HOW TO AVOID SMOOTHING A HISTOGRAM, AND WHY

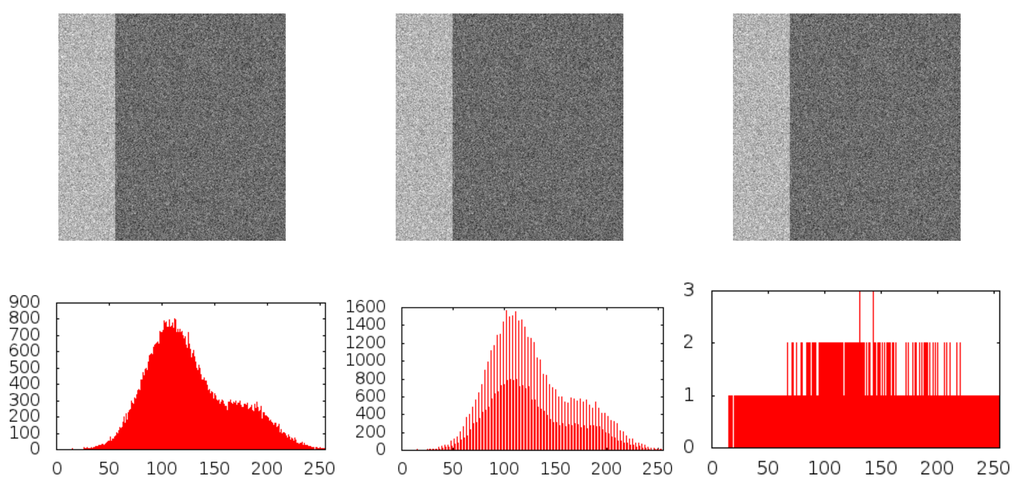
Enric Meinhardt-Llopis (ENS Cachan, France)

Summary

Smoothing a histogram is a common operation to obtain a better approximation of the underlying density. It corresponds to increasing the number of data points by adding noise to them. In the case of image histograms, we can also increase the number of data points by image interpolation.

Three images and their histograms

u $\qquad$ $\frac{3}{2}\left\lfloor\frac{2}{3}u\right\rfloor$ $\qquad$ $u+0.5n$

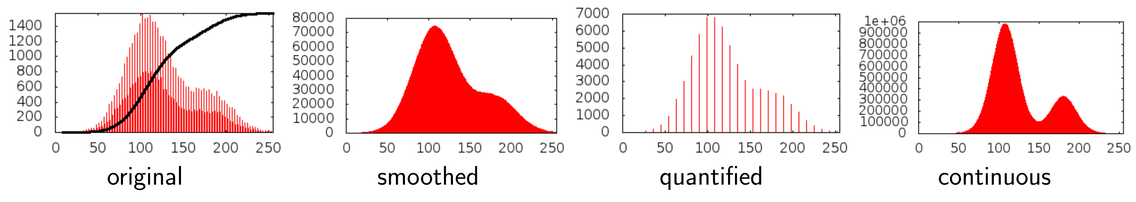


Problem: Essentially identical images have very different histograms due to quantization artifacts.

Goal: these histograms should resemble

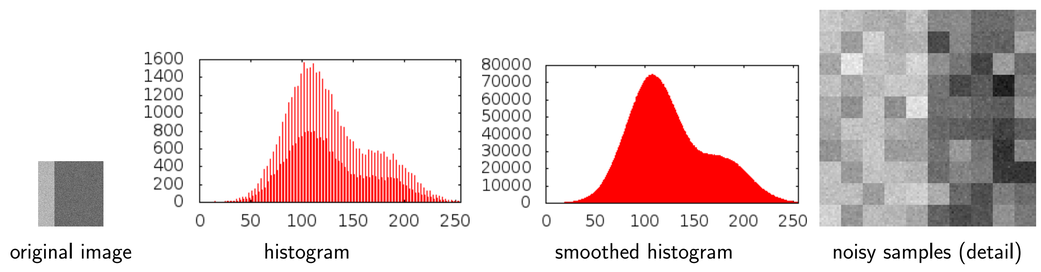
Three solutions to this problem:

- ✘ Smooth the histogram (≡ add random noise to the samples)
- ✘ Reduce the number of bins (≡ quantify the samples)
- ✔ Zoom the image (≡ add new realistic samples)



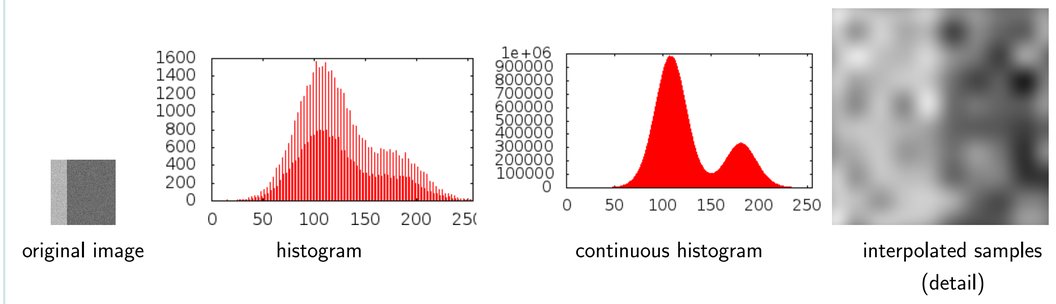
Histogram smoothing

Corresponds to adding noise to the image zoomed by repetition

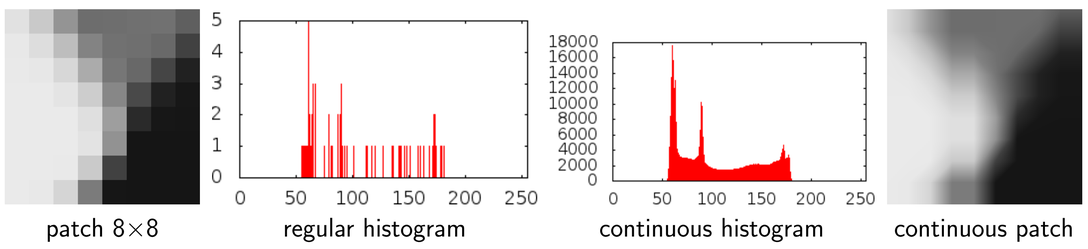


Our proposal: continuous histogram

Compute the histogram of the image zoomed $100\times$ (ideally, the continuous image)



Example: Continuous histogram of a small patch



Properties and Applications

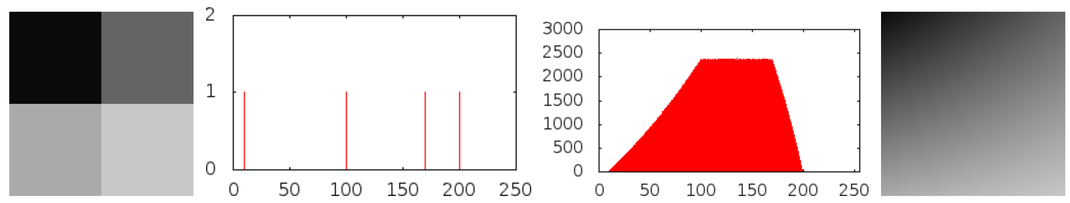
- No parameters
- The support of the histogram coincides exactly with the range of the image
- The histogram is always dense and smooth, even for very small images
- The histogram is robust to deformation and re-sampling of the image domain
- The histogram is robust to quantization of the image
- Direct extension to color images: smooth densities in RGB space

Implementations

We propose **4 different methods** to compute continuous histograms, with **different trade-offs**:

Method 1: Brute force

Zoom $100\times$ each cell of 4 pixels (by bilinear interpolation) and compute its histogram
Then sum the histograms of all cells.

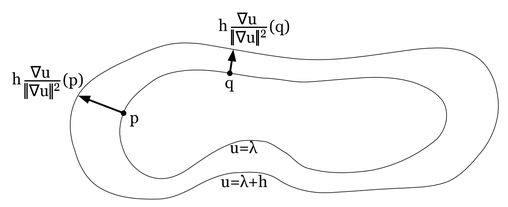


✔ Very easy to implement ✘ Very slow (running time multiplied by 10000)

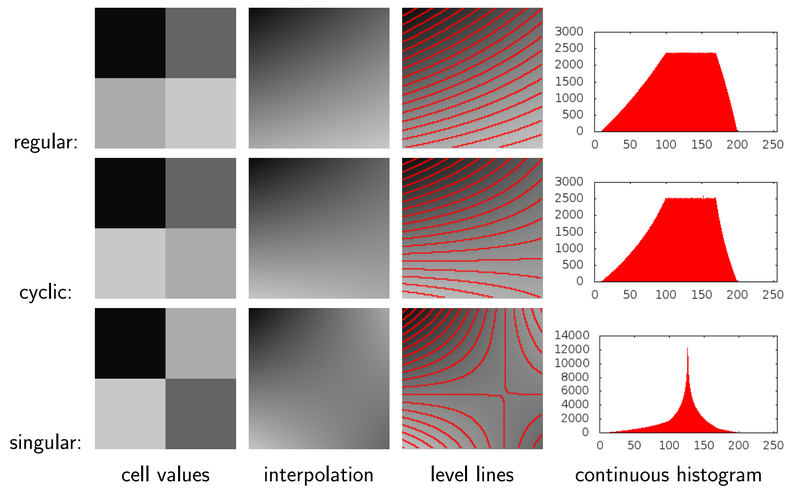
Method 2: Exact analytic expression

Use the definition of the continuous histogram:

$$h(\lambda) = \int_\Omega \delta(u(x)-\lambda)\,\mathrm{d}x = \int_{\{u=\lambda\}} \frac{1}{\|\nabla u\|}\,\mathrm{d}l$$



For bilinear interpolation, there are three topologically different cases:

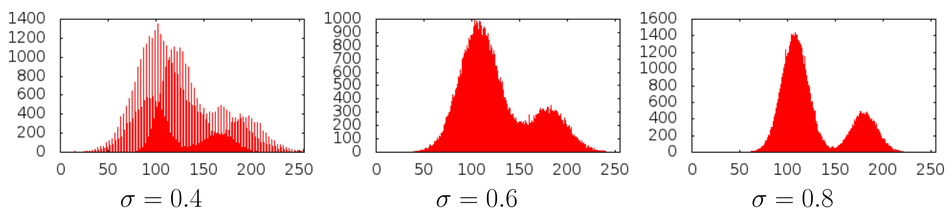


The resulting continuous histograms have the following piecewise form: $p(t) = \alpha\log(\mu t+\nu)+\beta t+\gamma$
Where $\alpha,\beta,\gamma,\mu,\nu$ are rational expressions involving the four values of the cell.

✔ Exact computation ✘ Thin vertical asymptotes at saddle points ✘ Cumbersome implementation

Method 3: Blur the image

For large enough images, the effects can be *simulated* by pre-smoothing the input image



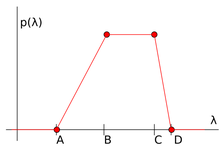
$\sigma = 0.4$ $\qquad$ $\sigma = 0.6$ $\qquad$ $\sigma = 0.8$

Magical value: $\sigma = 0.65$

✔ Trivial implementation ✘ Only reasonable for large enough images

Method 4: Approximate analytic expression

Approximate the local histograms of method 2 by piecewise affine functions.



Algorithm: `ContinuousHistogram`

Input : Discrete image $I(i,j)$ of size $W \times H$
Output : Continuous histogram $h(\lambda)$

$h := 0$
for $i = 1,\dots,W-1$ **do**
  for $j = 1,\dots,H-1$ **do**

$$A,B,C,D := \mathtt{Sort}\Big(I(i,j),\ I(i+1,j),\ I(i,j+1),\ I(i+1,j+1)\Big)$$

$$p(\lambda) := \begin{cases} 0 & \lambda \in (-\infty, A] \\ 2(\lambda-A)\ /\ (B-A)(C+D-A-B) & \lambda \in (A,B) \\ 2\ /\ (C+D-A-B) & \lambda \in [B,C] \\ 2(\lambda-D)\ /\ (C-D)(C+D-A-B) & \lambda \in (C,D) \\ 0 & \lambda \in [D,+\infty) \end{cases}$$

$h := h + p$
  end
end

✔ Easy implementation ✔ Best result for small images ✘ Somewhat slow: $O(N_{\text{pixels}}N_{\text{bins}})$

Conclusion

- Small images: use the continuous histogram
- Large images: pre-smooth ($\sigma = 6.5$) and compute the histogram