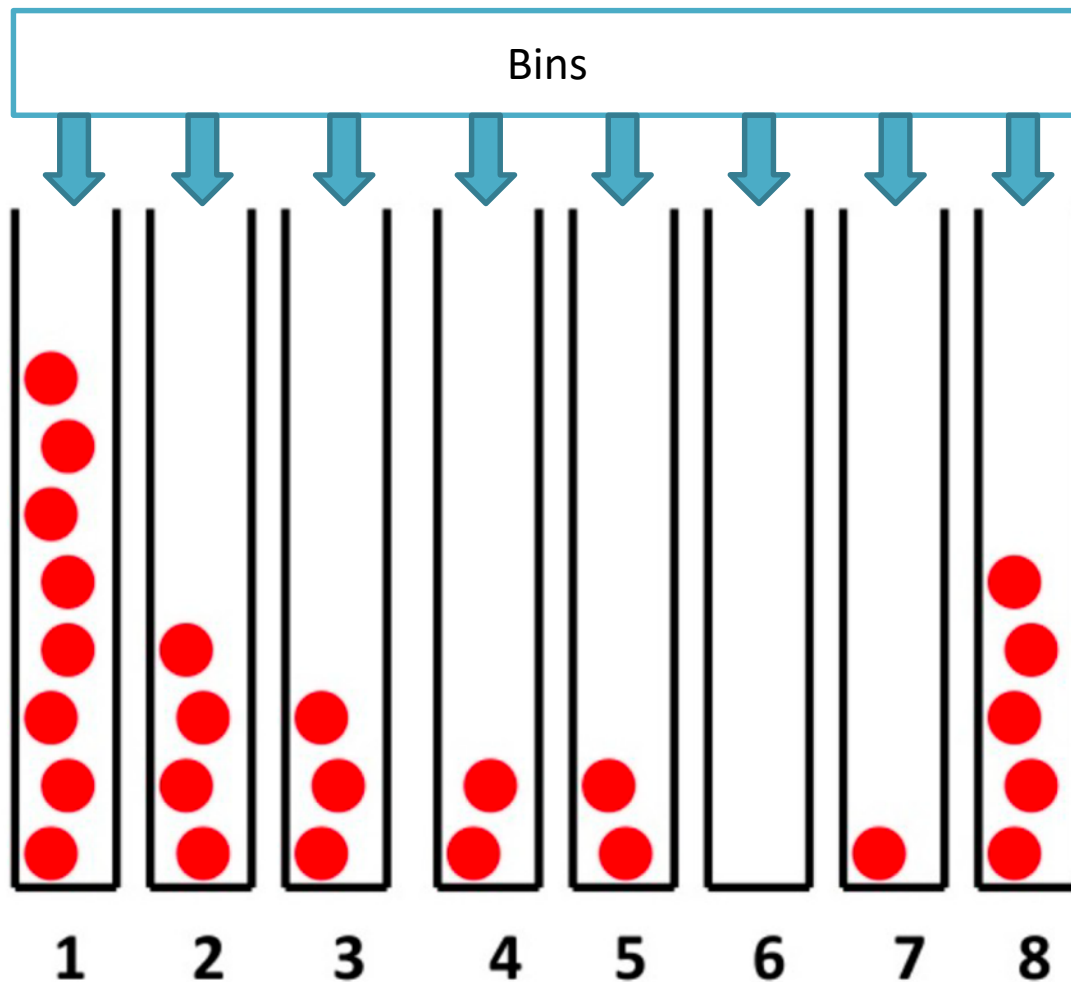# Image histogram and equalization

Stefano Ghidoni

- Image histograms

- Working on the histogram of an image
  - Histogram equalization

- What is a histogram?

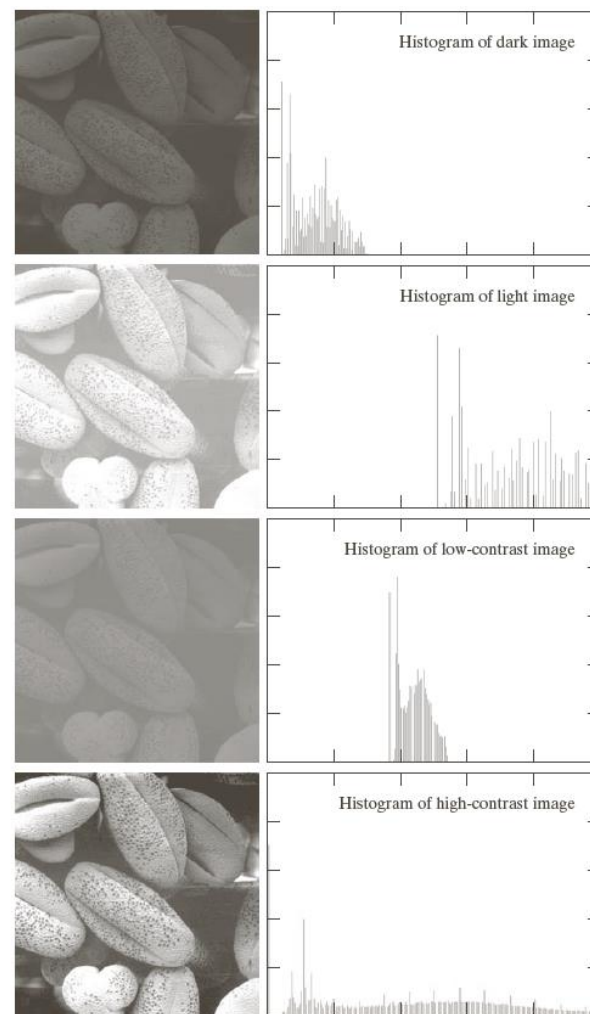- How can a histogram be evaluated from an image?

- Histograms of the grayscale values

$$p(r_k) = \frac{\boxed{h(r_k)}}{MN} = \frac{n_k}{MN}$$

# of pixels whose intensity is r_k

- Can be treated as a probabilistic density function (PDF)

Histogram of dark image

Histogram of light image

Histogram of low-contrast image
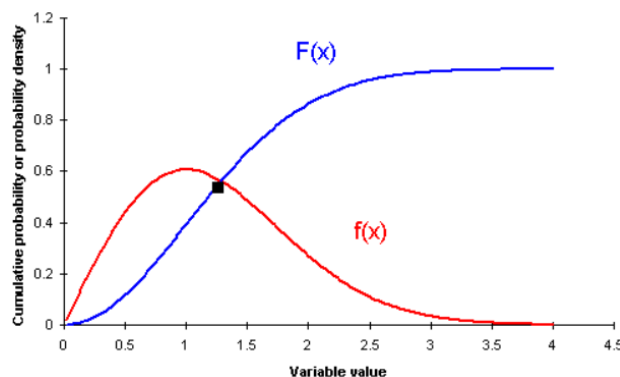
Histogram of high-contrast image

- Cumulative Distribution Function (CDF)

$$F_X(x) = P(X \leq x)$$

- Probability Density Function (PDF)

$$f_X(x) = \frac{d}{dx} F_X(x)$$

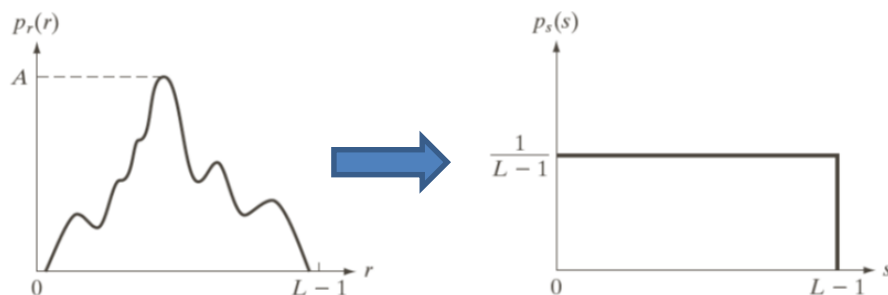$$F_X(x) = \int_{-\infty}^{x} f_X(t) dt$$

- Histograms are widely used for:
  - Evaluating image statistics
  - Compression
  - Segmentation
  - Image enhancement
- Is it meaningful to "modify the histogram"?
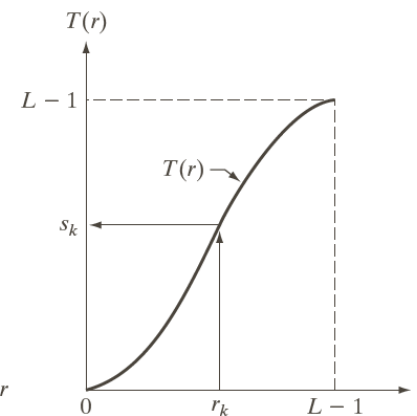 - Which operations could we apply?

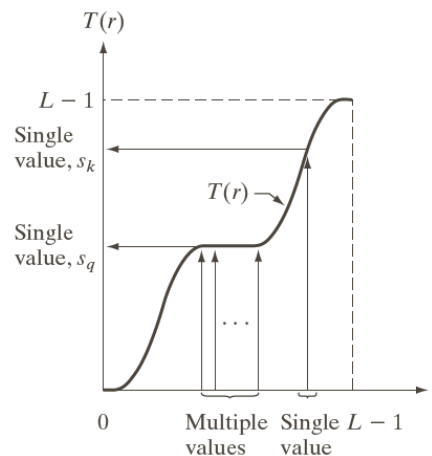- No spoiler ☺

- Histogram equalization is a process that flattens the histogram

- ## Histogram equalization

- ## Based on a equalization function
  - How to find this function?
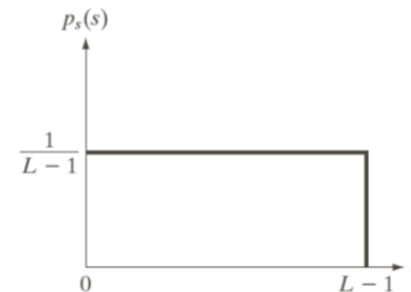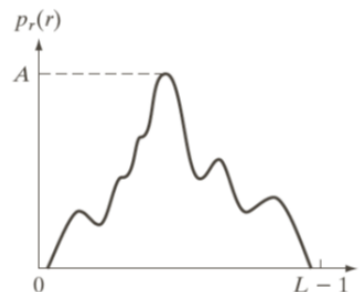
- The function $T(r)$ capable
  of equalizing the histogram
  is the CDF

$$s = T(r) = \boxed{(L-1)\int_0^r p_r(w)\,dw}$$

$$s_k = T(r_k) = (L-1)\sum_{j=0}^{k} p_r(r_j)$$
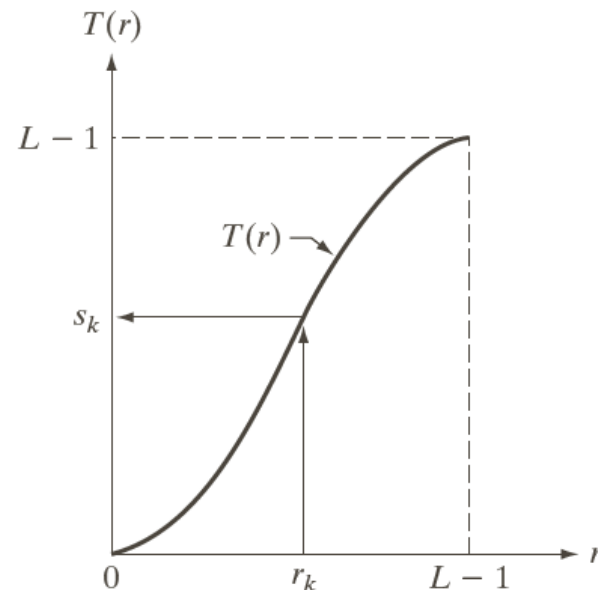
Cumulative
Distribution
Function (CDF)
of the RV r

- $T(r)$ is monotonically non-decreasing
  - The inverse function is available
- The function is bounded
$$0 \leq T(r) \leq L - 1$$
$$0 \leq r \leq L - 1$$
- $T(r)$ continuous and differentiable
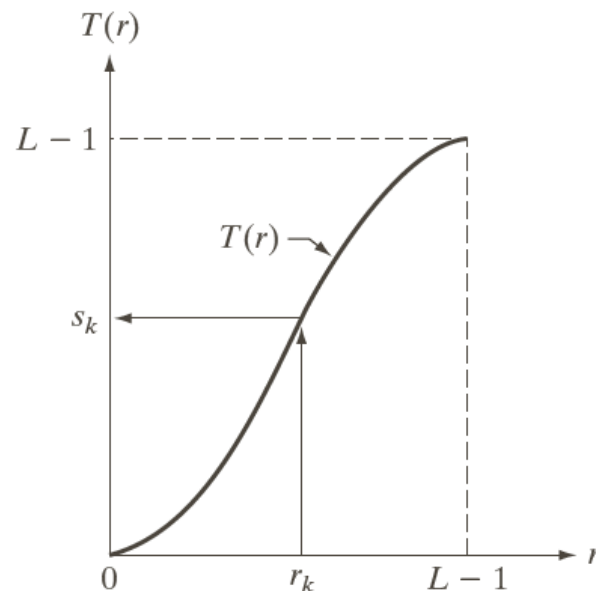
- Now consider an example
  - An image with 8 gray levels
  - Gray level distribution is given by

r_k	n_k	$p_r(r_k) = n_k/MN$
$r_0 = 0$	790	0.19
$r_1 = 1$	1023	0.25
$r_2 = 2$	850	0.21
$r_3 = 3$	656	0.16
$r_4 = 4$	329	0.08
$r_5 = 5$	245	0.06
$r_6 = 6$	122	0.03
$r_7 = 7$	81	0.02

- Now consider an example
  - The CDF is given by:

$$s_i = 7\sum_{j=0}^{i} p_r(r_j)$$

- How can we apply this formula to equalize the image?

- Using the info about pixels and the CDF formula we can fill this table

r_k	n_k	$p_r(r_k) = n_k/MN$
$r_0 = 0$	790	0.19
$r_1 = 1$	1023	0.25
$r_2 = 2$	850	0.21
$r_3 = 3$	656	0.16
$r_4 = 4$	329	0.08
$r_5 = 5$	245	0.06
$r_6 = 6$	122	0.03
$r_7 = 7$	81	0.02

$$s_i = 7 \sum_{j=0}^{i} p_r(r_j)$$

r		s_i	round
0	S_0	1.33	1
1	S_1	3.08	3
2	S_2	4.55	5
3	S_3	5.67	6
4	S_4	6.23	6
5	S_5	6.65	7
6	S_6	6.86	7
7	S_7	7.00	7

Input values

Output values

Equalization: a 3-bit example

r_k	n_k	$p_r(r_k) = n_k/MN$
$r_0 = 0$	790	0.19
$r_1 = 1$	1023	0.25
$r_2 = 2$	850	0.21
$r_3 = 3$	656	0.16
$r_4 = 4$	329	0.08
$r_5 = 5$	245	0.06
$r_6 = 6$	122	0.03
$r_7 = 7$	81	0.02

$$s_i = 7\sum_{j=0}^{i} p_r(r_j)$$

r		s_i	round
0	S_0	1.33	1
1	S_1	3.08	3
2	S_2	4.55	5
3	S_3	5.67	6
4	S_4	6.23	6
5	S_5	6.65	7
6	S_6	6.86	7
7	S_7	7.00	7

a b c

FIGURE 3.19 Illustration of histogram equalization of a 3-bit (8 intensity levels) image. (a) Original histogram. (b) Transformation function. (c) Equalized histogram.

- The output is not perfectly flat
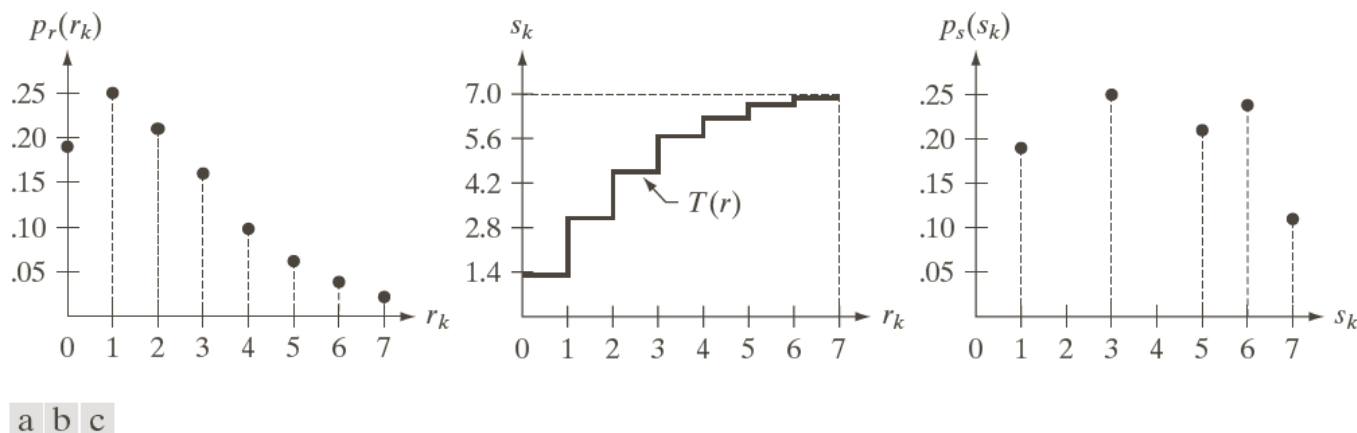 - Caused by the discrete nature of data

a b c
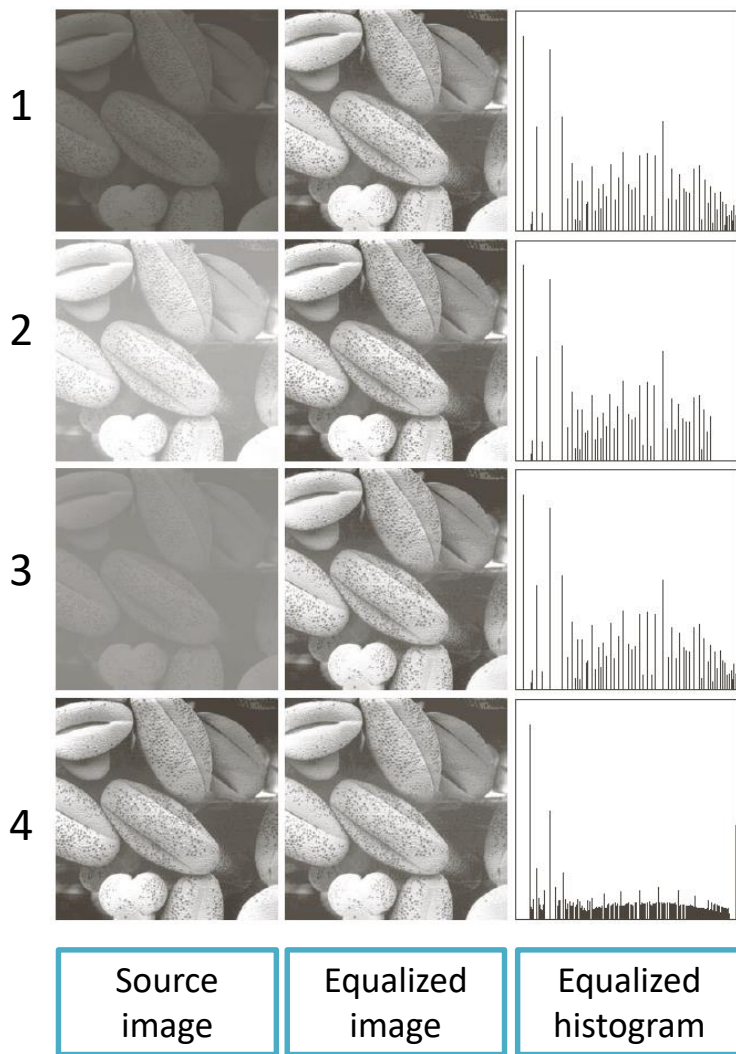
FIGURE 3.19 Illustration of histogram equalization of a 3-bit (8 intensity levels) image. (a) Original histogram. (b) Transformation function. (c) Equalized histogram.

Source image	Equalized image	Equalized histogram

- Here you can see the corresponding input and output images

- Compare the source images

- Compare the equalized images

- What do you observe?

| Source image | Equalized image | Equalized histogram |
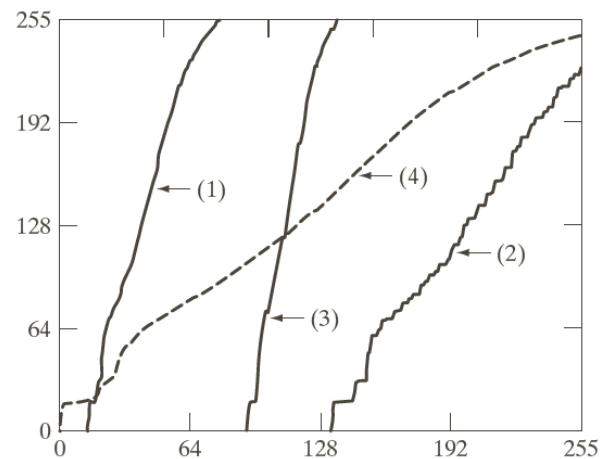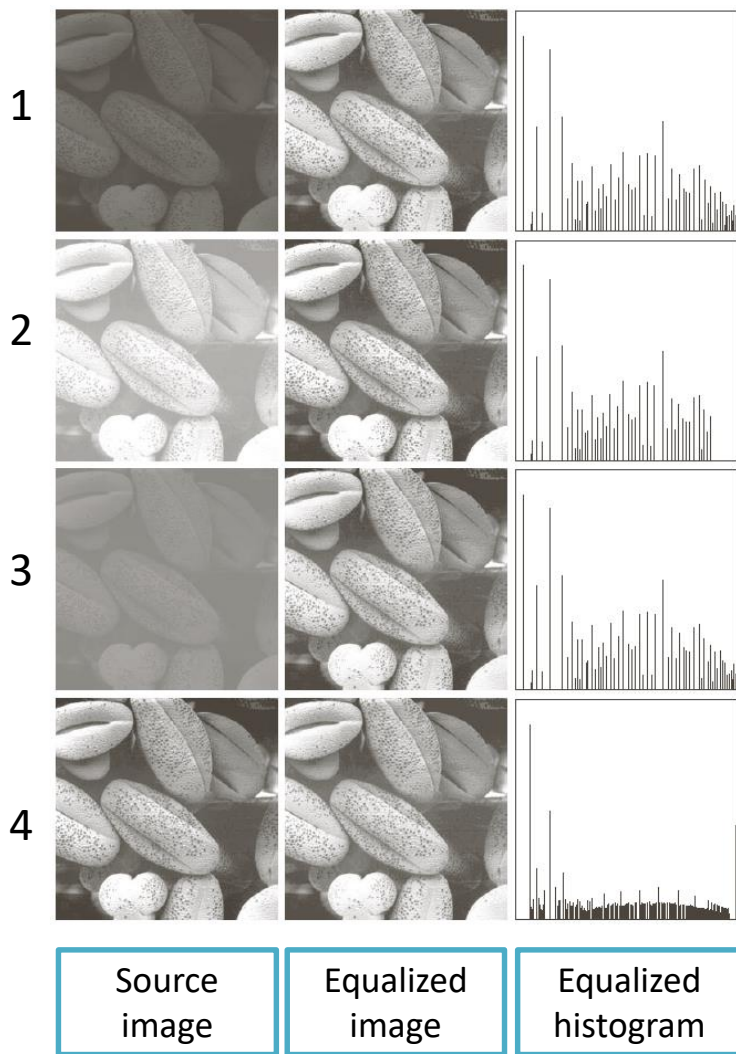
Equalization functions

Histograms in OpenCV

- ## Data structure: array
  - – `cv::Mat` **and** `vector<>`

- ## cv::calcHist function

```
void cv::calcHist(
  cv::InputArrayOfArrays   images,             // vector of 8U or 32F images
  const vector<int>&       channels,           // lists channels to use
  cv::InputArray           mask,               // in 'images' count, iff 'mask'
                                               // nonzero
  cv::OutputArray          hist,               // output histogram array
  const vector<int>        histSize,           // hist sizes in each dimension
  const vector<float>&     ranges,             // pairs give bin sizes in a
                                               // flat list
  bool                     accumulate = false  // if true, add to 'hist', else
                                               // replace
);
```

- Your duty: check the calcHist function and tutorial

§ **calcHist()** [1/3]

```
void cv::calcHist ( const Mat *    images,
                    int            nimages,
                    const int *    channels,
                    InputArray     mask,
                    OutputArray    hist,
                    int            dims,
                    const int *    histSize,
                    const float ** ranges,
                    bool           uniform = true,
                    bool           accumulate = false
                    )
```

Python:

```
hist = cv.calcHist( images, channels, mask, histSize, ranges[, hist[, accumulate]] )
```

Parameters

images   Source arrays. They all should have the same depth, CV_8U, CV_16U or CV_32F , and the same size. Each of them can have an arbitrary number of channels.

nimages   Number of source images.

channels   List of the dims channels used to compute the histogram. The first array channels are numerated from 0 to images[0].channels()-1 , the second array channels are counted from images[0].channels() to images[0].channels() + images[1].channels()-1, and so on.

mask   Optional mask. If the matrix is not empty, it must be an 8-bit array of the same size as images[i] . The non-zero mask elements mark the array elements counted in the histogram.

hist   Output histogram, which is a dense or sparse dims -dimensional array.

dims   Histogram dimensionality that must be positive and not greater than CV_MAX_DIMS (equal to 32 in the current OpenCV version).

histSize   Array of histogram sizes in each dimension.

ranges   Array of the dims arrays of the histogram bin boundaries in each dimension. When the histogram is uniform (uniform =true), then for each dimension i it is enough to specify the lower (inclusive) boundary L_0 of the 0-th histogram bin and the upper (exclusive) boundary $U_{\mathtt{histSize[i]}-1}$ for the last histogram bin histSize[i]-1 . That is, in case of a uniform histogram each of ranges[i] is an array of 2 elements. When the histogram is not uniform (uniform=false), then each of ranges[i] contains histSize[i]+1 elements: $L_0, U_0 = L_1, U_1 = L_2, \ldots, U_{\mathtt{histSize[i]}-2} = L_{\mathtt{histSize[i]}-1}, U_{\mathtt{histSize[i]}-1}$. The array elements, that are not between L_0 and $U_{\mathtt{histSize[i]}-1}$, are not counted in the histogram.

uniform   Flag indicating whether the histogram is uniform or not (see above).

accumulate   Accumulation flag. If it is set, the histogram is not cleared in the beginning when it is allocated. This feature enables you to compute a single histogram from several sets of arrays, or to update the histogram in time.

```cpp
#include <opencv2/imgproc.hpp>
#include <opencv2/highgui.hpp>

using namespace cv;

int main( int argc, char** argv )
{
 Mat src, hsv;
 if( argc != 2 || !(src=imread(argv[1], 1)).data )
 return -1;

 cvtColor(src, hsv, COLOR_BGR2HSV);

 // Quantize the hue to 30 levels
 // and the saturation to 32 levels
 int hbins = 30, sbins = 32;
 int histSize[] = {hbins, sbins};
 // hue varies from 0 to 179, see cvtColor
 float hranges[] = { 0, 180 };
 // saturation varies from 0 (black-gray-white) to
 // 255 (pure spectrum color)
 float sranges[] = { 0, 256 };
 const float* ranges[] = { hranges, sranges };
    MatND hist;
 // we compute the histogram from the 0-th and 1-st channels
 int channels[] = {0, 1};

 calcHist( &hsv, 1, channels, Mat(), // do not use mask
             hist, 2, histSize, ranges,
 true, // the histogram is uniform
 false );
 double maxVal=0;
 minMaxLoc(hist, 0, &maxVal, 0, 0);
```
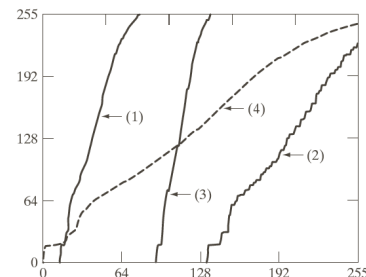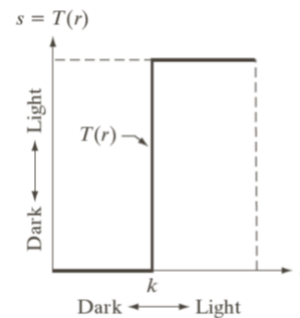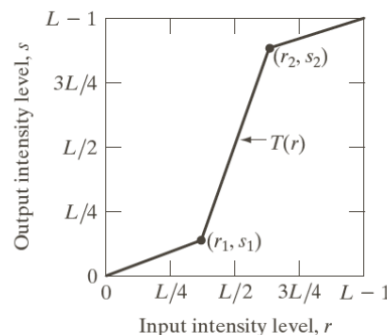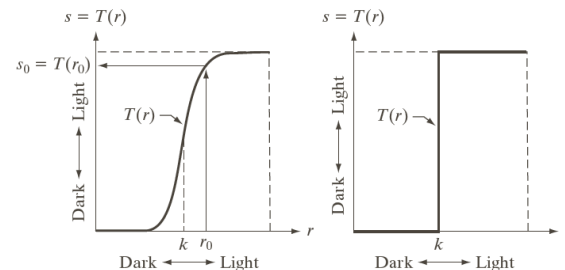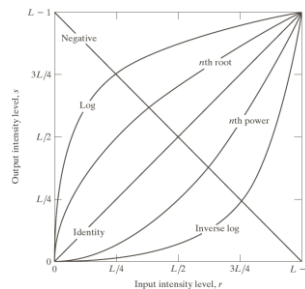
```cpp
int scale = 10;
Mat histImg = Mat::zeros(sbins*scale, hbins*10, CV_8UC3);

for( int h = 0; h < hbins; h++ )
for( int s = 0; s < sbins; s++ )
        {
float binVal = hist.at<float>(h, s);
int intensity = cvRound(binVal*255/maxVal);
rectangle( histImg, Point(h*scale, s*scale),
Point( (h+1)*scale - 1, (s+1)*scale - 1),
Scalar::all(intensity),
CV_FILLED );
        }

namedWindow( "Source", 1 );
imshow( "Source", src );

namedWindow( "H-S Histogram", 1 );
imshow( "H-S Histogram", histImg );
waitKey();
}
```

- ## Recap of the transformations analyzed so far
  - Negative
  - Logarithm
  - Gamma
  - Contrast stretching
  - Thresholing
  - Histogram equalization

Image histogram and equalization

Stefano Ghidoni

DIPARTIMENTO
DI INGEGNERIA
DELL'INFORMAZIONE

UNIVERSITÀ
DEGLI STUDI
DI PADOVA