# *Image Processing I Exercise Class*

WS 2017/2018

Noha Sarhan

Cognitive Vision Group, Department of Informatics
University of Hamburg, Germany

# Assignment #2

1. It was noted in the lecture that it is possible to compute the variance of the image in one run over the image. Starting with the formula you learned in the lecture, derive a formula that allows you to do so.

$$\sigma_I^2 = \frac{1}{N_{cols} \cdot N_{rows}} \sum_{x=1}^{N_{cols}} \sum_{y=1}^{N_{rows}} [I(x,y) - \mu_I]^2$$

2. Write your own MATLAB function to compute the variance. Use this function to compute the variance of any gray-scale image and compare the result to the MATLAB built in **var** function.

Stanford - An introduction to MATLAB Handouts

# Assignment #2

3. Implement a function hist_dist_Euclidean which takes as an input two histograms h1 and h2 and return the Euclidean distance between them.

$$D_{L2} = \sqrt{\sum_i \left(h_1(i) - h_2(i)\right)^2}$$

4. Implement a function hist_dist_Manhattan which takes as an input two histograms h1 and h2 and returns the Manhattan distance between them

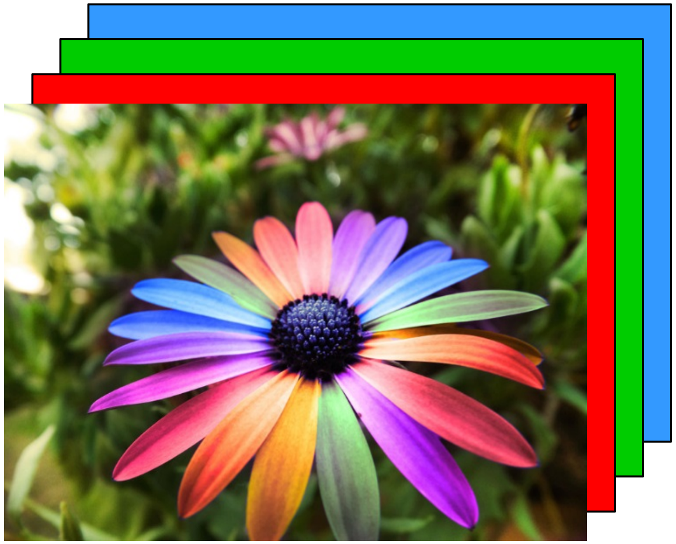
$$D_{L1} = \sum_i |h_1(i) - h_2(i)|$$

5. Implement a function hist_dist_intersection which takes as an input two histograms h1 and h2 and returns their intersection

$$D_{\cap} = \sum_i \min(h_1, h_2)$$

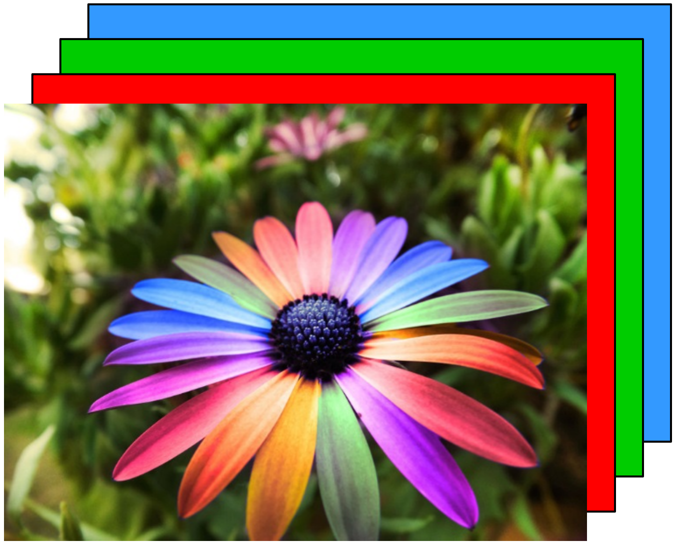
# Color images

- For a color image, the matrix has a third extra dimension. This dimension has only 3 elements, and they represent the three channels namely R, G, and B.



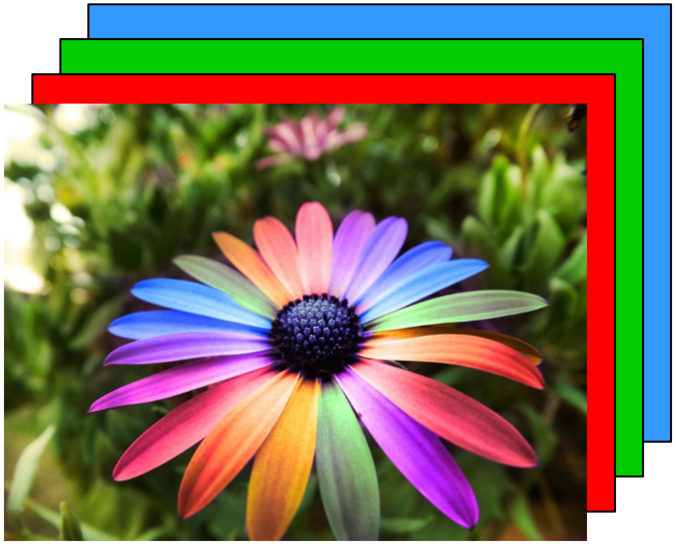
# Color images

- For a color image, the matrix has a third extra dimension. This dimension has only 3 elements, and they represent the three channels namely R, G, and B.
- Read the colored image “**peppers.png**”
  - `img = imread(‘peppers.png’);`



# Color images

- For a color image, the matrix has a third extra dimension. This dimension has only 3 elements, and they represent the three channels namely R, G, and B.
- Read the colored image “**peppers.png**”
  - `img = imread(‘peppers.png’);`
- Convert to grayscale:
  - `img_gray = rgb2gray (img);`



# *Histogram of color images*

- Create a histogram for each color channel.

-
  ```
  img = imread (‘peppers.png’);
  figure (1), imhist(img(:, :, 1));
  figure (2), imhist(img(:, :, 2));
  figure (3), imhist(img(:, :, 3));
  ```

# *Color spaces*

RGB → HSV (hue, saturation & value)

- `hsv_img = rgb2hsv(rgb_img)`
- **hsv2rgb()** goes the other way round

# Color spaces

RGB → HSV (hue, saturation & value)

- `hsv_img = rgb2hsv(rgb_img)`
- **hsv2rgb()** goes the other way round

RGB → LAB

- `lab_img = rgb2lab (rgb_img)`
- **lab2rgb()** goes the other way round

# Binary images

- Converting a grayscale image to a binary image by thresholding

$$g(x,y) \Rightarrow \begin{cases} 0 & if\ g(x,y) < T \\ 1 & if\ g(x,y) \geq T \end{cases}$$

# Binary images

- Converting a grayscale image to a binary image by thresholding

$$g(x,y) \Rightarrow \begin{cases} 0 & if\ g(x,y) < T \\ 1 & if\ g(x,y) \geq T \end{cases}$$

- How to choose a threshold?

# Binary images

- Converting a grayscale image to a binary image by thresholding

$$g(x,y) \Rightarrow \begin{cases} 0 & \text{if } g(x,y) < T \\ 1 & \text{if } g(x,y) \geq T \end{cases}$$

Hint: use the function **B = logical (A)** to convert from uint8 to logical/binary

- How to choose a threshold?
  - Load the image “**coins.png**”
  - Create a new black & white (binary) version of it using the above conditions
    - Set the threshold to **50**

# *Binary images*

- Converting a grayscale image to a binary image by thresholding

$$g(x,y) \Rightarrow \begin{cases} 0 & if\ g(x,y) < T \\ 1 & if\ g(x,y) \geq T \end{cases}$$

Hint: use the function
**B = logical (A)**
to convert from uint8
to logical/binary

- How to choose a threshold?
  - Load the image "**coins.png**"
  - Create a new black & white (binary) version of it using the above conditions
    - Set the threshold to **50**
    - Now set the threshold to **200**

# *Binary images*

- Converting a grayscale image to a binary image by thresholding

$$g(x,y) \Rightarrow \begin{cases} 0 & \mathit{if}\ g(x,y) < T \\ 1 & \mathit{if}\ g(x,y) \geq T \end{cases}$$

Hint: use the function
**B = logical (A)**
to convert from uint8
to logical/binary

- How to choose a threshold?
  - Load the image "**coins.png**"
  - Create a new black & white (binary) version of it using the above conditions
    - Set the threshold to **50**
    - Now set the threshold to **200**
    - Probably if you choose something in the middle (such as **120**) you would get better results! Give it a try!

# *Binary images*

- Converting a grayscale image to a binary image by thresholding

$$g(x,y) \Rightarrow \begin{cases} 0 & \mathit{if}\ g(x,y) < T \\ 1 & \mathit{if}\ g(x,y) \geq T \end{cases}$$

Hint: use the function
**B = logical (A)**
to convert from uint8
to logical/binary

- How to choose a threshold?
  - Load the image "**coins.png**"
  - Create a new black & white (binary) version of it using the above conditions
    - Set the threshold to **50**
    - Now set the threshold to **200**
    - Probably if you choose something in the middle (such as **120**) you would get better results! Give it a try!
    - Plot the histogram of the image. What is the most suitable range for thresholding?

# *Otsu's method for thresholding*

- `imgBW = im2bw (img, level)`
- Converts grayscale image to a binary image. `level` is set automatically to 0.5. `level` is a normalized intensity value between [0,1].

# *Otsu's method for thresholding*

- `imgBW = im2bw (img, level)`
- Converts grayscale image to a binary image. `level` is set automatically to 0.5. `level` is a normalized intensity value between [0,1].
- `imgOtsu = imbinarize(img)`
- By default uses Otsu's method to binarize (threshold value minimizes the intraclass variance of the thresholded black and white pixels).

# Otsu's method for thresholding

- `imgBW = im2bw (img, level)`
- Converts grayscale image to a binary image. `level` is set automatically to 0.5. `level` is a normalized intensity value between [0,1].
- `imgOtsu = imbinarize(img)`
- By default uses Otsu's method to binarize (threshold value minimizes the intraclass variance of the thresholded black and white pixels).

Try it with "coins.png"

# Assignment #3

**Task 1**

- Appearance-based classification
- Download the CIFAR-10 dataset (MATLAB version)
  - https://www.cs.toronto.edu/~kriz/cifar.html
- Convert all images to grayscale
- We will work only on the **first batch** and the **test batch,** and on only **three** class, namely “**automobiles**”, “**deer**”, and “**ships**”
- Training:
  - Extract the first 30 images from each class from the first batch.
  - Get their histograms and store it

# Assignment #3

- Testing:
  - Test using the first 10 images of each class in the test batch
  - Get the histogram of a certain test image, and compare it to all stored histograms from the training phase using the Euclidean distance measure
  - Classify this image according to the nearest distance measure.
  - Calculate the accuracy = number of correctly classified samples / total number of test samples

## Task 2

- Consider the images of the Latin letter “**T**” *(found on the next slide)*
- Use the Euclidean distance, measure the distances between the histograms of the query image and each of the training images.
- Specify the closest match to the query image.
- Use 4-bin histograms (and plot them too!)

# Assignment #3

| | | | |
|---|---|---|---|
| 1 | 2 | 2 | 0 |
| 0 | 1 | 0 | 0 |
| 0 | 3 | 0 | 0 |
| 0 | 3 | 0 | 0 |

Unknown query input image (*I*)

| | | | |
|---|---|---|---|
| 0 | 2 | 2 | 2 |
| 0 | 0 | 2 | 0 |
| 0 | 0 | 3 | 0 |
| 0 | 0 | 2 | 0 |

Training image ($T_1$)

| | | | |
|---|---|---|---|
| 3 | 1 | 2 | 1 |
| 0 | 1 | 0 | 0 |
| 0 | 1 | 0 | 0 |
| 0 | 1 | 0 | 0 |

Training image ($T_2$)

| | | | |
|---|---|---|---|
| 1 | 2 | 2 | 0 |
| 0 | 1 | 0 | 0 |
| 0 | 3 | 0 | 0 |
| 0 | 0 | 0 | 0 |

Training image ($T_3$)