sigma = 1.5$ are fair choices. In a $2k+1 \times 2k+1$ window, where the pixels are indexed starting at 1, the weights will be:

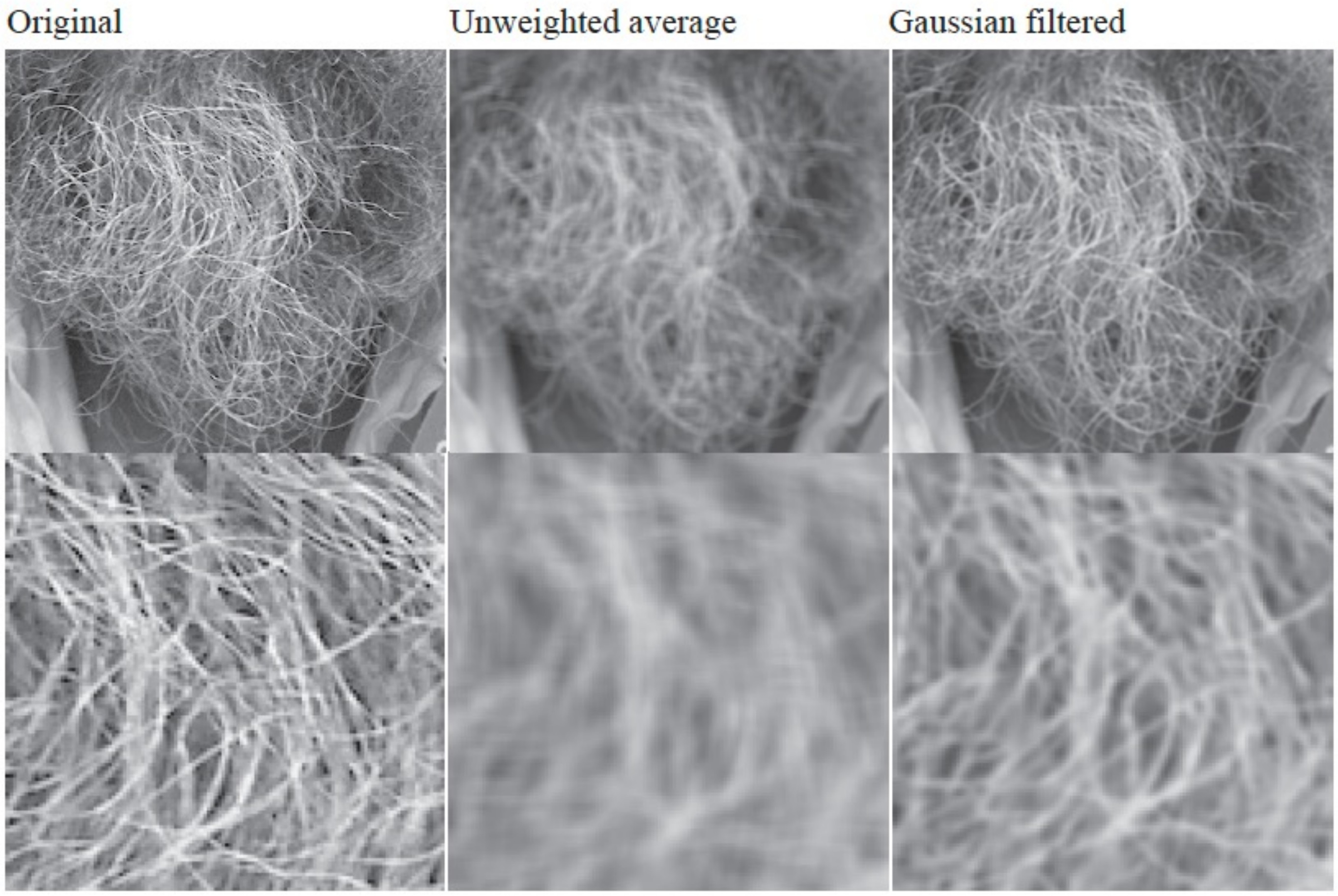
$$k_{ij} = \frac{1}{C} e^{-\left(\frac{(i-k-1)^2+(j-k-1)^2}{2\sigma^2}\right)}$$



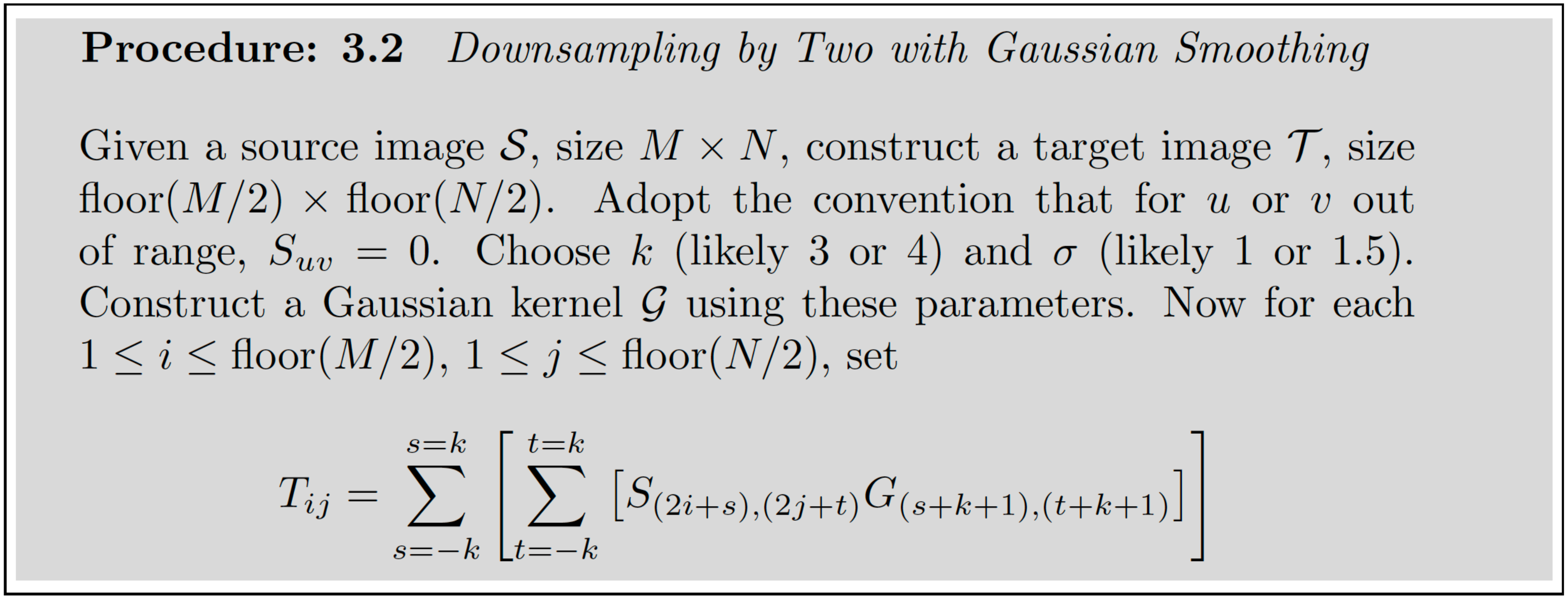
Just averaging is actually a bad idea



Just averaging is actually a bad idea



Easy case: downsample by two

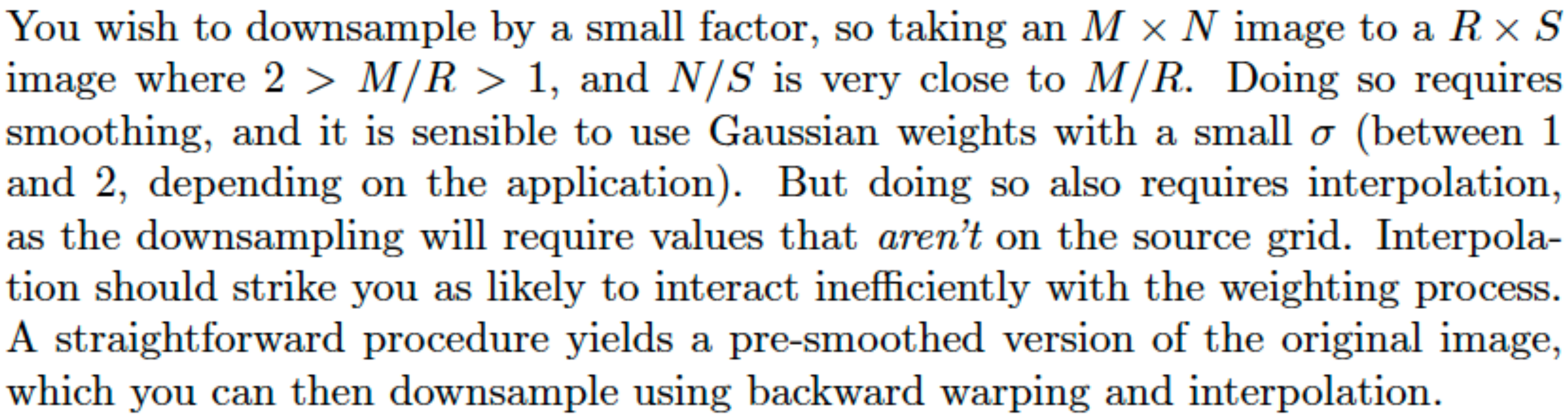


Procedure: 3.2 *Downsampling by Two with Gaussian Smoothing*

Given a source image $\mathcal{S}$, size $M \times N$, construct a target image $\mathcal{T}$, size floor$(M/2) \times$ floor$(N/2)$. Adopt the convention that for u or v out of range, $S_{uv} = 0$. Choose k (likely 3 or 4) and σ (likely 1 or 1.5). Construct a Gaussian kernel $\mathcal{G}$ using these parameters. Now for each $1 \leq i \leq$ floor$(M/2)$, $1 \leq j \leq$ floor$(N/2)$, set

$$T_{ij} = \sum_{s=-k}^{s=k} \left[\sum_{t=-k}^{t=k} \left[S_{(2i+s),(2j+t)} G_{(s+k+1),(t+k+1)} \right] \right]$$

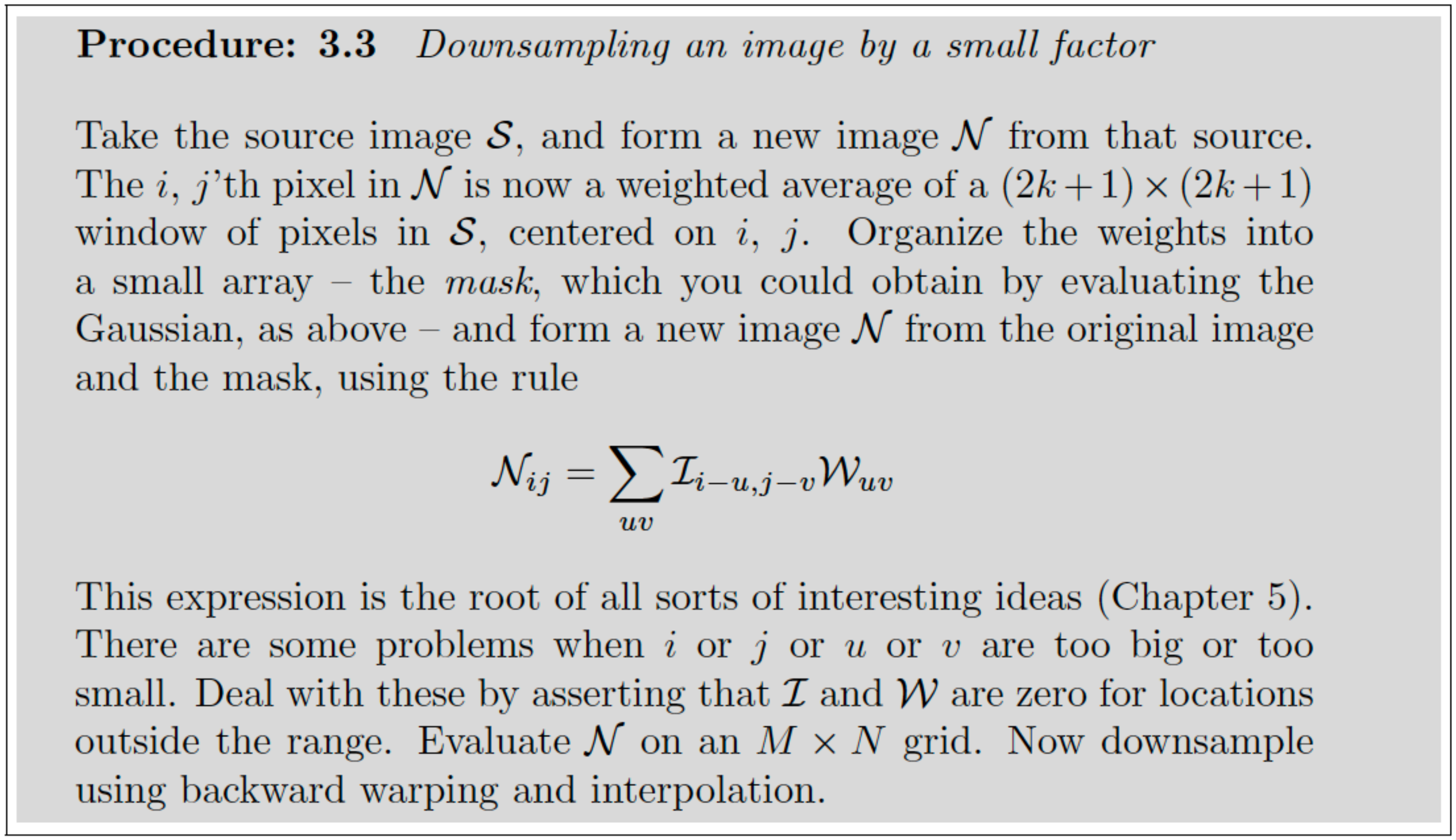
Downsample by small factor



You wish to downsample by a small factor, so taking an $M \times N$ image to a $R \times S$ image where $2 > M/R > 1$, and N/S is very close to M/R. Doing so requires smoothing, and it is sensible to use Gaussian weights with a small σ (between 1 and 2, depending on the application). But doing so also requires interpolation, as the downsampling will require values that *aren't* on the source grid. Interpolation should strike you as likely to interact inefficiently with the weighting process. A straightforward procedure yields a pre-smoothed version of the original image, which you can then downsample using backward warping and interpolation.

Downsample by small factor

- Smooth image, then downsample



Procedure: 3.3 *Downsampling an image by a small factor*

Take the source image $\mathcal{S}$, and form a new image $\mathcal{N}$ from that source. The i, j'th pixel in $\mathcal{N}$ is now a weighted average of a $(2k+1) \times (2k+1)$ window of pixels in $\mathcal{S}$, centered on i, j. Organize the weights into a small array – the *mask*, which you could obtain by evaluating the Gaussian, as above – and form a new image $\mathcal{N}$ from the original image and the mask, using the rule

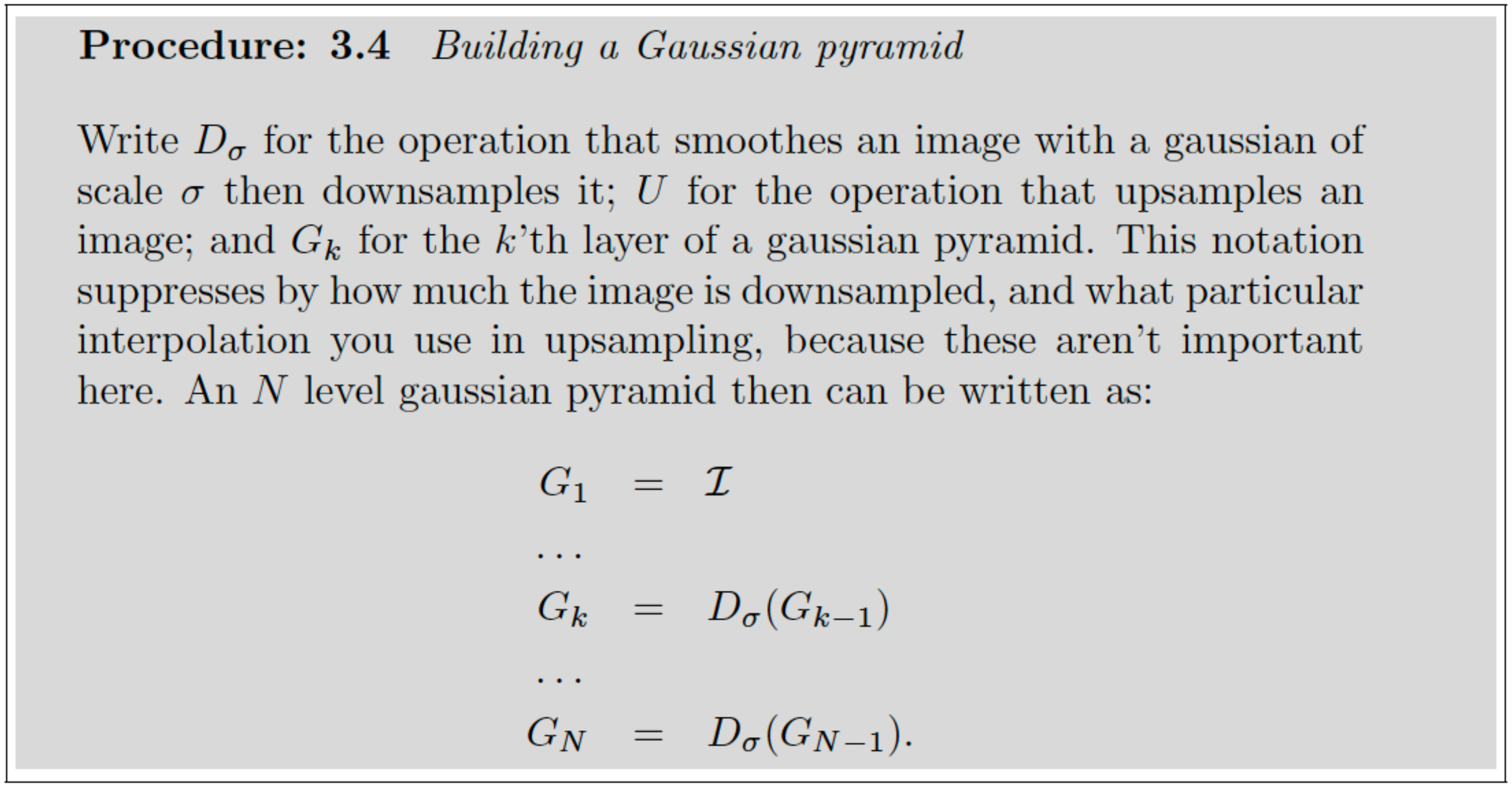
$$\mathcal{N}_{ij} = \sum_{uv} \mathcal{I}_{i-u,j-v} \mathcal{W}_{uv}$$

This expression is the root of all sorts of interesting ideas (Chapter 5). There are some problems when i or j or u or v are too big or too small. Deal with these by asserting that $\mathcal{I}$ and $\mathcal{W}$ are zero for locations outside the range. Evaluate $\mathcal{N}$ on an $M \times N$ grid. Now downsample using backward warping and interpolation.

Downsampling by a big factor

Now consider downsampling by a large factor. You *could* (but shouldn't) smooth with a gaussian with large σ, then downsample. This is not a good idea, because the support of the gaussian is infinite, meaning that working with a $2k+1 \times 2k+1$ window involves some truncation. As σ gets bigger, k will need to get bigger to keep this truncation reasonable, so the smoothing process will be expensive. The more efficient alternative is to smooth, downsample by two, then smooth the result and downsample *that* by two and so on, until the image size is only slightly larger than what you want. Then downsample that by a small factor.

Gaussian pyramid

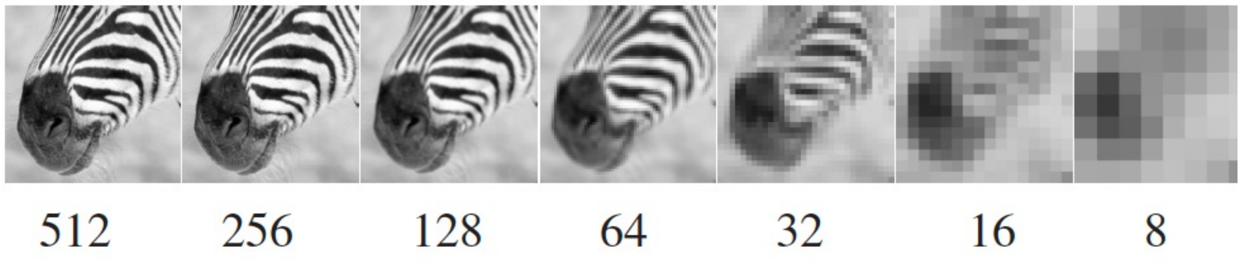


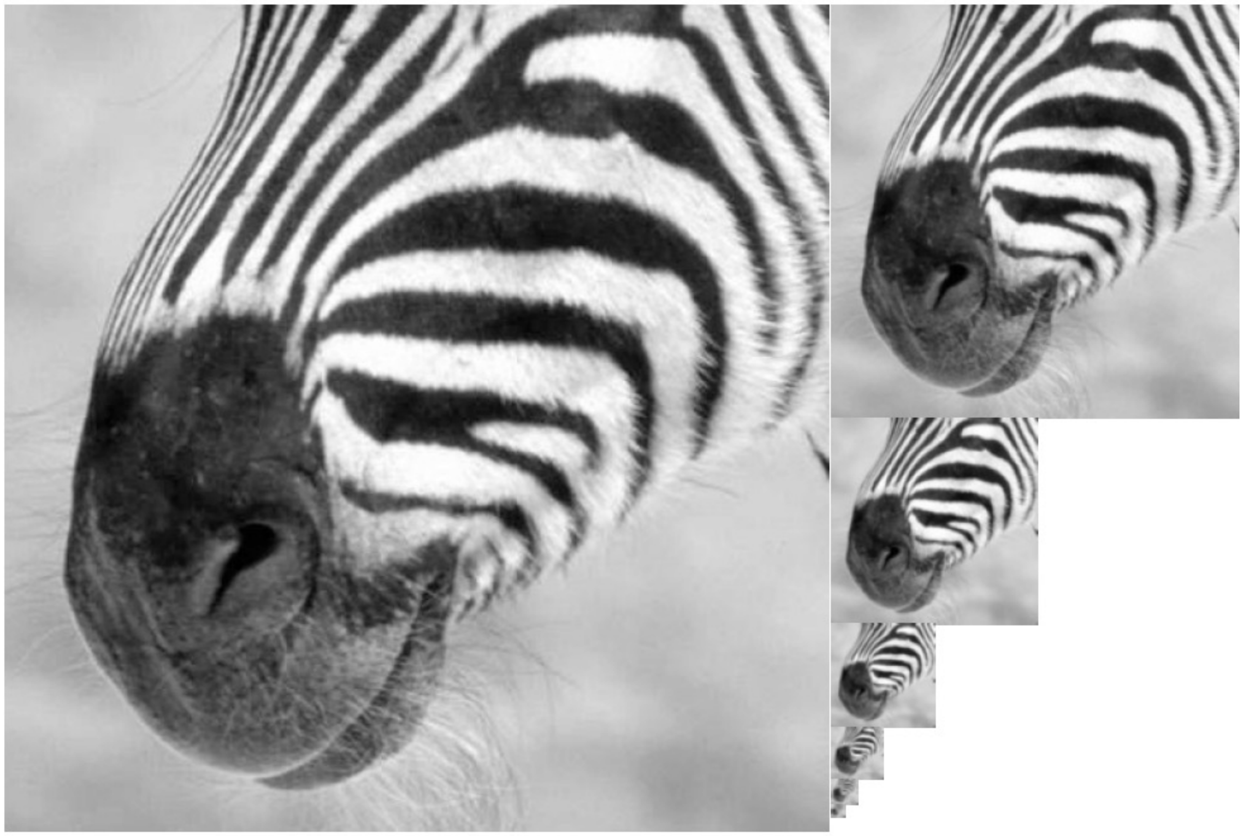
Procedure: 3.4 *Building a Gaussian pyramid*

Write D_σ for the operation that smoothes an image with a gaussian of scale σ then downsamples it; U for the operation that upsamples an image; and G_k for the k'th layer of a gaussian pyramid. This notation suppresses by how much the image is downsampled, and what particular interpolation you use in upsampling, because these aren't important here. An N level gaussian pyramid then can be written as:

$$\begin{aligned} G_1 &= \mathcal{I} \\ \ldots & \\ G_k &= D_\sigma(G_{k-1}) \\ \ldots & \\ G_N &= D_\sigma(G_{N-1}). \end{aligned}$$

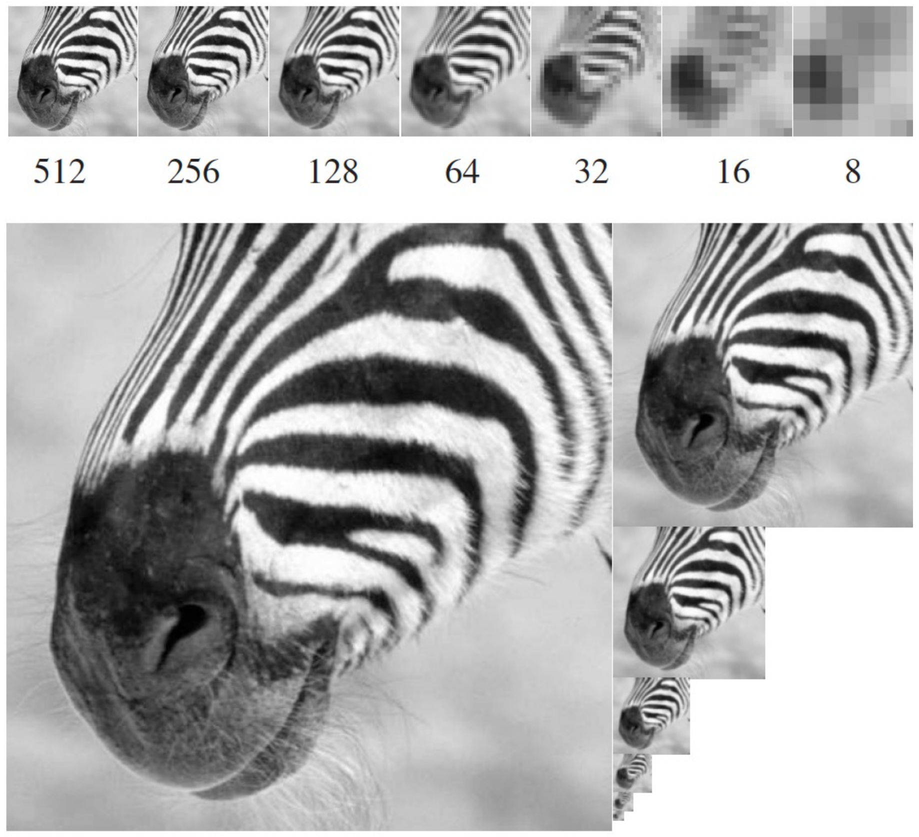
G.P.



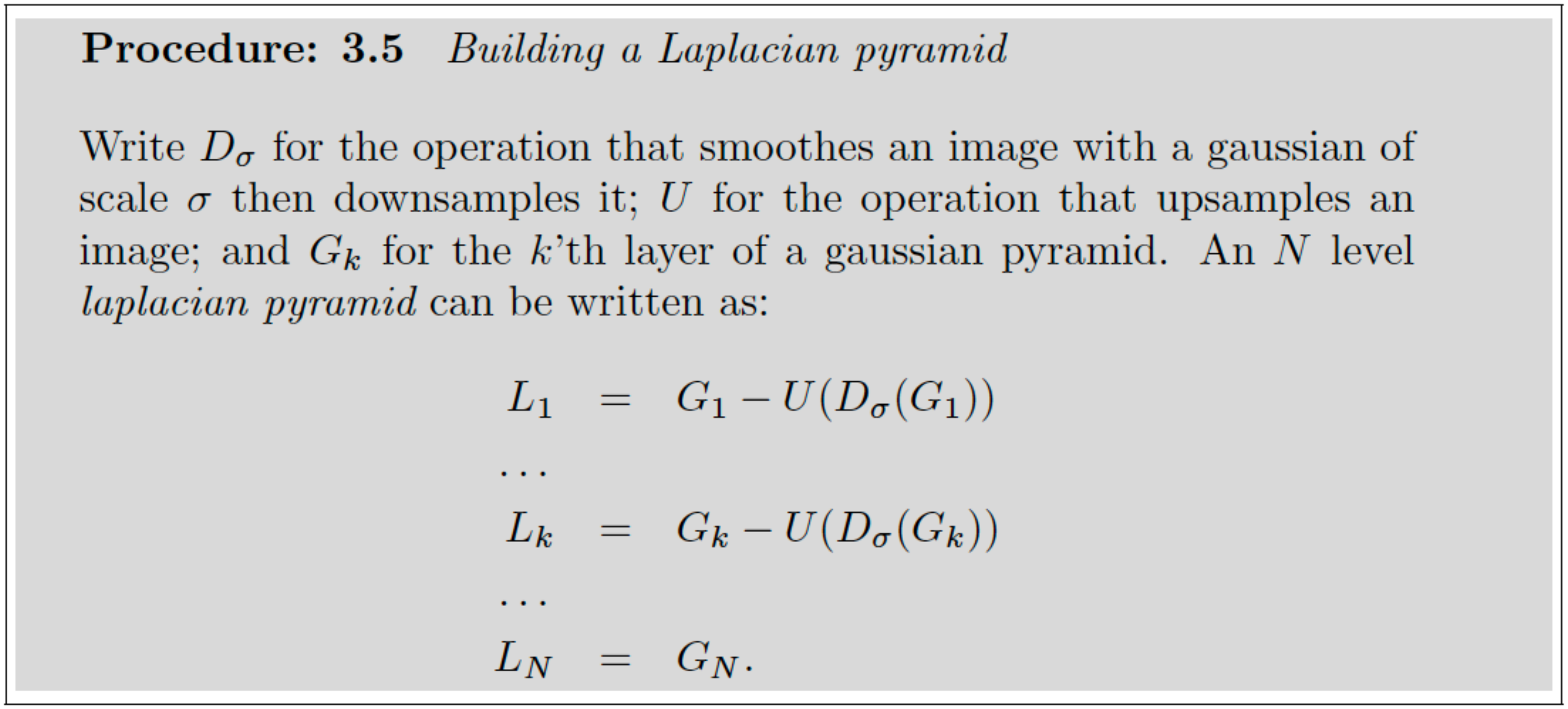


Laplacian pyramid

- Gaussian pyramids are redundant



Laplacian pyramid

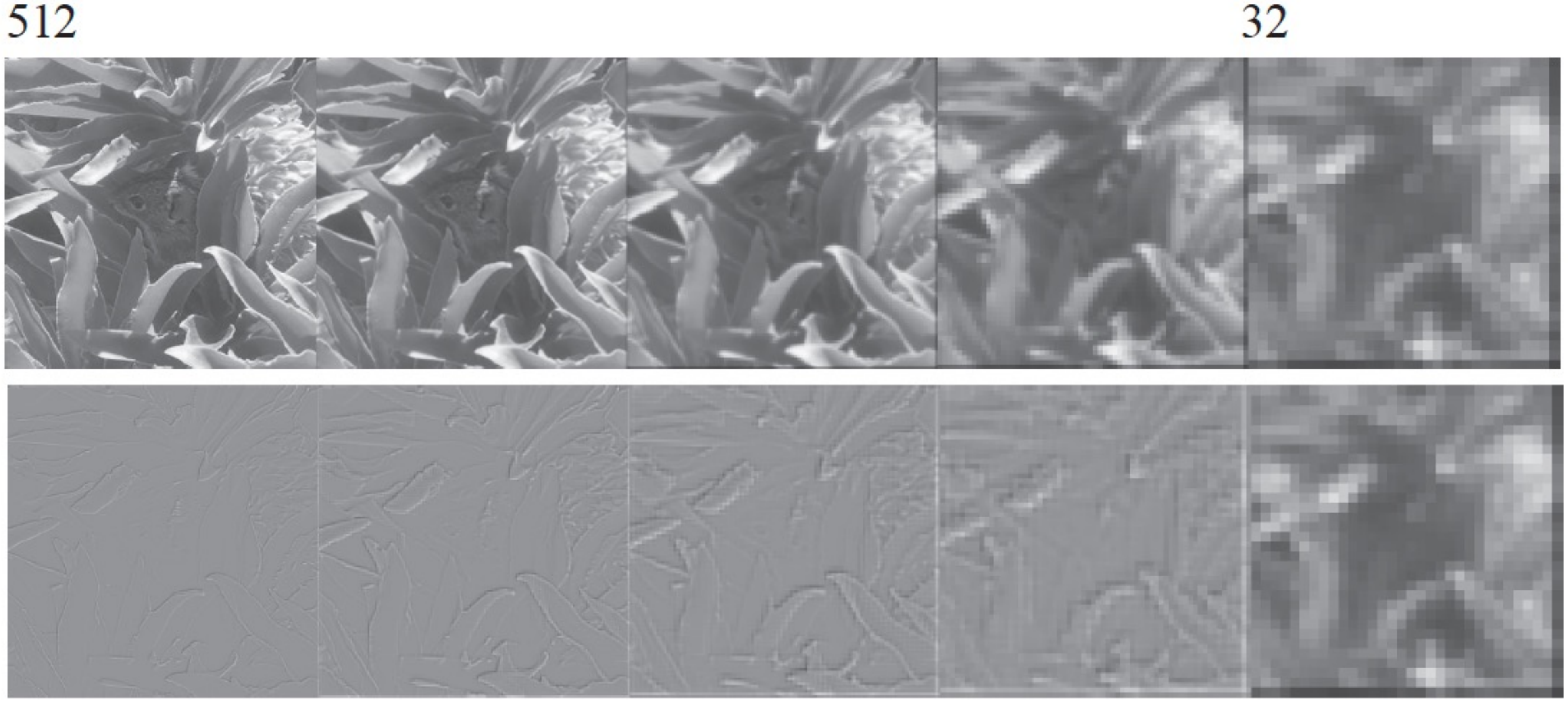


Procedure: 3.5 *Building a Laplacian pyramid*

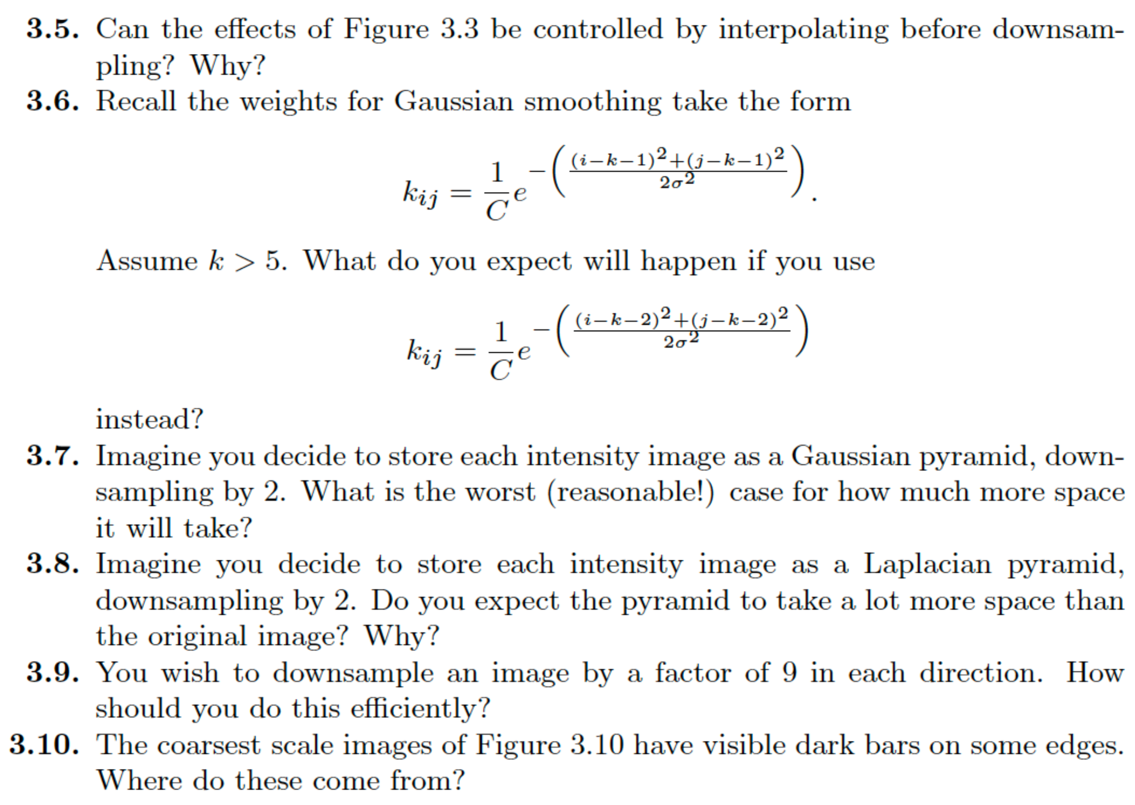
Write D_σ for the operation that smoothes an image with a gaussian of scale σ then downsamples it; U for the operation that upsamples an image; and G_k for the k'th layer of a gaussian pyramid. An N level *laplacian pyramid* can be written as:

$$\begin{aligned} L_1 &= G_1 - U(D_\sigma(G_1)) \\ \ldots & \\ L_k &= G_k - U(D_\sigma(G_k)) \\ \ldots & \\ L_N &= G_N. \end{aligned}$$

Gaussian vs Laplacian



Things to think about…



3.5. Can the effects of Figure 3.3 be controlled by interpolating before downsampling? Why?

3.6. Recall the weights for Gaussian smoothing take the form

$$k_{ij} = \frac{1}{C} e^{-\left(\frac{(i-k-1)^2+(j-k-1)^2}{2\sigma^2}\right)}.$$

Assume $k > 5$. What do you expect will happen if you use

$$k_{ij} = \frac{1}{C} e^{-\left(\frac{(i-k-2)^2+(j-k-2)^2}{2\sigma^2}\right)}$$

instead?

3.7. Imagine you decide to store each intensity image as a Gaussian pyramid, downsampling by 2. What is the worst (reasonable!) case for how much more space it will take?

3.8. Imagine you decide to store each intensity image as a Laplacian pyramid, downsampling by 2. Do you expect the pyramid to take a lot more space than the original image? Why?

3.9. You wish to downsample an image by a factor of 9 in each direction. How should you do this efficiently?

3.10. The coarsest scale images of Figure 3.10 have visible dark bars on some edges. Where do these come from?