172 3 Image processing

**Ex 3.4: Difference keying** Implement a difference keying algorithm (see Section 3.1.3) (Toyama, Krumm, Brumitt *et al.* 1999), consisting of the following steps:

1. Compute the mean and variance (or median and robust variance) at each pixel in an “empty” video sequence.
2. For each new frame, classify each pixel as foreground or background (set the back- ground pixels to RGBA=0).
3. (Optional) Compute the alpha channel and composite over a new background.
4. (Optional) Clean up the image using morphology (Section 3.3.1), label the connected components (Section 3.3.4), compute their centroids, and track them from frame to frame. Use this to build a “people counter”.

**Ex 3.5: Photo effects** Write a variety of photo enhancement or effects ﬁlters: contrast, so- larization (quantization), etc. Which ones are useful (perform sensible corrections) and which ones are more creative (create unusual images)?

**Ex 3.6: Histogram equalization** Compute the gray level (luminance) histogram for an im- age and equalize it so that the tones look better (and the image is less sensitive to exposure settings). You may want to use the following steps:

1. Convert the color image to luminance (Section 3.1.2).
2. Compute the histogram, the cumulative distribution, and the compensation transfer function (Section 3.1.4).
3. (Optional) Try to increase the “punch” in the image by ensuring that a certain fraction of pixels (say, 5%) are mapped to pure black and white.
4. (Optional) Limit the local *gain f′*(*I*) in the transfer function. One way to do this is to limit *f* (*I*) *< γI* or *f′*(*I*) *< γ* while performing the accumulation (3.9), keeping any unaccumulated values “in reserve”. (I’ll let you ﬁgure out the exact details.)
5. Compensate the luminance channel through the lookup table and re-generate the color image using color ratios (2.116).
6. (Optional) Color values that are *clipped* in the original image, i.e., have one or more saturated color channels, may appear unnatural when remapped to a non-clipped value. Extend your algorithm to handle this case in some useful way.

**Ex 3.7: Local histogram equalization** Compute the gray level (luminance) histograms for each patch, but add to vertices based on distance (a spline).

1. Build on Exercise 3.6 (luminance computation).
2. Distribute values (counts) to adjacent vertices (bilinear).
3. Convert to CDF (look-up functions).
4. (Optional) Use low-pass ﬁltering of CDFs.