

HAN AEA - Embedded Vision & Machine Learning

EVD1 - Week 1



Image Fundamentals Histogram Operations

By Hugo Arends

Image Fundamentals

- Functions for creating and deleting images
- Functions for converting images
- Functions for reading and writing pixels
- Basic image processing operators
- Scaling

Scaling

- Used to enhance contrast
- Used to scale larger pixel datatypes to smaller pixel datatypes
e.g. float_pixel_t to uint8_pixel_t
- Min-max scaling is defined as

$$p_{dst}(x, y) = \frac{dst_{max} - dst_{min}}{src_{max} - src_{min}} \cdot (p_{src}(x, y) - src_{min}) + dst_{min}$$

where

src_{min}: global minimum of the source image

src_{max}: global maximum of the source image

dst_{min}: global minimum of the destination image

dst_{max}: global maximum of the destination image

Scaling

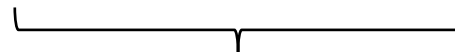
- Used to enhance contrast
- Used to scale larger pixel datatypes to smaller pixel datatypes
e.g. float_pixel_t to uint8_pixel_t
- Min-max scaling is defined as

$$p_{dst}(x, y) = \underbrace{\frac{dst_{max} - dst_{min}}{src_{max} - src_{min}}}_{\substack{\text{Scale factor} \\ \text{(fraction)} \\ = \\ \frac{\text{new range}}{\text{old range}}}} \cdot (p_{src}(x, y) - src_{min}) + dst_{min}$$

Scaling

- Used to enhance contrast
- Used to scale larger pixel datatypes to smaller pixel datatypes
e.g. float_pixel_t to uint8_pixel_t
- Min-max scaling is defined as

$$p_{dst}(x, y) = \frac{dst_{max} - dst_{min}}{src_{max} - src_{min}} \cdot (p_{src}(x, y) - src_{min}) + dst_{min}$$

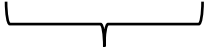


Scale with respect to
the lowest pixel
value in the source
image

Scaling

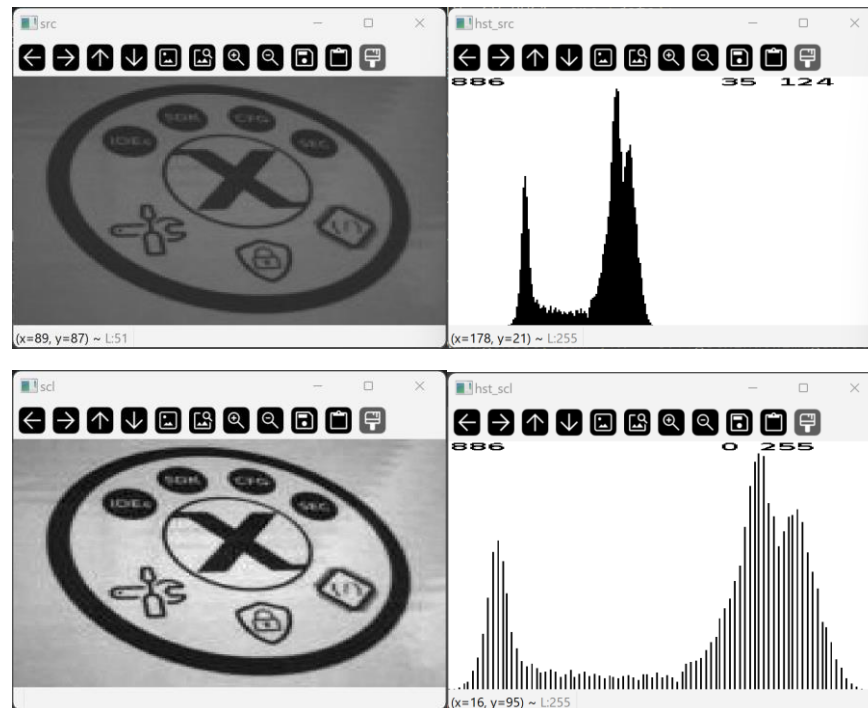
- Used to enhance contrast
- Used to scale larger pixel datatypes to smaller pixel datatypes
e.g. float_pixel_t to uint8_pixel_t
- Min-max scaling is defined as

$$p_{dst}(x, y) = \frac{dst_{max} - dst_{min}}{src_{max} - src_{min}} \cdot (p_{src}(x, y) - src_{min}) + dst_{min}$$


Move scaled values
to new minimum
value

Scaling - example

$$p_{dst}(x, y) = \frac{255 - 0}{src_{max} - src_{min}} \cdot (p_{src}(x, y) - src_{min}) + 0$$



Scaling - algorithm

```
void scale(    const image_t *src, image_t *dst);
```

See file **evdk_operators\image_fundamentals.c**

Histogram Operations

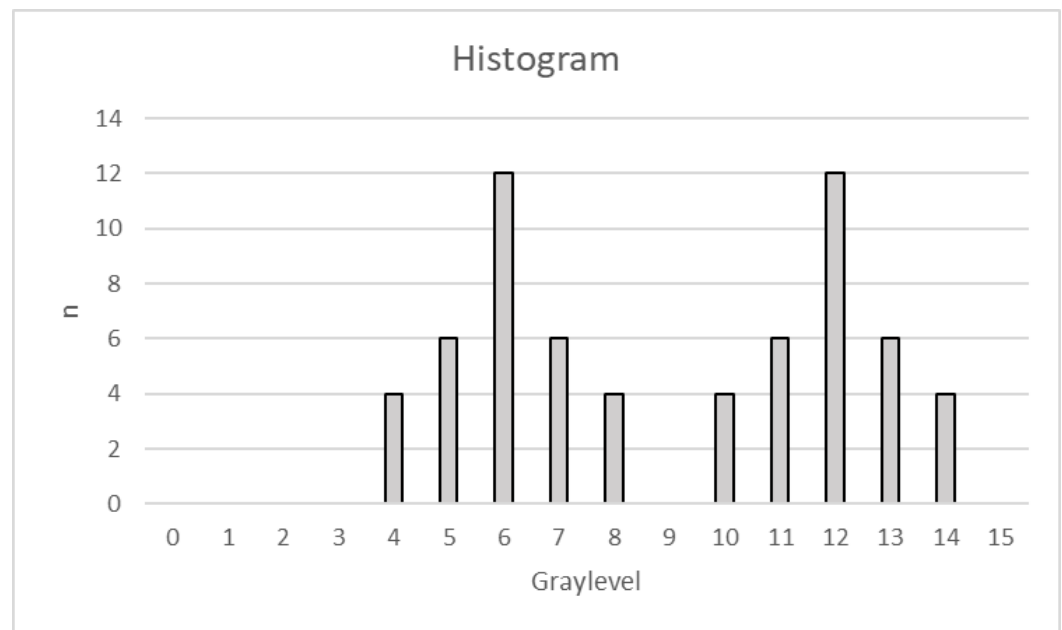
- Are operations that do not require spatial information
- Treat the graylevels as a set of numbers represented by a histogram
- Modifying the graylevel histogram improves visual appearance
- Histogram
- Brightness correction
- Contrast correction

Histogram

- Shows for each graylevel the number of pixels in the image

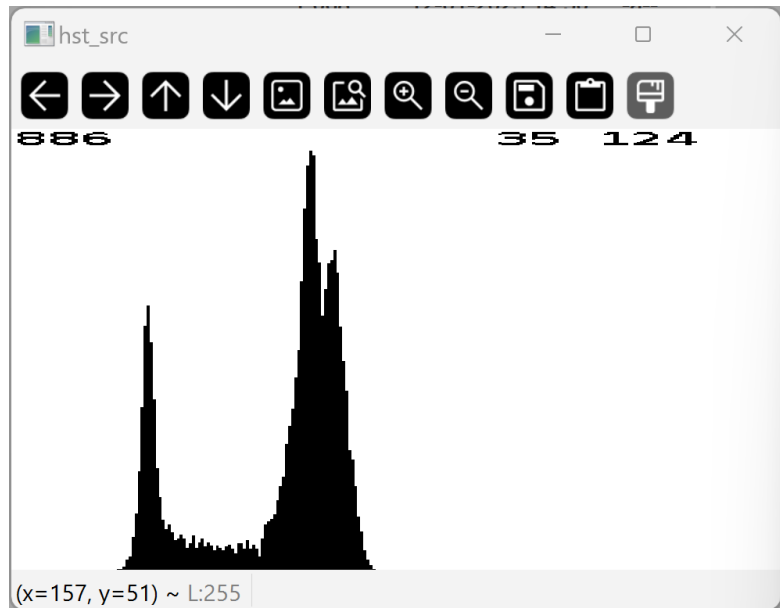
image

4	4	4	4	5	5	5	5
5	5	6	6	6	6	6	6
6	6	6	6	6	6	7	7
7	7	7	7	8	8	8	8
10	10	10	10	11	11	11	11
11	11	12	12	12	12	12	12
12	12	12	12	12	12	13	13
13	13	13	13	14	14	14	14



Histogram - example

See apps example evdk5_histogram_webcam



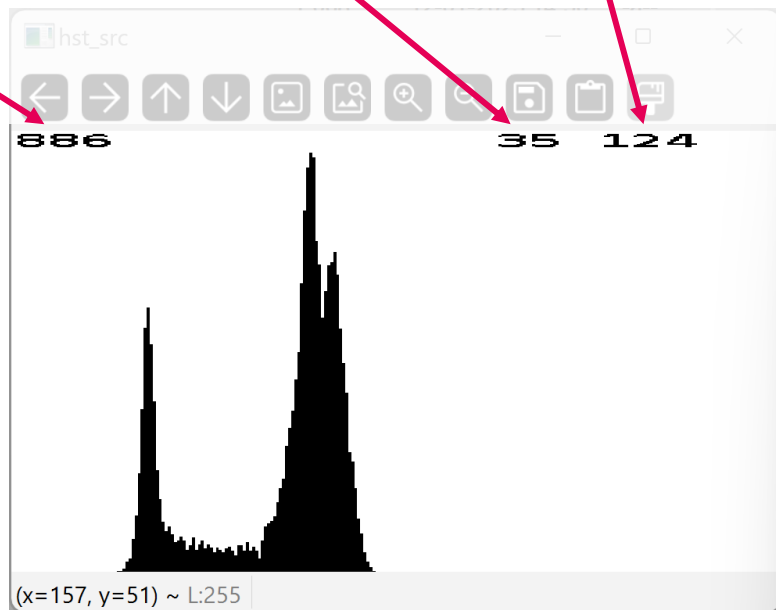
Histogram - example

See apps example `evdk5_histogram_webcam`

Max number of pixels
for single graylevel

min
graylevel

max
graylevel



Histogram - algorithm

```
void histogram(  const image_t *img, uint32_t *hist);
```

See file **EVDK_Operators\histogram_operations.c**

*The function does not check memory boundaries. It simply assumes that the **hist** pointer points to memory allocated by the caller of this function. The size of the histogram must be 256 times a uint32_t.*

Brightness correction

- Enhances the visual appearance of an image
- Brightness modification is defined as

$$s_{(x,y)} = g_{(x,y)} + \textit{brightness}$$

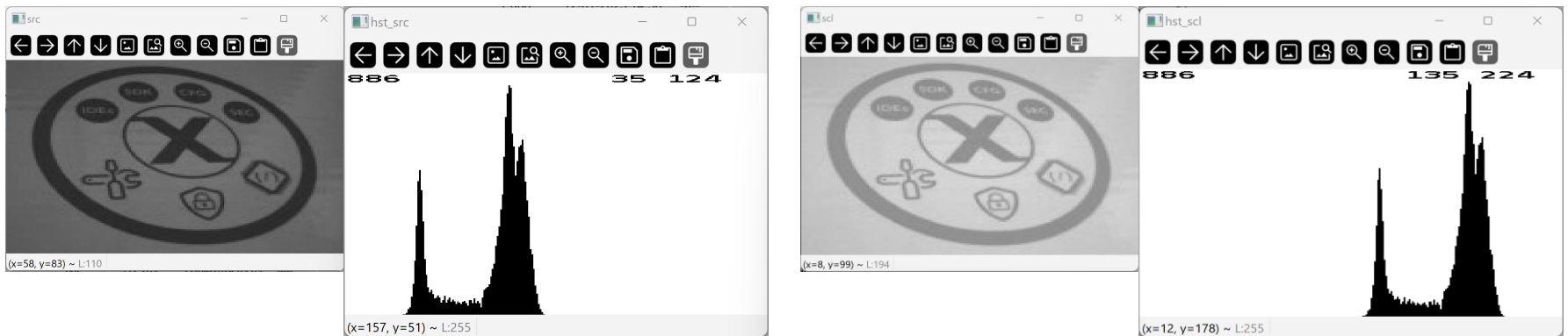
where

$s_{(x,y)}$: *graylevel of the enhanced pixel at (x, y)*

$g_{(x,y)}$: *graylevel of the original pixel at (x, y)*

Brightness correction - example

brightness = 100



Brightness correction - example

brightness = 200



Brightness correction - algorithm

```
void brightness( const image_t *src, image_t *dst,  
                 const uint32_t brightness);
```

See file **EVDK_Operators\histogram_operations.c**

Contrast correction

- Enhances the visual appearance of an image
- Contrast correction is defined as

$$s_{(x,y)} = \text{contrast} \cdot (g_{(x,y)} - \text{average}) + \text{average}$$

where

$s_{(x,y)}$: graylevel of the enhanced pixel at (x, y)

$g_{(x,y)}$: graylevel of the original pixel at (x, y)

average : mean pixel value of the original image given by

$$\text{average} = \frac{1}{n_x n_y} \sum_{x=0}^{n_x} \sum_{y=0}^{n_y} p(x, y)$$

where

n_x : number of columns of the image

n_y : number of rows of the image

$p(x, y)$: graylevel of the original pixel at (x, y)

Contrast correction

- Calculating the average

$$average = \frac{1}{n_x n_y} \underbrace{\sum_{x=0}^{n_x} \sum_{y=0}^{n_y} p(x, y)}$$

The sum of
all pixel
values, can
be calculated
in a double
for-loop

Contrast correction

- Calculating the average

$$average = \frac{1}{\underbrace{n_x n_y}} \sum_{x=0}^{n_x} \sum_{y=0}^{n_y} p(x, y)$$

Divided by
the number
of pixels:
 $rows \times cols$

Contrast correction

- Calculate the contrast modification

$$s_{(x,y)} = contrast \cdot \underbrace{(g_{(x,y)} - average)} + average$$

Distance to the
average pixel
value (positive,
zero, or negative)

Contrast correction

- Calculate the contrast modification

$$s_{(x,y)} = \underbrace{\text{contrast} \cdot (g_{(x,y)} - \text{average})}_{\text{Change the distance to the average pixel value}} + \text{average}$$

Contrast correction

- Calculate the contrast modification

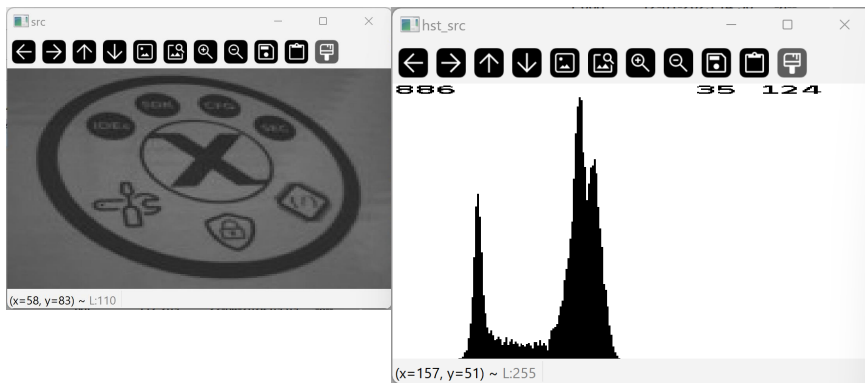
$$s_{(x,y)} = contrast \cdot (g_{(x,y)} - average) + average$$

└──────────┘

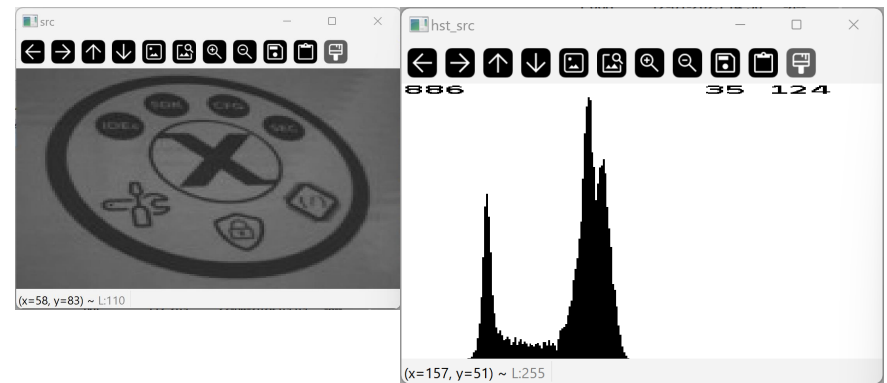
Move
back to
original
average
pixel
value

Contrast correction - example

$contrast = 1$
Contrast is equal



↑
average

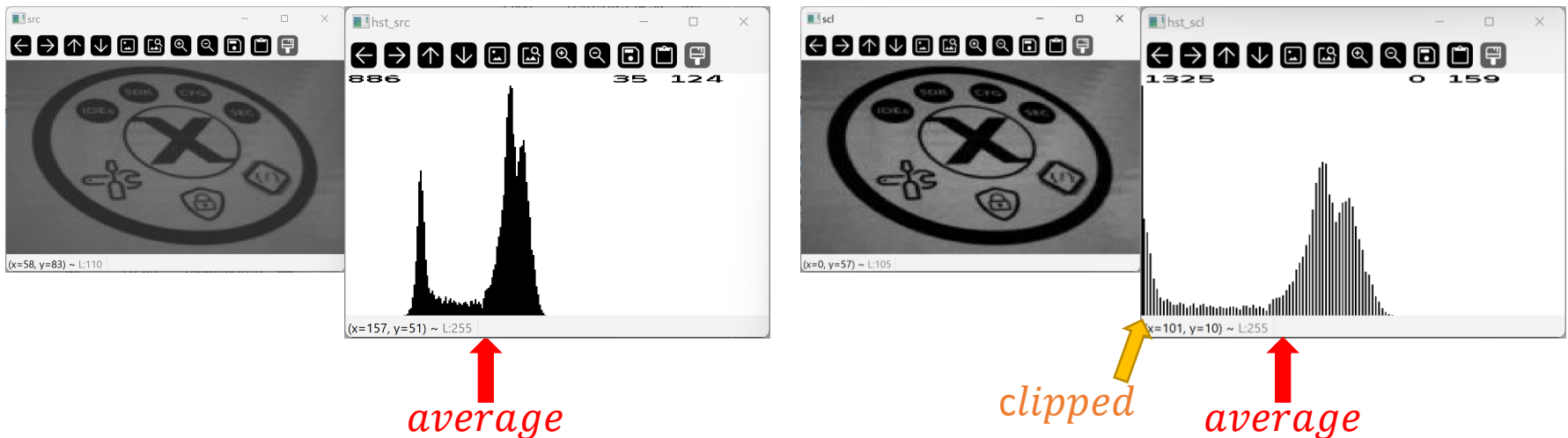


↑
average

Contrast correction - example

$$\text{contrast} = 2$$

The histogram shows twice the contrast, however, **the results are clipped**



Contrast correction - example

$$\text{contrast} = 0.5$$

The histogram shows half the contrast



Contrast correction - example

$contrast = 10$

The histogram shows ten times the contrast, **clipping the result**



EVD1 – Assignment



Study guide

Week 1

8 Histogram operations – contrast()

References

- Myler, H. R., & Weeks, A. R. (2009). *The pocket handbook of image processing algorithms in C*. Prentice Hall Press.