Image processing
“Points-based operators”

Mathieu Delalandre
University of Tours, Tours city, France
mathieu.delalandre@univ-tours.fr

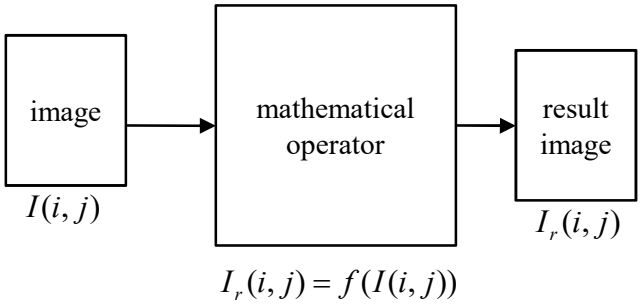
Lecture available at http://mathieu.delalandre.free.fr/teachings/image.html

Points-based operators

1. Introduction
2. Arithmetic and logic operations
3. Thresholding
4. Intensity transformation functions

Introduction

Points based operators deal with single pixel processing, where each pixel value is replaced with a new value obtained from the old one using an operator / function *f*. They could be applied to single or set of images.



e.g.

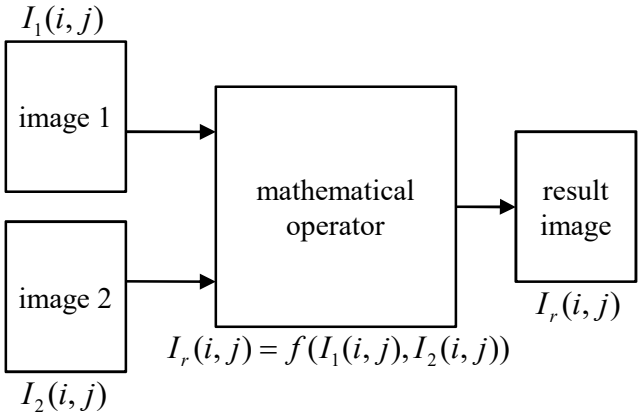
0	0	0	0	0	0
0	0	0	0	0	0
0	0	1	2	2	0
0	0	0	2	3	1
0	0	0	1	1	0

$I(i,j)$

$I(i,j)+1$

1	1	1	1	1	1
1	1	1	1	1	1
1	1	2	3	3	1
1	1	1	3	4	2
1	1	1	2	2	1

$I_r(i,j)$



e.g.

$I_1(i,j)$

0	0	0	0
0	1	3	0
1	2	5	0
0	1	4	3

1	0	2	0
1	0	2	0
1	0	2	0
1	0	2	0

$I_2(i,j)$

$I_1(i,j) \times I_2(i,j)$

0	0	0	0
0	0	6	0
1	0	10	0
0	0	8	0

$I_r(i,j)$

Points-based operators

1. Introduction
2. Arithmetic and logic operations
3. Thresholding
4. Intensity transformation functions

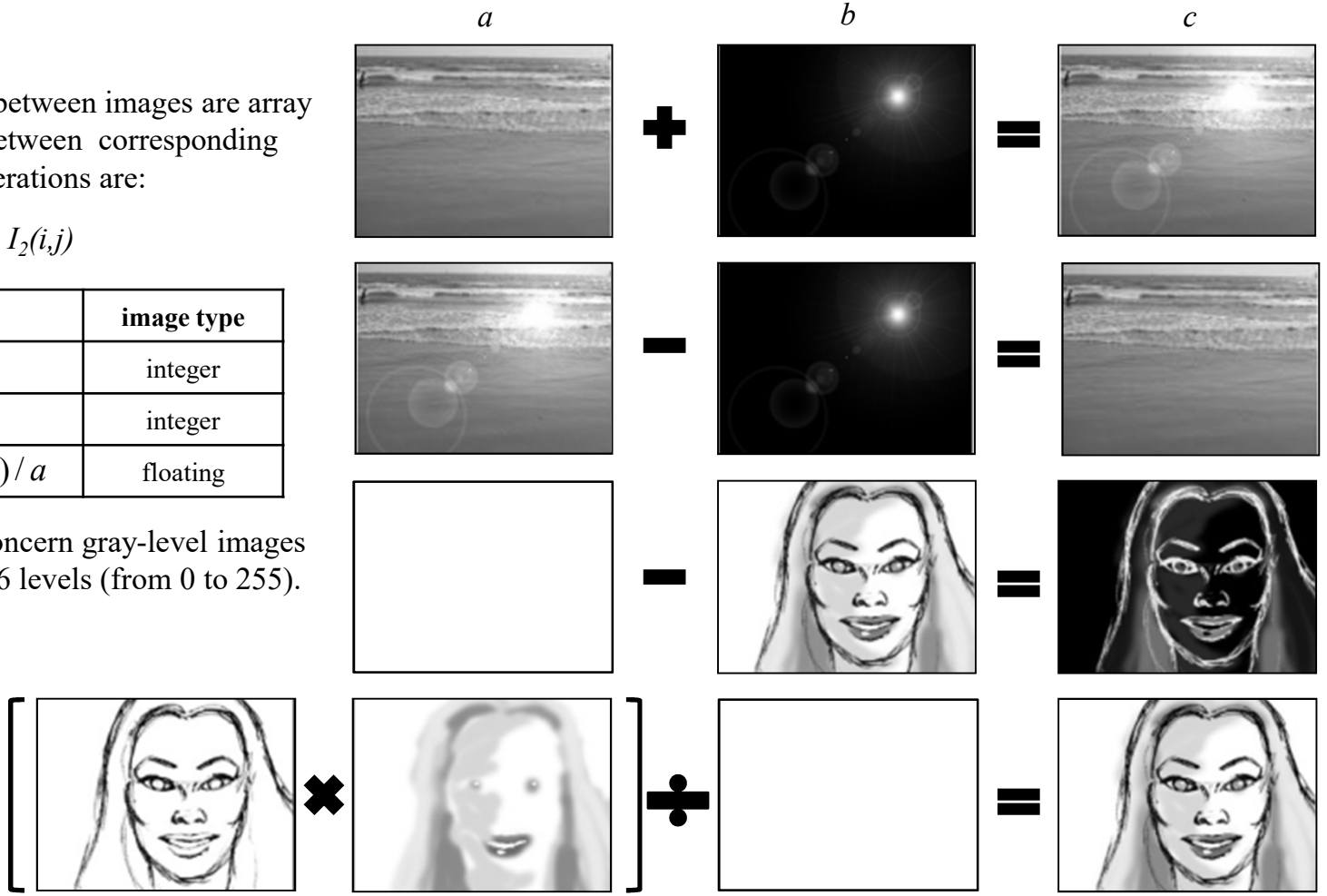
Arithmetic & logic operations (1)

Arithmetic operations between images are array operations carried out between corresponding pixel pairs. Common operations are:

c is $I_r(i,j)$, a is $I_1(i,j)$, b is $I_2(i,j)$

operation	equation	image type
ADD	$c = a + b$	integer
SUB	$c = a - b$	integer
MUL / DIV	$c = (a \times b) / a$	floating

Arithmetic operations concern gray-level images q = 8 corresponds to 256 levels (from 0 to 255).



Arithmetic & logic operations (2)

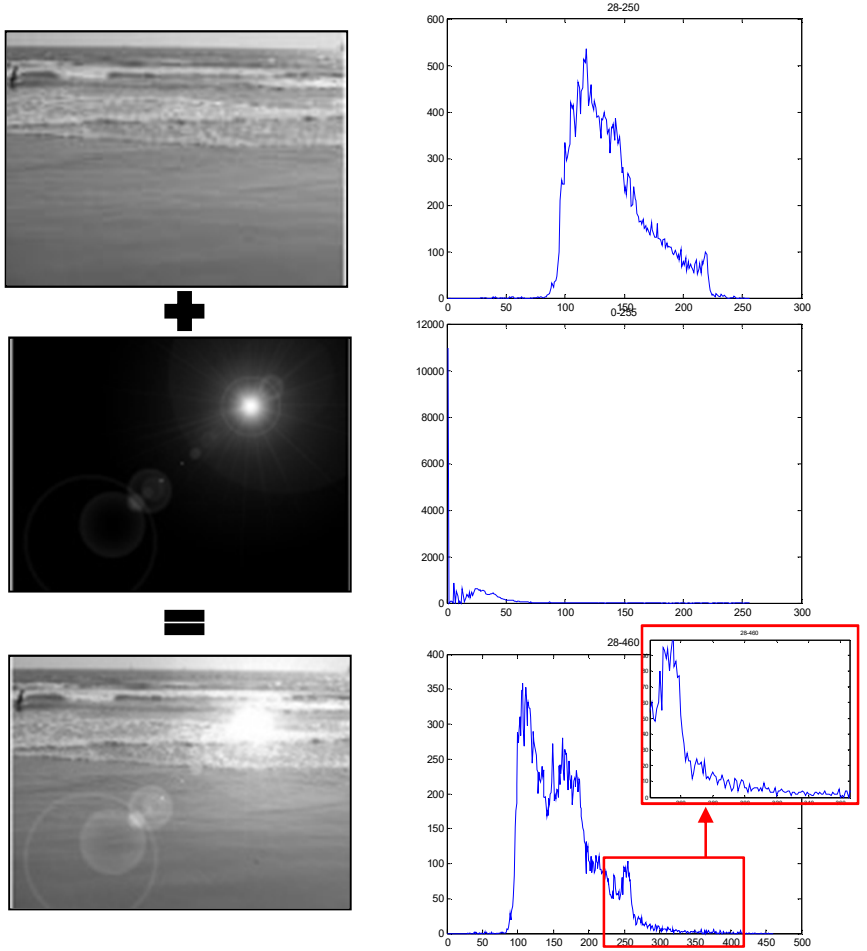
Arithmetic operations between images are array operations carried out between corresponding pixel pairs. Common operations are:

c is $I_r(i,j)$, a is $I_1(i,j)$, b is $I_2(i,j)$

operation	equation	image type
ADD	$c = a + b$	integer
SUB	$c = a - b$	integer
MUL / DIV	$c = (a \times b) / a$	floating

Arithmetic operations concern gray-level images
q = 8 corresponds to 256 levels (from 0 to 255).

When dealing with arithmetic operations, one has to take care to not exceed the maximum intensity level.



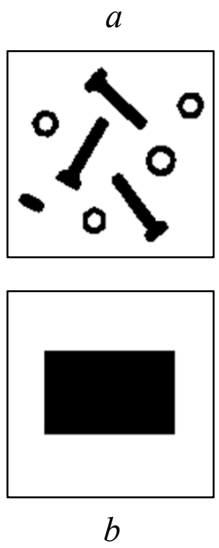
Arithmetic & logic operations (3)

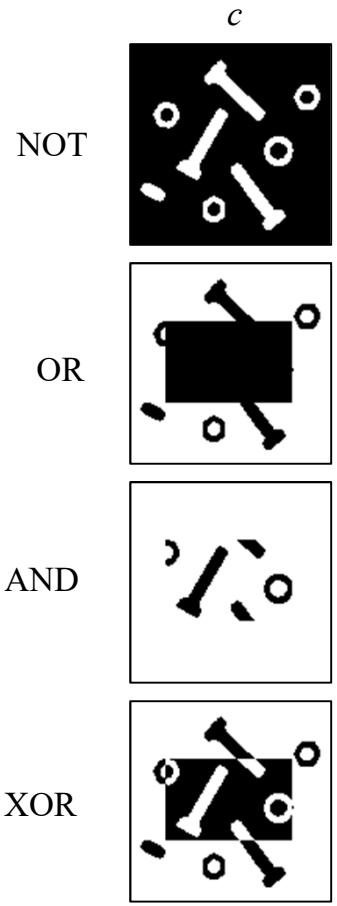
Logical operations: when dealing with images, we can think in terms of foreground (1-valued) and background (0-valued) sets of pixels. It becomes practice to refer to logic operations in which 1 and 0 denote true and false respectively.

Common operations are:

c is $I_r(i,j)$, a is $I_1(i,j)$, b is $I_2(i,j)$

operation	equation	image type
NOT	$c = \overline{a}$	boolean
OR	$c = a + b$	boolean
AND	$c = a \bullet b$	boolean
XOR	$c = a \oplus b = a \bullet \overline{b} + \overline{a} \bullet b$	boolean





Points-based operators

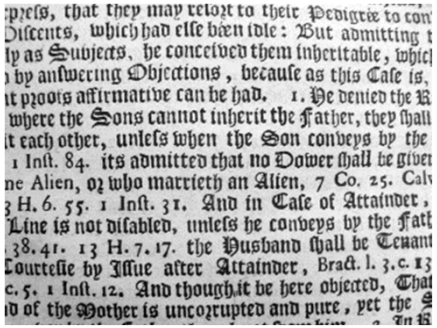
1. Introduction
2. Arithmetic and logic operations
3. Thresholding
4. Intensity transformation functions

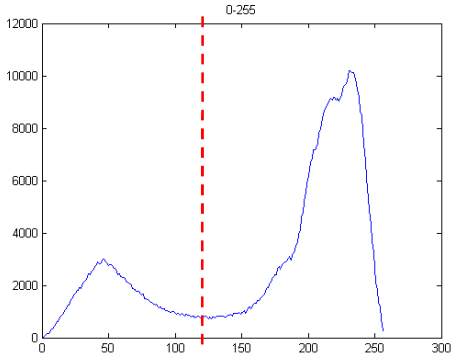
Thresholding (1)

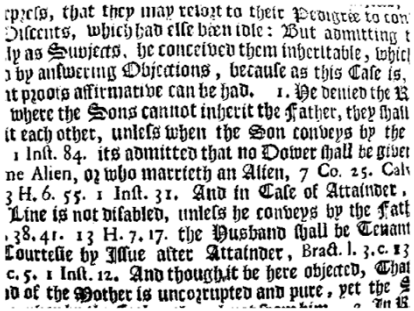
Thresholding: during the thresholding process, individual pixels in an image are marked as "object" pixels if their values belong to some threshold values and as "background" pixels otherwise.

Single thresholding $I_r(i,j) = \begin{cases} v_1 & if \quad I(i,j) \le T \\ v_2 & otherwise \end{cases}$

v_1, v_2 could be I(i,j), a constant, a rule







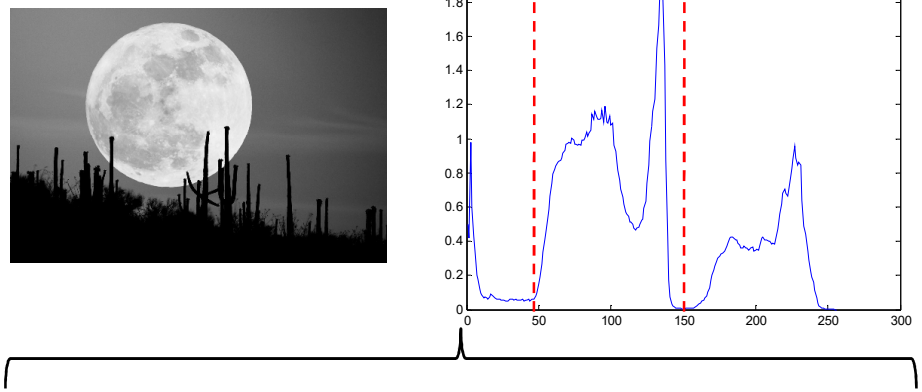
$I_r(i,j) = \begin{cases} 0 & if \quad I(i,j) \le 128 \\ 255 & otherwise \end{cases}$

Thresholding (2)

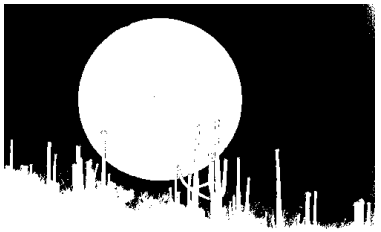
Thresholding: during the thresholding process, individual pixels in an image are marked as "object" pixels if their values belong to some threshold values and as "background" pixels otherwise.

Double thresholding $I_r(i,j) = \begin{cases} v_1 & if \quad T_1 \le I(i,j) \le T_2 \ with \ T_1 < T_2 \\ v_2 & otherwise \end{cases}$

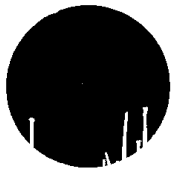
v_1, v_2 could be I(i,j), a constant, a rule



$$I_r(i,j) = \begin{cases} 0 & if \quad I(i,j) < 50 \\ 255 & otherwise \end{cases}$$



$$I_r(i,j) = \begin{cases} 0 & if \quad 45 \le I(i,j) \le 150 \\ 255 & otherwise \end{cases}$$



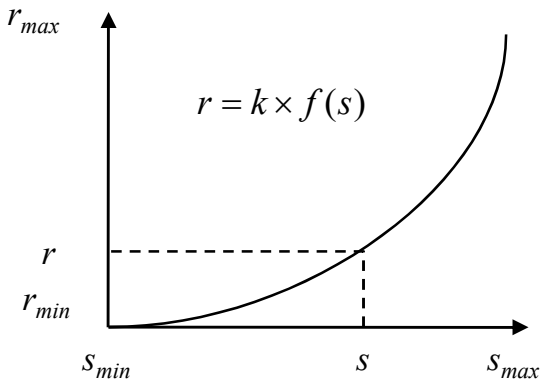
$$I_r(i,j) = \begin{cases} 0 & if \quad I(i,j) > 150 \\ 255 & otherwise \end{cases}$$

Points-based operators

1. Introduction
2. Arithmetic and logic operations
3. Thresholding
4. Intensity transformation functions

Intensity transformation functions
"Introduction"

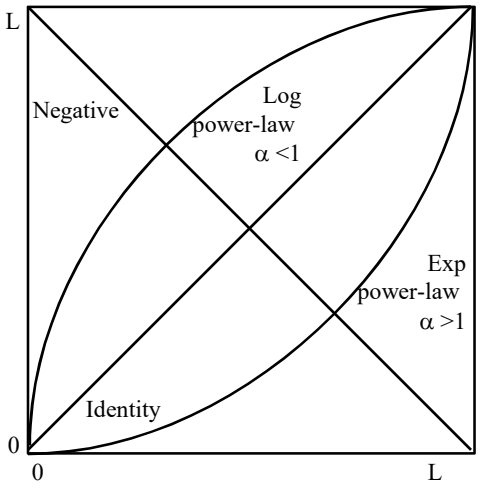
Intensity transformation functions: these functions are related by an expression of the form $r = k \times f(s)$ with r as $I_r(i,j)$, s as $I(i,j)$, f a transformation function that maps the pixel value s into a pixel value r with k a normalization parameter that depends of $L=2^q-1$.

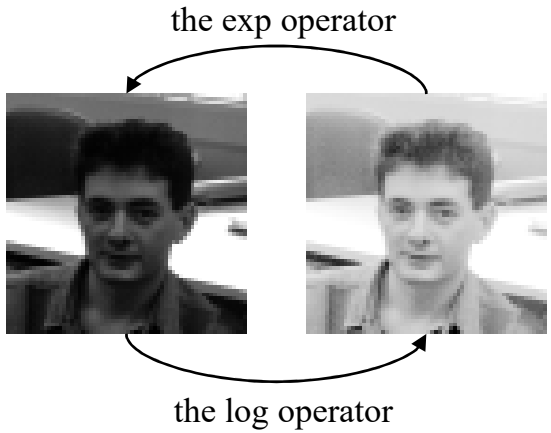


$$r = k \times f(s) \quad \text{with} \quad k = \frac{L}{f(L)}$$

Common functions are:

functions	***f(s)***	**image type**
INDENTITY	s	integer
NEGATIVE	$L - s$	integer
LOG	$\log(1+s)$	floating
EXP	$\exp(s)-1$	floating
SQRT	$\sqrt{s}$	floating
Power-law	s^{α}	floating
Gaussian	$\frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(s-\mu)^2}{2\sigma^2}}$	floating
…	and so on	…



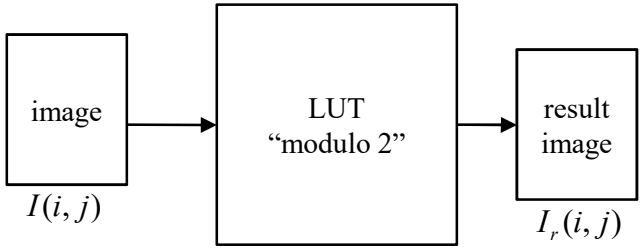


Intensity transformation functions
"LookUp Table (LUT)" (1)

Lookup table (LUT) is a data structure used to replace a runtime computation with a simpler array indexing operation. Operations are computed offline, entries and results are structured through the LUT data structure (e.g. array). Time processing is reduced to a single access O(1).

	complexity
LUT	O(1)
Operation	>O(1)

e.g. modulo operation



$$I_r(i,j) = \mathrm{mod}(I(i,j),2)$$

$I(i,j)$	$I_r(i,j)$
0	0
1	1
2	0
3	1
4	0
5	1
…	…

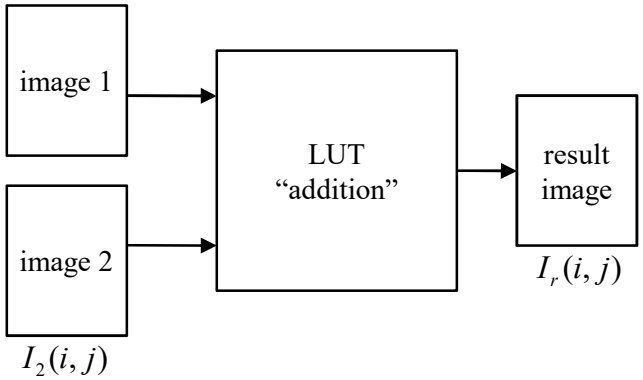
input	output
2^q	2^q

Intensity transformation functions
"LookUp Table (LUT)" (2)

Lookup table (LUT) is a data structure used to replace a runtime computation with a simpler array indexing operation. Operations are computed offline, entries and results are structured through the LUT data structure (e.g. array). Time processing is reduced to a single access O(1).

	complexity
LUT	O(1)
Operation	>O(1)

e.g. addition operation



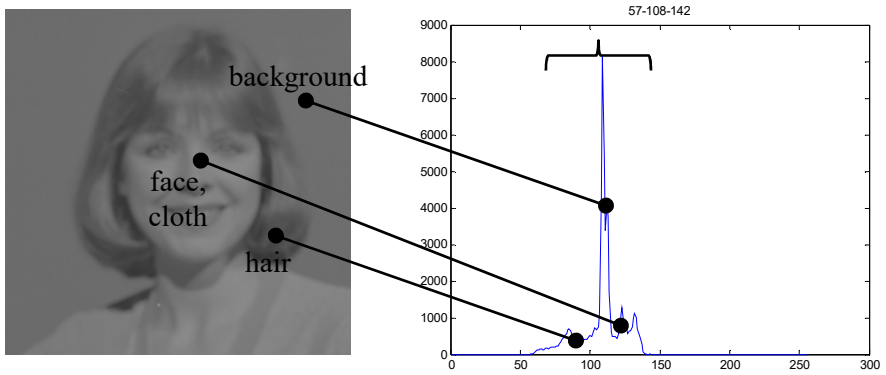
$$I_r(i,j) = I_1(i,j) + I_2(i,j)$$

$I_1(i,j)$	$I_2(i,j)$	$I_r(i,j)$
0	0	0
0	1	1
0	2	2
1	0	1
1	1	2
1	2	3
2	0	2
2	1	3
2	2	4

input	output
$2\times2^q=2^{q+1}$	$(2^q)^2=2^{2q}$

Intensity transformation functions
“Use-case: log-based” (1)

How to restore the following image with an intensity transformation function?



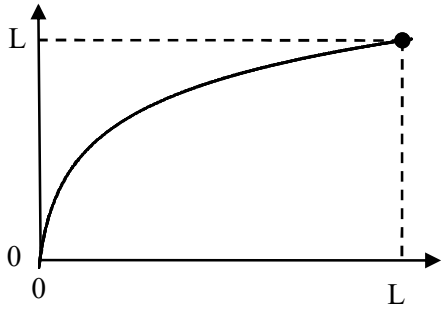
It looks like a log based transformation problem, but with two open problems.

(1) How to adapt the transformation function to the weak amplitude of the histogram?

(2) Signal pattern is face, cloth; how to adapt the transformation function to the top part of the histogram?

Intensity transformation functions
“Use-case: log-based” (2)

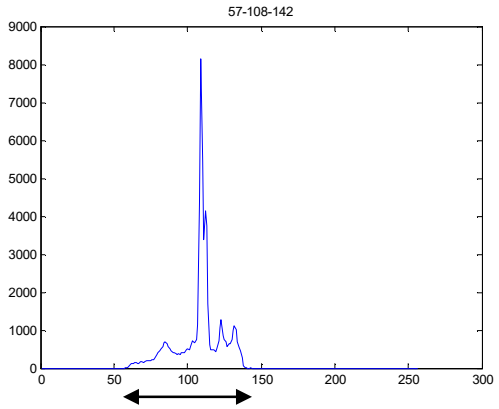
(1) How to adapt the transformation function to the weak amplitude of the histogram?
Application of the direct log transformation function results in reducing the histogram amplitude and increasing the mean value.



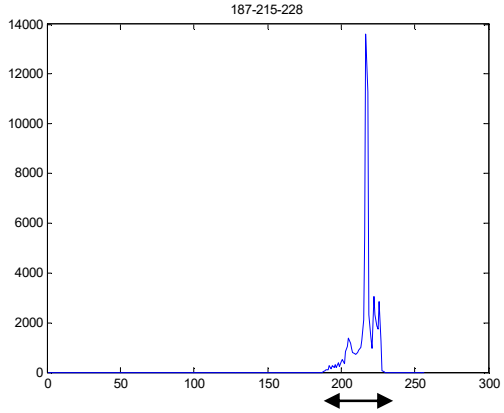
$$r = k \times f(s) = k \times \log(1+s)$$

$$k \times \log(1+L) = L$$

$$k = \frac{L}{\log(1+L)} \qquad \text{if q = 8} \qquad k = \frac{255}{\log(256)} \approx 45.98$$



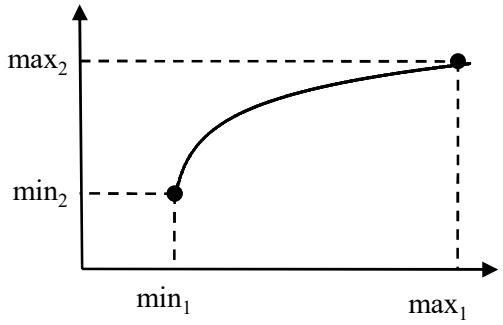
amplitude	85
mean	108



amplitude	41
mean	215

Intensity transformation functions
“Use-case: log-based” (3)

(1) How to adapt the transformation function to the weak amplitude of the histogram?
Application of an **adapted log** transformation function respects the histogram amplitude, however the histogram is still shifted (i.e. mean) and it increases the amplitude mainly in the bottom part of the histogram.

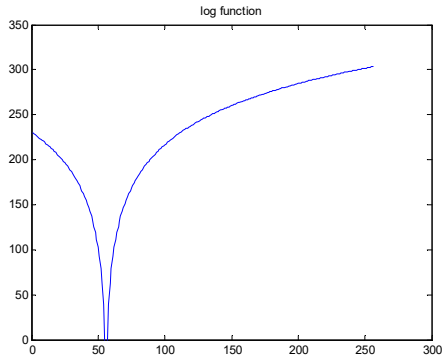


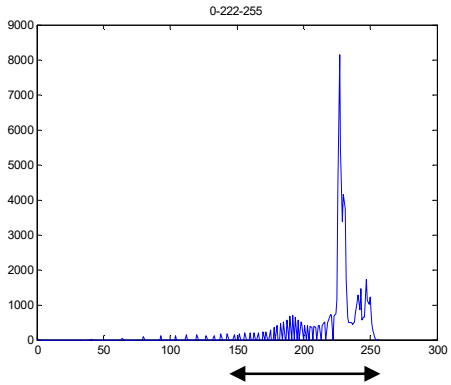
$$\max_2 = \min_2 + k \times \log(1 + \max_1 - \min_1)$$

$$k = \frac{\max_2 - \min_2}{\log(1 + \max_1 - \min_1)} = \frac{\Delta_2}{\log(1 + \Delta_1)} \quad \begin{cases} \Delta_2 = \max_2 - \min_2 \\ \Delta_1 = \max_1 - \min_1 \end{cases}$$

$$r = \min_2 + k \times \log(1 + s - \min_1)$$

$$r = C_2 + k \times \log(s + C_1) \quad \begin{cases} C_2 = \min_2 \\ k = \dfrac{\Delta_2}{\log(1 + \Delta_1)} \\ C_1 = 1 - \min_1 \end{cases}$$

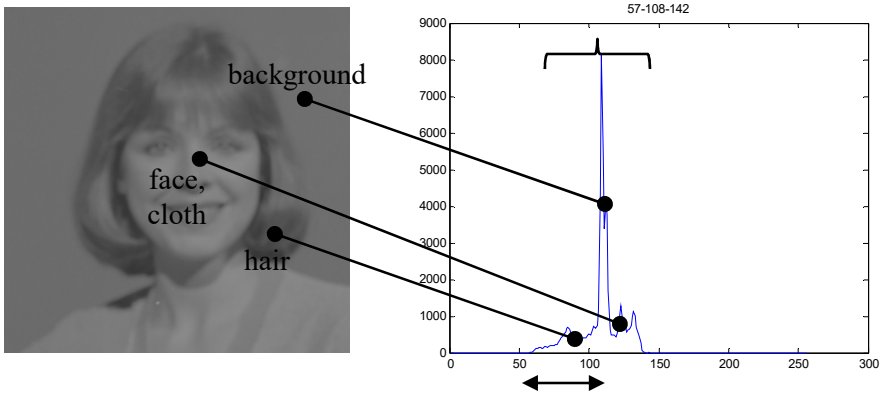




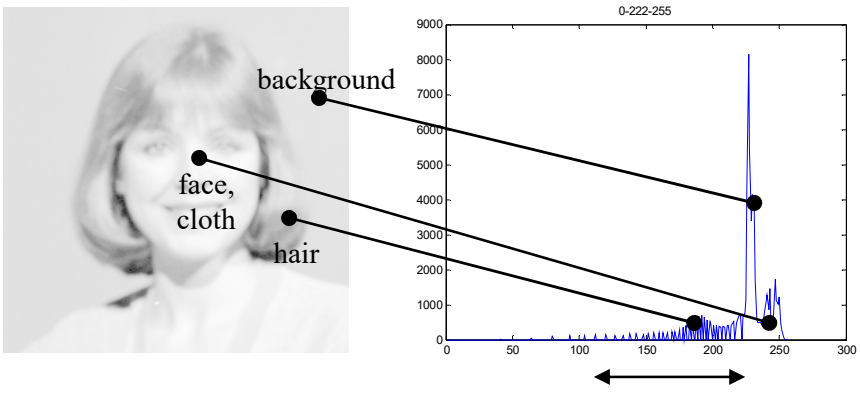
amplitude	≈105
mean	222

Intensity transformation functions
“Use-case: log-based” (4)

(1) How to adapt the transformation function to the weak amplitude of the histogram?
Application of an **adapted log** transformation function respects the histogram amplitude, however the histogram is still shifted (i.e. mean) and it increases the amplitude mainly in the bottom part of the histogram.



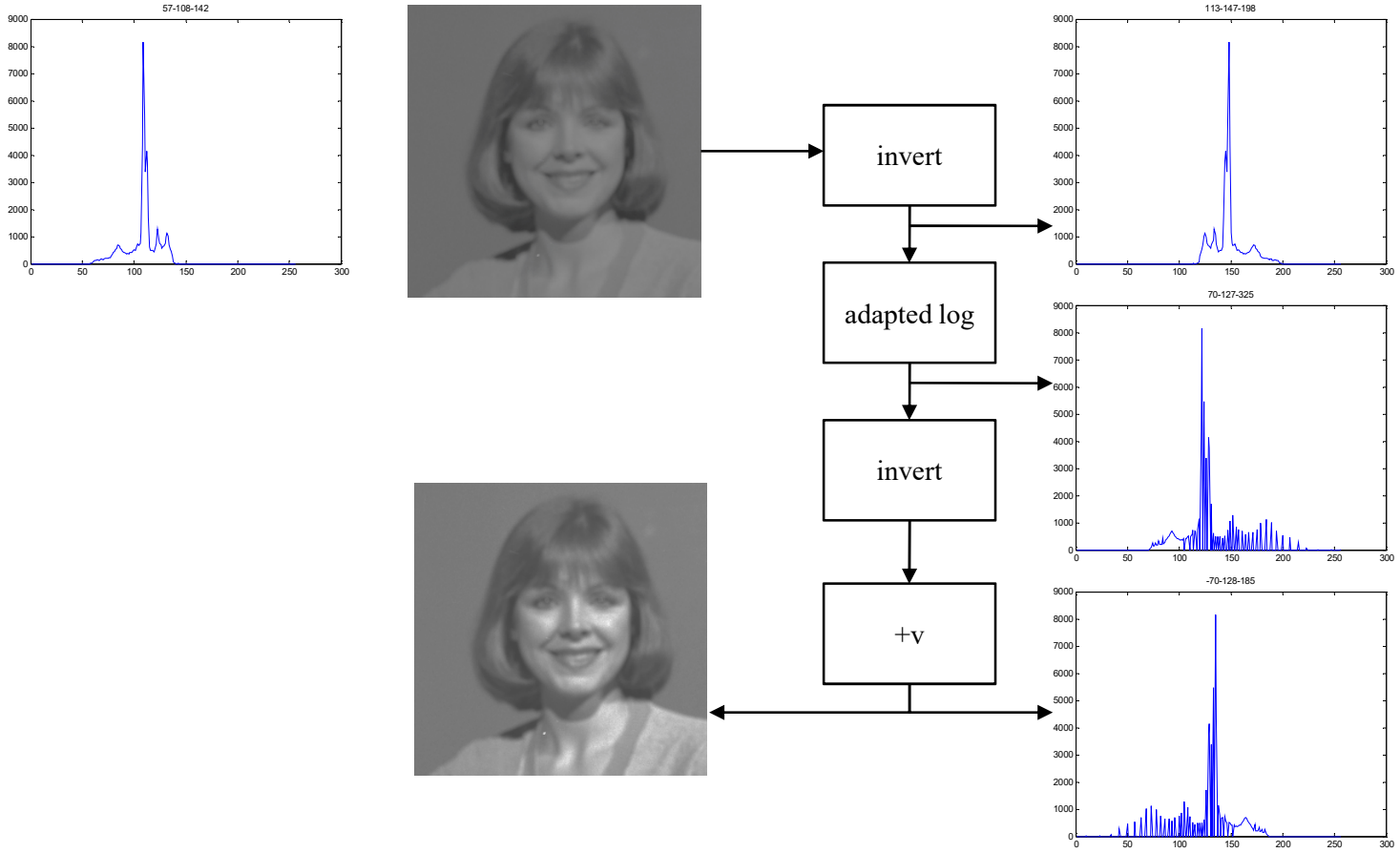
bottom part of
the histogram



bottom part of
the histogram

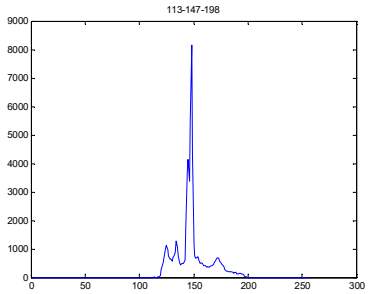
Intensity transformation functions
“Use-case: log-based” (5)

(2) Signal pattern is face, cloth; how to adapt the transformation function to the top part of the histogram? Application of the **adapted log** transformation function on the negative image will target the bottom part, mean shift can be compensated with an arithmetic operation.

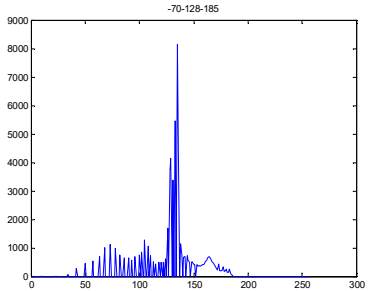


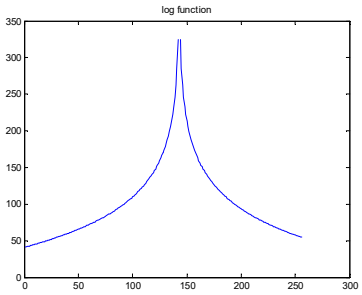
Intensity transformation functions
“Use-case: log-based” (6)

(2) Signal pattern is face, cloth; how to adapt the transformation function to the top part of the histogram?
The previous processing requires a four-step operation, we can embed all of them in a single intensity transformation function to save processing time.



invert
adapted log
invert + v





$$r = v + \overline{k \times f(\bar{s})}$$

$$r = v + L - k \times f(\bar{s})$$

$$f(\bar{s}) = \min_2^{'} + k^{'} \times \log\left(1 + (L - s) - \min_1^{'}\right)$$

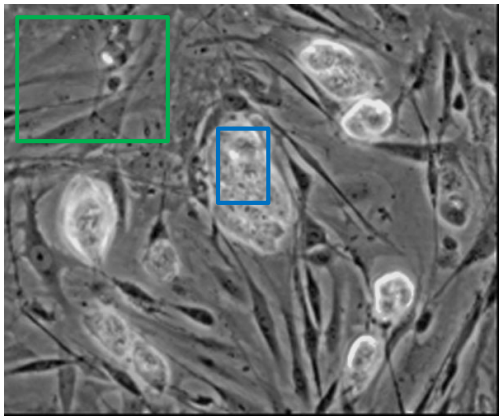
$$r = C_2 - k^{'} \times \log(C_1 - s)$$

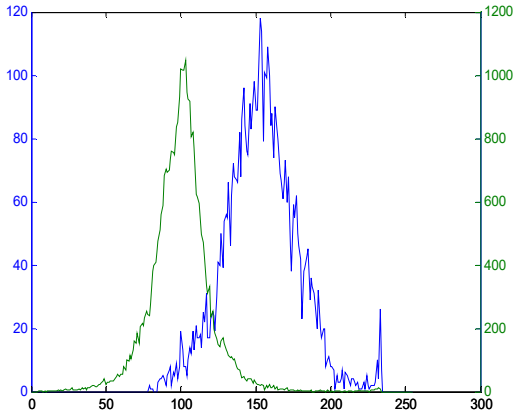
$$\begin{cases} \min_1^{'} = L - \max_1 \\ \max_1^{'} = L - \min_1 \\ \min_2^{'} = \min_2 \\ \max_2^{'} = \max_2 \end{cases}$$

$$\begin{cases} C_2 = v + L - \min_2^{'} \\ k^{'} = \dfrac{\Delta_2^{'}}{\log\left(1 + \Delta_1^{'}\right)} \\ \Delta_2^{'} = \max_2^{'} - \min_2^{'} \\ \Delta_1^{'} = \max_1^{'} - \min_1^{'} \\ C_1 = 1 + L - \min_1^{'} \end{cases}$$

Intensity transformation functions
“Use-case: Gaussian-based” (1)

How to enhance the visualization of cells cytoplasm?



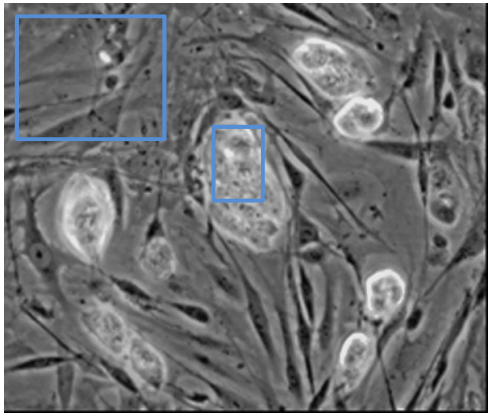


It is a two-steps problem where

(1) Probability distribution of cells intensities can be used to design an intensity transformation function.

(2) This function should take into account the background / foreground separation problem to minimize the background detection artifacts.

Intensity transformation functions “Use-case: Gaussian-based” (2)



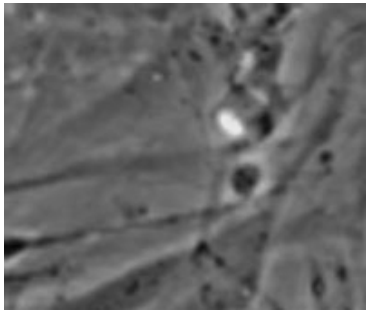
(1) Probability distribution of cells intensities can be used to design an intensity transformation function.

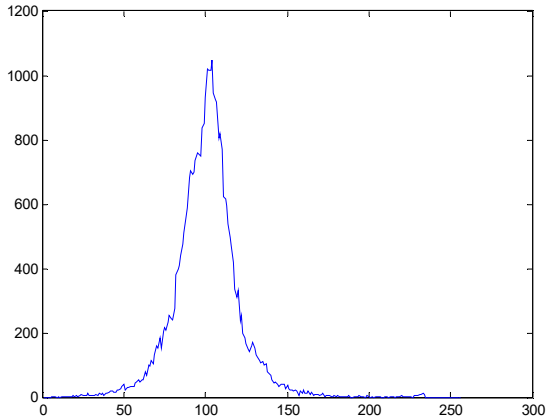
The background and foreground distributions look-like pseudo-Gaussian functions.

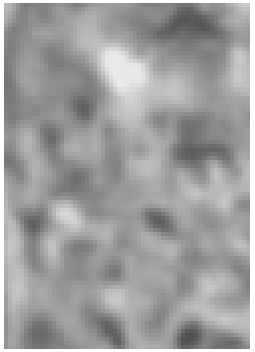
e.g. the Matlab code to display the distributions.

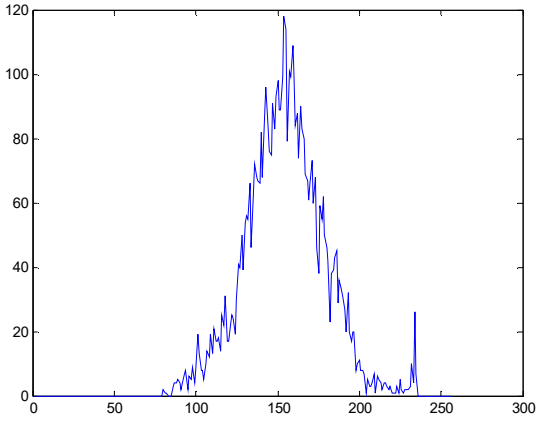
main

```
img = imread('cells8.tif');
figure; imshow(img);

cell = img(170:255,250:310);
figure; imshow(cell);
h1 = imhist(cell); figure; plot(h1);

back = img(1:170,1:200);
figure; imshow(back);
h2 = imhist(back); figure; plot(h2);
```









Intensity transformation functions “Use-case: Gaussian-based” (3)

(1) Probability distribution of cells intensities can be used to design an intensity transformation function.

We can design a Gaussian distribution and set it manually, to fit with the histogram distribution.

$$f(s) = k\frac{1}{\sigma\sqrt{2\pi}}e^{-\frac{(s-\mu)^2}{2\sigma^2}}$$

e.g. the Matlab code to overlay the Gaussian function and the histogram distribution.

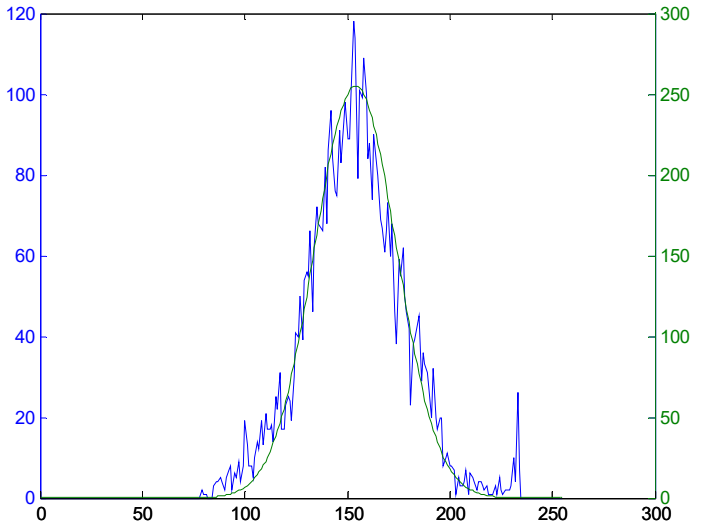
$$\begin{cases} f(s=\mu) = k\dfrac{1}{\sigma\sqrt{2\pi}}e^{0} = L \\ k = \sigma\sqrt{2\pi}L \\ \mu = 155, \sigma = 20 \end{cases}$$

main

```
m = 155; v = 20.0;
f = gF (255,m,v);

img = imread('cells8.tif');
cell = img(170:255,250:310);
h = imhist(cell);

figure; plotyy(0:255,h',0:255,f);
```

function

```
function f = gF (L,m,v)
k = v*sqrt(2*pi)*L;
f = zeros(1,256);
for i=1:256
    f(i) = gaussian(k,v,m,i);
end
```



function

```
function y = gaussian(k,v,m,x)
y = k * 1 / (v*sqrt(2*pi)) * exp(- (x-m)^2 / (2*v^2) );
```

Intensity transformation functions “Use-case: Gaussian-based” (4)

(1) Probability distribution of cells intensities can be used to design an intensity transformation function.

To design a Gaussian distribution, it makes sense to set it automatically with an estimation of parameters from the histogram distribution.

$$f(s) = k\frac{1}{\sigma\sqrt{2\pi}}e^{-\frac{(s-\mu)^2}{2\sigma^2}}$$

e.g. the Matlab code to compute σ,μ and to overlay the function and the distribution.

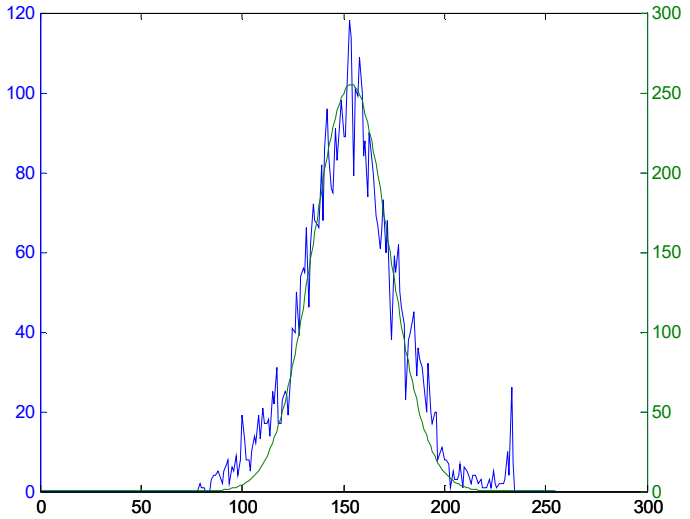
$$\begin{cases} f(s=\mu) = k\dfrac{1}{\sigma\sqrt{2\pi}}e^{0} = L \\ k = \sigma\sqrt{2\pi}L \\ \mu = 157.7, \sigma = 18.7 \end{cases}$$

main

```
img = imread('cells8.tif');  s = size(img);
cell = img(170:255,250:310);
h = imhist(cell);

[m,v] = estimateH(h)
f = gF(255,m,v);
figure; plotyy(0:255,h1',0:255,f);
```

function

```
function [m,v] = estimateH(h)
m = 0;
for i=1:256
    m = m + h(i)*i;
end
m = m/sum(h);
v = 0;
for i=1:256
    v = v + sqrt(((i-m)*h(i))^2);
end
v = v/sum(h);
```



Intensity transformation functions “Use-case: Gaussian-based” (5)

(1) Probability distribution of cells intensities can be used to design an intensity transformation function.

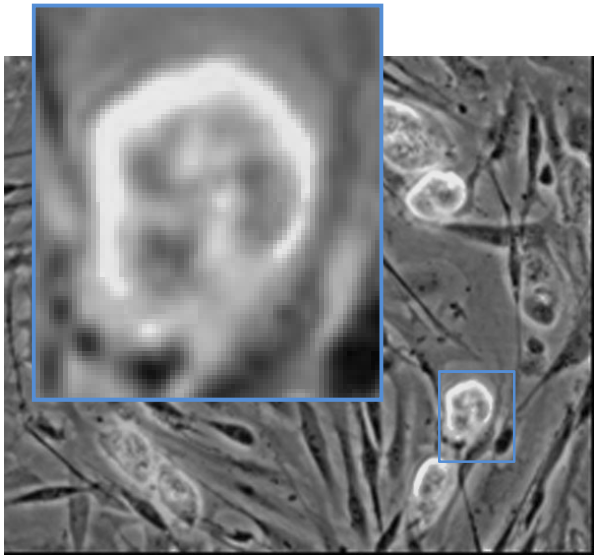
The Gaussian distribution will display the frequent pixels in white (i.e. the negative of the target), we can embedded the negative process within the distribution.

e.g. the Matlab code to get $\overline{f(s)}$ through a LUT and to apply it to the image.

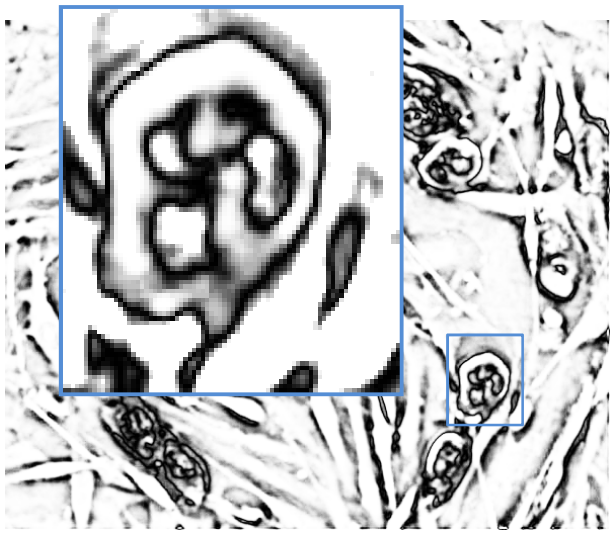
main
```
max = 255; m = 155; v = 20.0;
f = gFC(max,m,v);
img = imread('cells8.tif');
imgr = transform(img,f);
figure; imshow(uint8(imgr));
```

function
```
function f = gFC(L,m,v)
k = v*sqrt(2*pi)*L;
f = zeros(1,256);
for i=1:256
    f(i) = L - gaussian(k,v,m,i);
end
```

function
```
function Ir = transform(I,f)
s = size(I);  Ir = 255*ones(s(1,1),s(1,2));
for i=1:s(1,1)
    for j=1:s(1,2)
        x = I(i,j);
        Ir(i,j) = f(x+1);
    end
end
```



$$\overline{f(s)} = L - k\frac{1}{\sigma\sqrt{2\pi}}e^{-\frac{(s-\mu)^2}{2\sigma^2}}$$



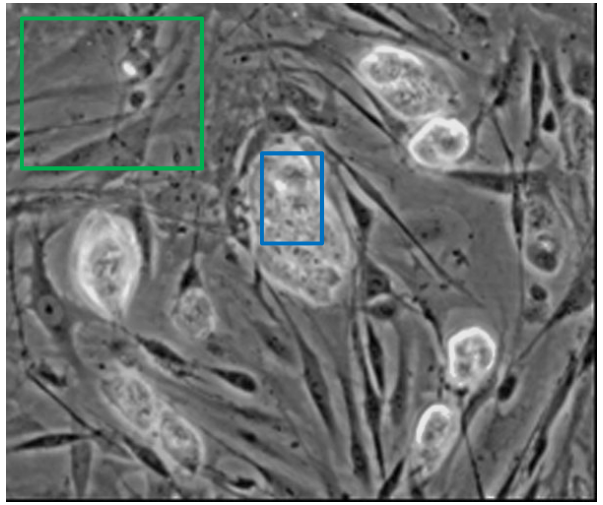
Intensity transformation functions “Use-case: Gaussian-based” (6)

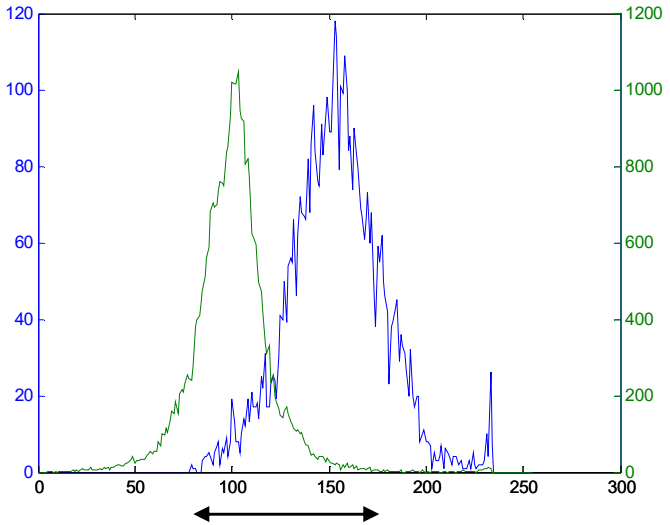
(2) This function should take into account the background / foreground separation problem to minimize the background detection artifacts.

e.g. the Matlab code to overlay the two histogram distributions.

main

```
img = imread('cells8.tif');

cell = img(170:255,250:310);
h1 = imhist(cell);
[m1,v1] = estimate(h1);

back = img(1:170,1:200);
h2 = imhist(back);
[m2,v2] = estimate(h2);

figure; plotyy(0:255,h1,0:255,h2);
```





overlap between the two distributions

Intensity transformation functions "Use-case: Gaussian-based" (7)

(2) This function should take into account the background / foreground separation problem to minimize the background detection artifacts.

To minimize the background artifacts, we can model the background with another function to deduct from the foreground function. When the negative process is embedded and the intensity levels bounded, the final function $g_{\min}(s)$ is given as

$$\overline{g_{\min}(s)} = \min\left(L, \overline{g(s)}\right)$$

$$\begin{cases} \overline{g(s)} = L - (f_1(s) - f_2(s)) \\ \overline{g(s)} = L + f_2(s) - f_1(s) \\ f_1(s) = k_1 \dfrac{1}{\sigma_1\sqrt{2\pi}} e^{-\frac{(s-\mu_1)^2}{2\sigma_1^2}} \\ f_2(s) = k_2 \dfrac{1}{\sigma_2\sqrt{2\pi}} e^{-\frac{(s-\mu_2)^2}{2\sigma_2^2}} \end{cases}$$

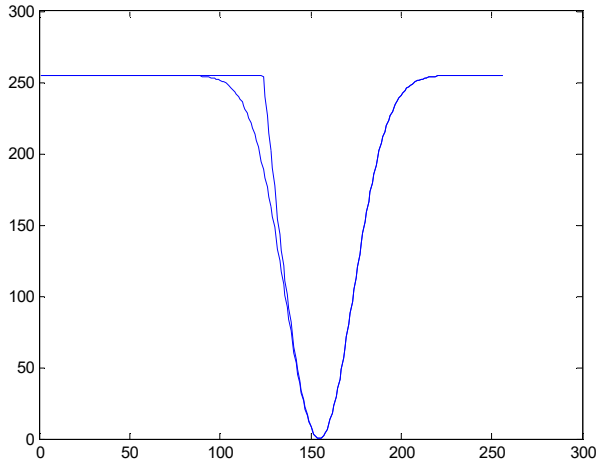
e.g. the Matlab code to get the functions through LUTs, to set them automatically, and to apply them to the image.

main

```
f1 = gFC(255,m1,v1);
figure; plot(f1);
f2 = gGMC(255,m1,v1,m2,v2);
hold on; plot(f2);

img = imread('cells8.tif');

imgr = transform(img,f1);
figure; imshow(uint8(imgr));

imgr = transform(img,f2);
figure; imshow(uint8(imgr));
```

function

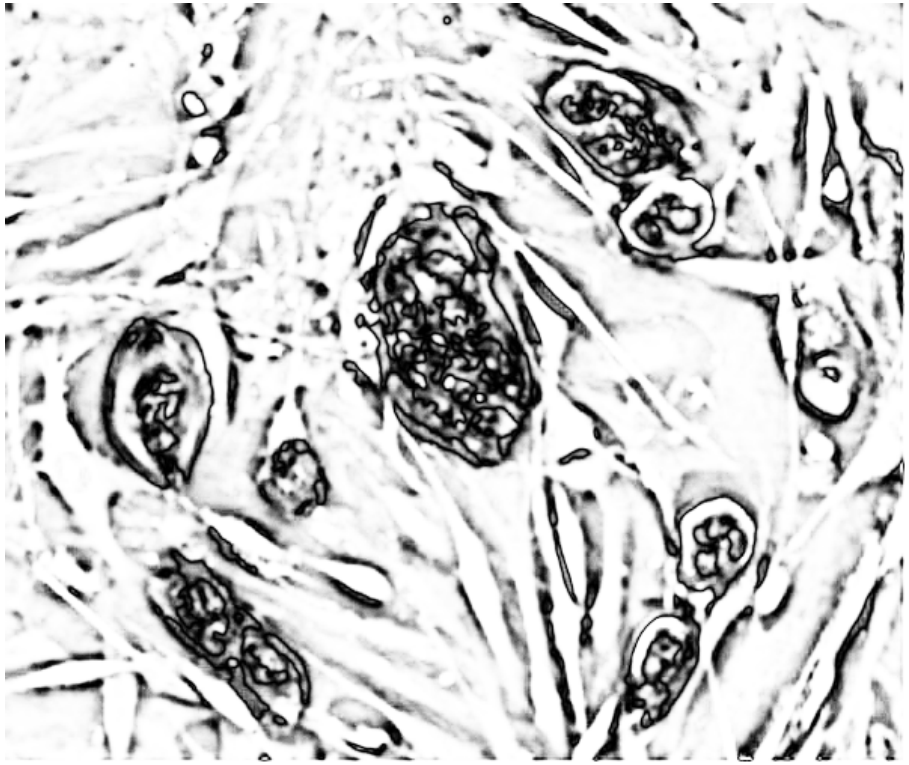
```
function f = gGMC(L,m1,v1,m2,v2)

k1 = v1*sqrt(2*pi)*L;
k2 = v2*sqrt(2*pi)*L;

f = zeros(1,256);
for i=1:256
    f(i) = L
    + gaussian(k2,v2,m2,i)
    - gaussian(k1,v1,m1,i);
    f(i) = min(L,f(i));
end
```



Intensity transformation functions “Use-case: Gaussian-based” (8)

(2) This function should take into account the background / foreground separation problem to minimize the background detection artifacts.

with $\overline{f(s)}$



with $\overline{g_{\min}(s)}$