EECS C145B / BioE C165: Image Processing and Reconstruction Tomography

Lecture 7

Jonathan S. Maltz

jon@eecs.berkeley.edu http://muti.lbl.gov/145b
510-486-6744

Topics to be covered

1. 2D low-pass filters in space domain (smoothing)
2. 2D high-pass filters in space domain (sharpening)
3. Use of derivatives for edge enhancement
4. High-boost filtering
5. Non-linear space domain filters
6. The image histogram

Reading

- Gonzalez and Woods pp. 112-142, 88-108

Optional additional reading

- Bracewell pp. 267-287
- Jain pp. 235-255, 347-353

Filtering in the space domain

We will first consider filters that can be implemented using the convolution operator (linear spatially invariant filters).

$$f(x,y) \longrightarrow \boxed{h(x,y)} \longrightarrow g(x,y)$$

- These filters are completely described by their point spread function $h(x,y)$.
- Discrete space domain LSI filters are often called “masks” or “convolution masks”.

Filtering in the space domain: Five minute class exercise

We have already seen that a smoothing filter may be implemented as an "averaging mask" such as:

$$\mathbf{H} = \begin{bmatrix} 1 & 4 & 6 & 4 & 1 \\ 4 & 16 & 24 & 16 & 4 \\ 6 & 24 & 36 & 24 & 6 \\ 4 & 16 & 24 & 16 & 4 \\ 1 & 4 & 6 & 4 & 1 \end{bmatrix} /256$$

Consulting only other students, try to come up with masks to perform the following:

1. Image sharpening (high-pass filter)
2. Horizontal edge enhancement
3. Vertical edge enhancement

Low-pass spatial filters

We will consider the Gaussian low-pass spatial filter. The true Gaussian filter has infinite support, and so is not practical to implement. Its PSF is given by:

$$h(x, y) = \frac{1}{2\pi\sigma^2}\,\mathrm{e}^{-(x^2+y^2)/\sigma^2}$$

Note that this PSF is separable:

$$h(x, y) = h'(x)\,h'(y) = \frac{1}{\sqrt{2\pi\sigma^2}}\,\mathrm{e}^{-x^2/\sigma^2} \times \frac{1}{\sqrt{2\pi\sigma^2}}\mathrm{e}^{-y^2/\sigma^2}$$

This means that the Gaussian filter can be implemented by applying two 1D Gaussian filters.

Low-pass Gaussian spatial filters

In practical image processing, we need filters of finite length. The **binomial approximation** is very useful for making spatial filters based on Gaussians. Using this approximation, the function:

$$h'(x) = \frac{1}{\sqrt{2\pi\sigma^2}}\,\mathrm{e}^{-x^2/\sigma^2}$$

may be approximated by the Nth row of Pascal's triangle, where $N = 4\,\sigma^2 + 1$. The first few rows of Pascal's triangle are:

1

1 1

1 2 1

1 3 3 1

1 4 6 4 1

Low-pass Gaussian spatial filters

Row N for $N > 2$ of Pascal's triangle may be created by convolving the vector:

$$\begin{bmatrix} 1 \\ 1 \end{bmatrix}$$

with itself $N-2$ times, and then normalizing by $2^{-(N-1)}$. For example row 4, which gives the 4-point Gaussian filter, may be calculated as follows:

$$\left(\begin{bmatrix} 1 \\ 1 \end{bmatrix} * \begin{bmatrix} 1 \\ 1 \end{bmatrix} * \begin{bmatrix} 1 \\ 1 \end{bmatrix} \right) \times 2^{-(4-1)} = \begin{bmatrix} 1 \\ 3 \\ 3 \\ 1 \end{bmatrix} \times \frac{1}{8} = \begin{bmatrix} 0.125 \\ 0.375 \\ 0.375 \\ 0.1250 \end{bmatrix}$$

Low-pass Gaussian spatial filters: 2D implementation.

Let the binomial approximation PSF h[n] form the elements of the column vector $\mathbf{h}$. The 2D convolution is implemented by:

1. Performing the 2D convolution:

$$g'[m,n] = h[n] * f[m,n]$$

   This is equivalent to convolving the image with the vector $\mathbf{h}$ using a 2D convolution algorithm.

2. Performing the 2D convolution:

$$g[m,n] = h[m] * g'[m,n]$$

   This is equivalent to convolving the image with the vector $\mathbf{h}^T$ using a 2D convolution algorithm.

Low-pass Gaussian spatial filters: 2D implementation

As formulated above, the filter will smooth equally in x and y directions. However, it is possible to use filters of different length in x and y directions. The effective 2D filter can then be found by taking the outer product:

$$\mathbf{H} = \mathbf{h}_1 \mathbf{h}_2^T$$

where $\mathbf{h}_1$ is the vertical smoothing kernel and $\mathbf{h}_2$ is the horizontal smoothing kernel. For example:

$$\frac{1}{4} \times \frac{1}{8} \times \begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix} \begin{bmatrix} 1 & 3 & 3 & 1 \end{bmatrix} = \frac{1}{32} \times \begin{bmatrix} 1 & 3 & 3 & 1 \\ 2 & 6 & 6 & 2 \\ 1 & 3 & 3 & 1 \end{bmatrix}$$

Would this filter be better suited for hiding horizontally or vertically oriented wrinkles in a photograph?

Low-pass Gaussian spatial filters: Demo

We will attempt to remove the horizontal furrows from this brow using a Gaussian LPF:

Original image

Low-pass Gaussian spatial filters: Demo

We will try an 8 row × 1 column, a 1 row × 8 column and an 8 row × 8 column Gaussian LPF. Which filter was used on each of the following images?

Low-pass Gaussian spatial filters: Demo

Filtered with ________ Gaussian LPF

Low-pass Gaussian spatial filters: Demo

Filtered with ________ Gaussian LPF

Low-pass Gaussian spatial filters: Demo

Filtered with ________ Gaussian LPF

High-pass Gaussian spatial filters

Recall that a high-pass filter $H_h(u, v)$ could be obtained from a low-pass filter prototype $H_l(u, v)$ in the frequency domain via the equation:

$$H_h(u, v) = 1 - H_l(u, v)$$

Taking the inverse FT yields:

$$h_h(x, y) = \delta(x, y) - h_l(x, y)$$

In practice, we obtain a high-pass Gaussian filter by taking the negative of the kernel and adding 1 to its central element. This is easier when the kernel is of odd length.

High-pass Gaussian spatial filters: Demo

An 8×1 Gaussian HPF was applied to the original image:

Filtered with high–pass 8x1 Gaussian HPF

Note the exaggerated edges. This filter attenuates low frequencies too effectively for use as a "cosmetic" image sharpening filter.

Detail-enhancing Gaussian spatial filters: Demo

If we add the original image to the high-pass filtered image, we restore the low frequency content and add the amplified detail. Finally we have an effective detail-enhancing filter:

$$g_{\text{sharp}}[m,n] = f[m,n] * h_h[m,n] + f[m,n]$$

When applied to the image, we get:

Detail-enhancing Gaussian spatial filters: Demo

Filtered with high–pass 8x1 Gaussian HPF + original

Note that both detail and noise are amplified.

Detail enhancing filters based on the Laplacian

The Laplacian is the simplest isotropic derivative operator:

$$\nabla^2 f(x,y) = \frac{\partial^2 f(x,y)}{\partial x^2} + \frac{\partial^2 f(x,y)}{\partial y^2}$$

When applying the Laplacian to an image we expect the value of the Laplacian to be sensitive to the rate of change of luminance in an image. Consequently the Laplacian

1. Will be zero inside areas with constant luminance.
2. Will be zero inside areas with constantly changing luminance (grey level ramps).
3. Will strongly emphasize features of small size, for example, isolated bright points and thin lines i.e., fine image detail

Since the operators based on the first derivative do not possess all of these desirable properties, we will consider only second order sharpening filters.

Detail enhancing filters based on the Laplacian

In the discrete space domain, we must approximate the derivative operator using differences. We have in the x direction:

$$L_x[m,n] = f[m+1,n] + f[m-1,n] - 2f[m,n]$$

and in the y direction:

$$L_y[m,n] = f[m,n+1] + f[m,n-1] - 2f[m,n]$$

Adding these, we get a discretization of the Laplacian:

$$L_{xy}[m,n] = f[m+1,n] + f[m-1,n] + f[m,n+1] + f[m,n-1] - 4f[m,n]$$

Detail enhancing filters based on the Laplacian

This is not a very good approximation because it is only isotropic along the x and y axes. In square pixel images, detail oriented at 45 degrees should also be emphasized. An improved mask is obtained by rotating the mask as presently defined:

$$\begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

by 45 degrees, and then summing the masks. A frequently used mask is:

$$\begin{bmatrix} 1 & -2 & 1 \\ -2 & 4 & -2 \\ 1 & -2 & 1 \end{bmatrix}$$

The Laplacian is generally a noisy operator. Usually a multiple of the original image is added so that low frequency content is present.

Detail enhancing filters based on the Laplacian: Demo

Original image

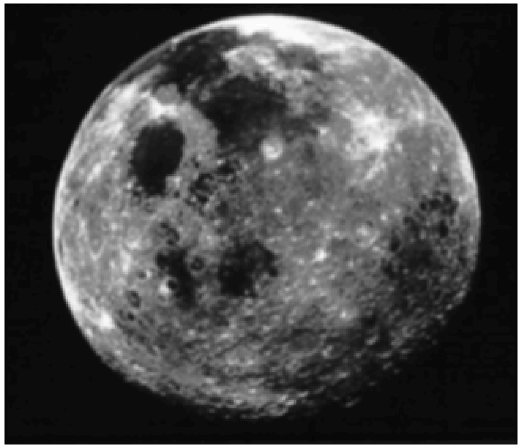


Image courtesy NASA.

Detail enhancing filters based on the Laplacian: Demo

Filtered with Laplacian mask + original image

Detail enhancing filters: High-boost filtering

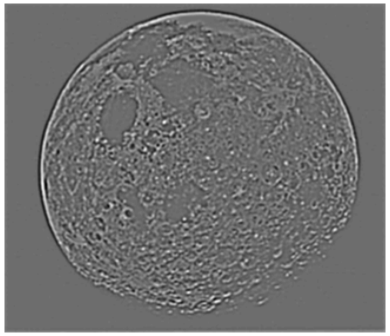
Detail can be enhanced by subtracting a smoothed version of an image from a multiple of the image:

$$f_{\text{sharp}}(x, y) = Af(x, y) - f_{\text{smooth}}(x, y)$$

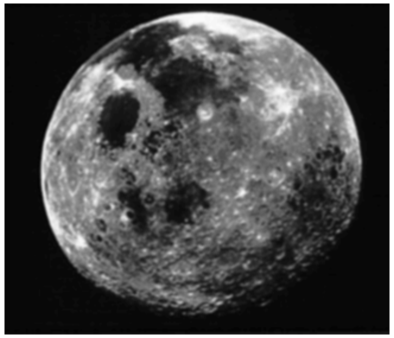
When $A = 1$ this is known as **unsharp masking**.

High-boost filter demo

1 x image minus 3x3 Gaussian LPF'd image



1.25 x image minus 3x3 Gaussian LPF'd image

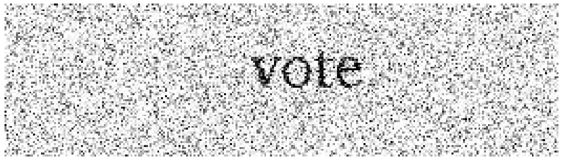


Practical image processing: 5 minute class problem

Consider the following image. It is employed to prevent computer scripts from logging in to a website. A user must read the text in the corrupted image and enter it at a prompt to verify that a real person is using the site. By the end of this course, you should be able to write a program that often succeeds at guessing the word.

What would your first steps be in preparing this image so that the text can be read by a computer?

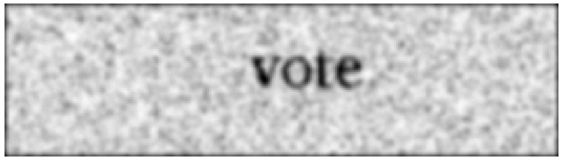
Original image



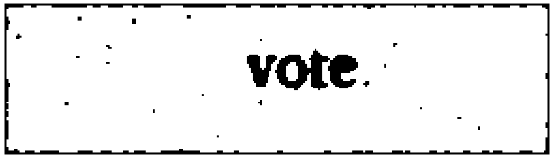
Notes:

One approach to despeckling

Filtered with 3x3 Gaussian LPF



Smoothed image thresholded at 0.75 of max. intensity



Non-linear filtering: Thresholding

When we threshold an image, we map an interval of gray level values to one, and all the rest to zero:

$$
\begin{aligned}
f_t(x, y) &= 1 \quad \text{for } I_l \leq f(x, y) \leq I_h \\
&= 0 \quad \text{otherwise}
\end{aligned}
$$

where I_h is the upper level threshold and I_l is the lower threshold. In the previous example, $I_l = 0$ and $I_h = 0.75 \max(f(x, y))$

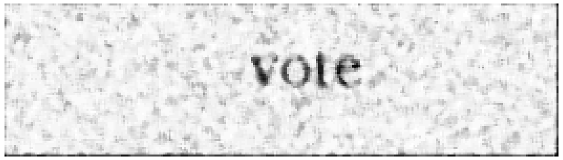
Non-linear filtering: Order statistics filters

- Order statistics filters rank the intensities of pixels within a window.
- The user specifies the rank of the pixel whose value will be chosen by the filter.
- This value is then substituted for the pixel at the center of the window.
- The window moves over every point in the image.
- The **median filter** substitutes the value of the pixel that is in the middle of the list of pixels in the window that has been sorted according to intensity.
- The **min filter** takes the value of the lowest intensity pixel.
- The **max filter** takes the value of the highest intensity pixel.
- In general, the value of the pixel at any percentile rank may be selected.

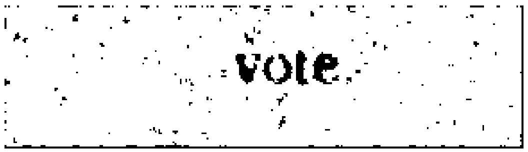
Order-statistics filters are good for removing **impulse noise** also called **salt and pepper** noise. It consists of randomly located dots of the same or similar intensities.

Despeckling with median filter

3x3 median filtered image



Median filtered image thresholded at 0.75 of max. intensity



What are the pros and cons of using the median filter?

Despeckling with median filter: Demo with true salt and pepper noise

Image contaminated by salt and pepper noise

Despeckling with median filter: Demo with true salt and pepper noise

Image smoothed with 3x3 Gaussian LPF

Despeckling with median filter: Demo with true salt and pepper noise

Filtered with 3x3 median filter

Despeckling with median filter

- The Gaussian LPF does not improve the ability to resolve obscured detail in the image. It merely smears out the dots and blurs true detail.
- The median filter is far more effective at removing the dots than a linear filter. In fact, the previously illegible text on the name tag becomes almost legible. The median filter tends to "eat away" a little at fine features such as text.
- Note that while linear filters can be applied in any order, non-linear filters cannot. What would happen if we applied the median filter to the Gaussian-smoothed image?

- Why do you think the median filter did not work as well on the "vote" text as on the photograph?

- Why did the LPF not do too badly on the "vote" text?

The Image Histogram

- The image histogram tells us how many pixels occur at each gray level value within an image.
- So that we can find the histogram of an image with continuous intensity values, we normally map ranges of intensities into **bins**.
- Let $I_{\max}$ represent the maximum intensity value in an image and $I_{\min}$ be slighty smaller than the lowest value. We calculate the value of the lth bin of a histogram with L bins as:

$$
\begin{aligned}
h[l] &= |\mathcal{I}_l| \qquad l = 0, 1, \ldots, (L-1) \\
\mathcal{I}_l &\triangleq \{m, n \mid \\
& \quad l \times (I_{\max} - I_{\min})/L < f[m,n] \le (l+1) \times (I_{\max} - I_{\min})/L\}
\end{aligned}
$$

Where $|\mathcal{I}_l|$ is the number of elements in, or **cardinality** of, the set $\mathcal{I}_l$.

The Image Histogram

- The histogram does not depend on where pixels are in an image.
- Example: An image has pixel values between 0 and 1. We wish to histogram the image into 5 bins. Then the gray level limits of each bin are:

$$
\begin{aligned}
\mathcal{I}_0 &: [0.0\ 0.2] \\
\mathcal{I}_1 &: (0.2\ 0.4] \\
\mathcal{I}_2 &: (0.4\ 0.6] \\
\mathcal{I}_3 &: (0.6\ 0.8] \\
\mathcal{I}_4 &: (0.8\ 1.0]
\end{aligned}
$$

The Image Histogram: Example

- Say an image contains the pixel values:

$$\mathcal{I} = \{0.1,\ 0.93,\ 0.3,\ 0.1,\ 1.0,\ 0.2,\ 0.8,\ 0.72,\ 0.1,\ 0.4,\ 0.35,\ 0.412\}$$

Then the bins contain the elements:

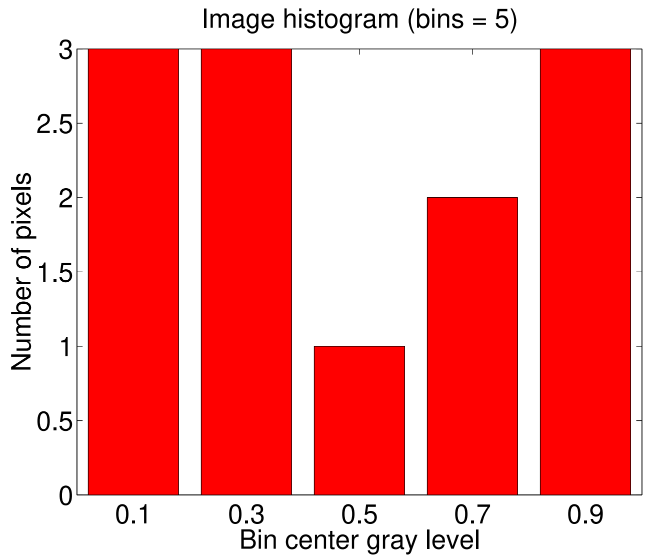
$$
\begin{aligned}
\mathcal{I}_0 &: \{0.1,\ 0.1,\ 0.1\ 0.2\} \\
\mathcal{I}_1 &: \{0.3,\ \underline{\hspace{2em}},\ 0.4] \\
\mathcal{I}_2 &: \{0.412\} \\
\mathcal{I}_3 &: \{\underline{\hspace{2em}},\ 0.8\} \\
\mathcal{I}_4 &: \{0.93,\ 1.0\}
\end{aligned}
$$

and the set cardinalities (histogram bin contents) are:

$$
\begin{aligned}
h[0] = |\mathcal{I}_0| &= 4 \\
h[1] = |\mathcal{I}_1| &= \underline{\hspace{2em}} \\
h[2] = |\mathcal{I}_2| &= 1 \\
h[3] = |\mathcal{I}_3| &= \underline{\hspace{2em}} \\
h[4] = |\mathcal{I}_4| &= 3
\end{aligned}
$$

The Image Histogram: Example

The resulting histogram is usually plotted as a bar graph:



Histogram equalization

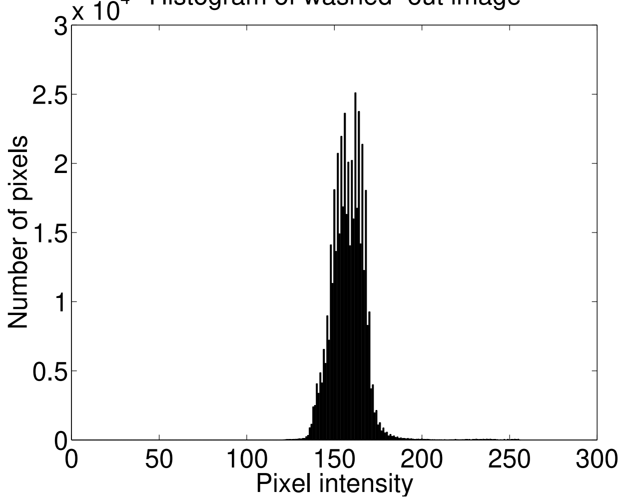
- Most often, images do not make use of the full grayscale. This can make visualization difficult.
- One way of improving contrast is to equalize the histogram. This entails **modifying the pixel values** so that all bins contain roughly the same number of pixels.

Histogram equalization: Example

Original washed−out image

Histogram equalization: Example

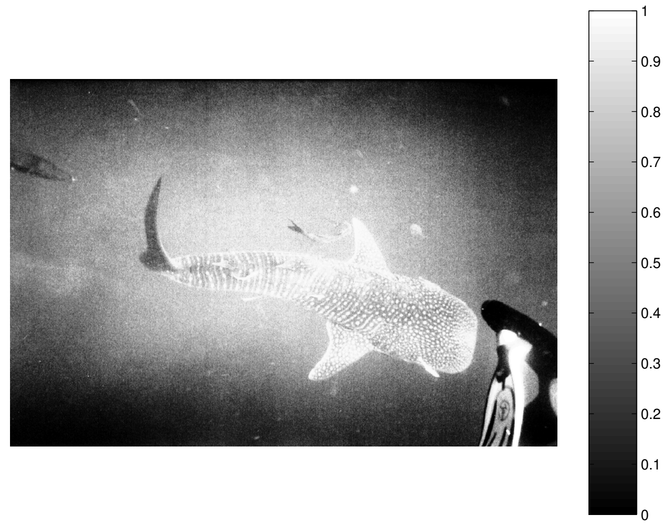
Histogram of washed−out image



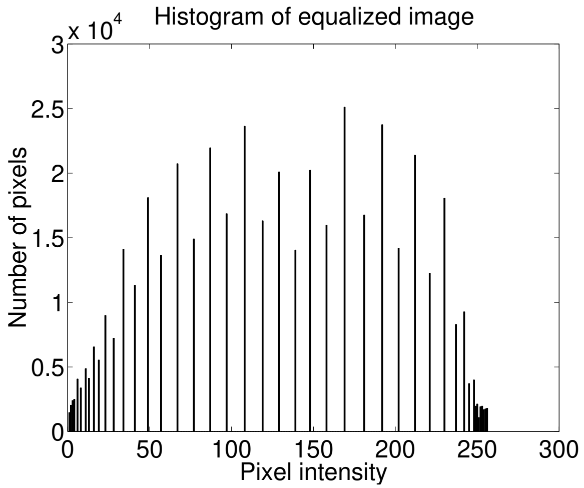
We see that image is only "using" a minority of the gray levels.

Histogram equalization: Example

Histogram−equalized image



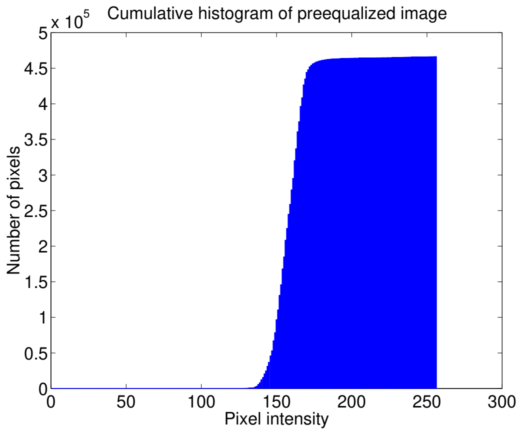
Histogram equalization: Example



The image pixel values are now spread far more evenly among the bins. Ideally, the histogram should be flat.

Histogram equalization: Theory

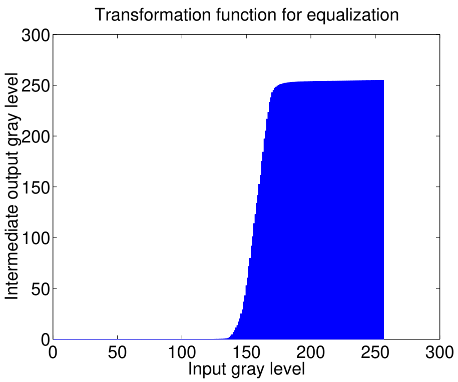
- First, we look at the cumulative histogram of the original image:



- We can imagine that if the cumulative histogram increased linearly, the pixels would be spread equally between the bins.

Histogram equalization: Theory

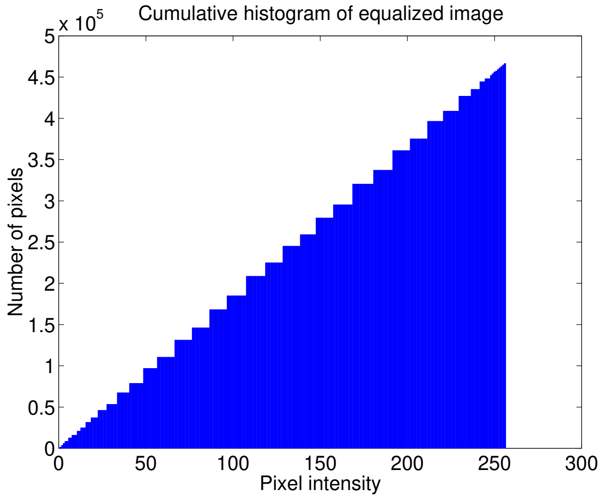
- The way to execute this transformation is to use the normalized cumulative histogram as a transformation function:



- We see that gray level intervals with the most pixels (where the count increases the fastest) will be mapped over many output gray levels.

Histogram equalization: Theory

- A look at the cumulative histogram of the equalized image shows the effect of the transformation on the histogram:



Histogram equalization: Theory

- The equation of the line defining the hypotenuse of the triangle is:

$$c(I) = \frac{|\mathcal{I}|}{(I_{\max} - I_{\min})} I$$

where I is the pixel intensity (histogram abscissa)

- Note that the derivative of $c(I)$ with respect to I is:

$$\frac{|\mathcal{I}|}{(I_{\max} - I_{\min})} = 560 \times 833/256 \approx 1822$$

and that this is the height of the (ideal) equalized histogram.

Histogram specification: Theory

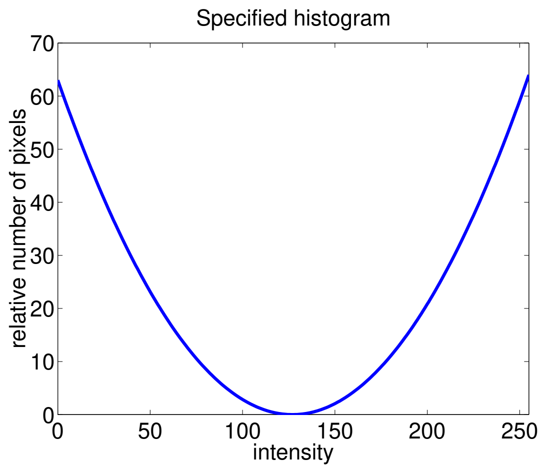
We can generalize this to allow us to specify a new histogram for an image.

1. Choose a histogram function $h_s(I)$.
2. Integrate this function to get $c'_s(I)$. The constant of integration should be chosen such that $c'_s(I_{\min}) = 0$.
3. Scale $c'_s(I)$ to create $c_s(I)$ that has a final value equal to the maximum intensity value $I_{\max}$.
4. Calculate the cumulative histogram of the image $c_i(I)$.
5. For a pixel of intensity I, look up $c_i(I)$.
6. Look up $c_i(I)$ on the ordinate axis of $c_s(I)$, and find the corresponding output gray level on the abscissa axis of $c_s(I)$.

Histogram specification: Example

- For an image with intensities from 0 to 255, we specify the histogram:

$$h_s(I) = \frac{1}{256}(I - 127)^2;$$



Histogram specification: Example

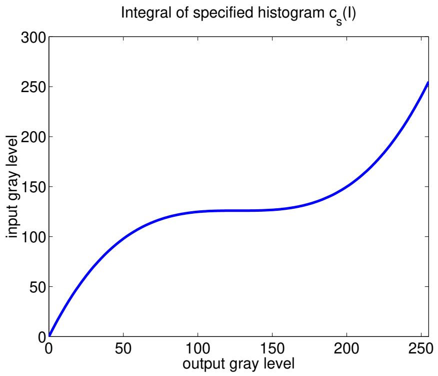
- The integral is

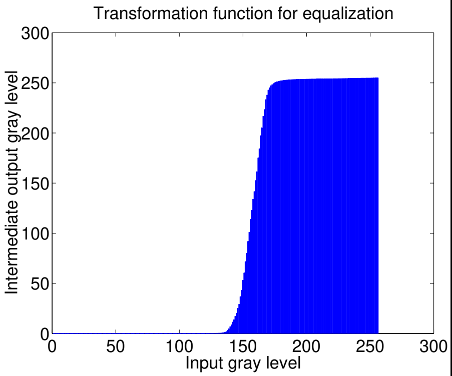
$$c'_s(I) = \frac{1}{3 \times 256}(I - 127)^3 + 2667,$$

which has a final value of ≈ 5398. Since we have $I_{\max} = 255$:

$$c_s(I) = \left[\frac{1}{3 \times 256}(I - 127)^3 + 2667\right] \times \frac{255}{5398}$$

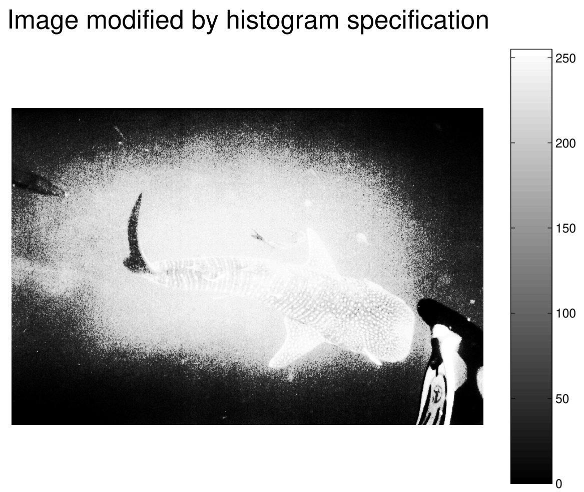
Histogram specification: Example





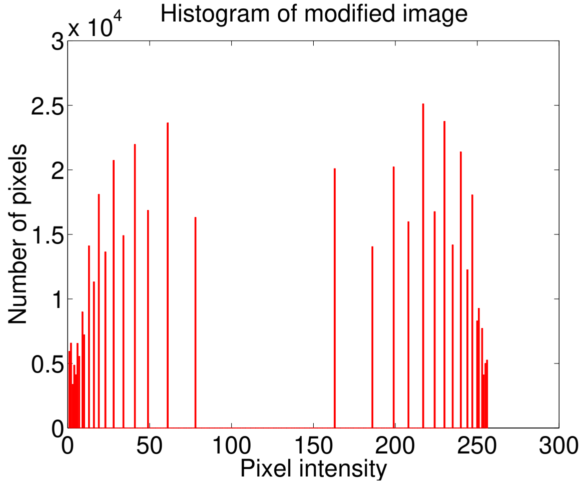
Histogram specification: Example

The modified image looks like this:



Histogram specification: Example

Finally, we check that the histogram of the modified image looks something like what we asked for:



Histogram specification: Rigorous theory

Note: The following material is **not examinable**. It is included for completeness.

- When the histogram is normalized to sum to unity, it gives an approximation to the continuous probability density function (pdf) of the image gray level values $f(i)$. Here i is a gray level value.
- The cumulative distribution function (cdf) is defined as:

$$F_I(i) \triangleq \mathbf{P}(I \leq i) = \int_{-\infty}^{i} f(q)dq$$

  where I is the gray level random variable. The cdf gives the probability that a pixel gray level will fall at a value less that the argument i. As a consequence, $F_I(i)$ has values from 0 to 1 and is strictly monotonically increasing.
- We will now show that the random variable:

$$Y = F_I(I)$$

  is uniformly distrubuted in the interval $(0, 1)$.

Histogram specification: Rigorous theory

- Let $g(i)$ be a function that satisfies $F_I(g(i)) = i$. Because $F_I(i)$ is strictly monotonically increasing and continuous, it uniquely defines $g(i)$ for every i in $(0, 1)$.
- The function $g(\cdot)$ is called the **inverse function** of $F_I(\cdot)$ and can be denoted as:

$$g(i) = F_I^{-1}(i)$$

- For an inverse function, the relation:

$$g(F_I(i)) = F_I(g(i)) = i$$

  holds.
- *Proof that y is uniformly distributed:* By definition of the cdf:

$$\begin{aligned} \mathbf{P}(Y \leq y) &= \mathbf{P}(F_I(I) \leq y) = \mathbf{P}(g(f_I(I) \leq g(y)) \\ &= \mathbf{P}(I \leq g(y)) = F(g(y)) = y. \end{aligned}$$

Histogram specification: Rigorous theory

- The probability that random variable Y falls below a value y is thus that value y. The cdf is a 45 degree ramp, and the pdf is therefore uniform on $(0, 1)$.
- As a consequence, any continuous random variable (rv) with the well-behaved cdf can be transformed to a uniform rv by its cdf $F_I(i)$.
- This uniform rv y can then be transformed to assume the original distribution of $f(i)$ using the inverse function $g(i) = F_I^{-1}(i)$.
- Let $s(z)$ be another distribution with a well-behaved cdf $S_Z(z)$. Then its inverse function $S_Z^{-1}(z)$ will transform any uniform rv to have a pdf of $s(z)$
- In **histogram equalization** we perform only one mapping, from an rv with pdf $f(i)$ to a uniform rv y.
- In **histogram specification** we perform two mappings, from an rv with pdf $f(i)$ to a uniform rv y, and then from this uniform rv to a specified distribution $s(z)$ via the inverse function for z.