Homework 0

Introduction to Signal and Image Processing

Handout: February 19th, 2020
Handin: March 4th, 2020, 15:15

Instructions

The aim of the exercises in this homework is to gain experience with Python. Your hand-in will consist of a `.zip` archive named `hw0_firstName_lastName.zip` containing the following:

- Python source files named `hw0_exY_firstName_lastName.py`, where Y is the exercise number.
- All necessary files to run the above.

Your archive must be uploaded on ILIAS before the deadline. To give you feedback, you will be provided with the solutions after the deadline.

Code

For this homework, no code templates are provided.

Report

For this homework, you don't have to submit a report - rating is performed according to the submitted code only.

Exercises [5 points]

0.1 Exercise 1 [0.5 points]

You are given the following function:

$$f(x) = \cos(x \cdot e^{\frac{-x}{100}}) + \sqrt{x} + 1.$$

Create a function `ex_0(a,b,c)` that plots $f(x)$, where x are c equally spaced numbers in the range of $[a, b]$. If a and b are non-negative numbers the function returns 0 and plots $f(x)$, otherwise it returns -1 and does not plot anything.

0.2 Exercise 2 [1.5 points]

- Write a function `create_image` that generates a color image with the height/width dimensions of $n \times m$ with uniform randomly distributed red and black pixels. Add a single green pixel at a random location.
- Next, write a function `find_pixels` that finds the indexes of pixels with the values `pixel_values` in an image `img`.
- Using the image `img`, compute the euclidean distance of each red pixel from the green pixel without the use of any for loops.
- Display the computed distance vector `dist` in a histogram (with 100 bins), compute the mean, standard deviation and the median. Display the values as a title above the plot.
- Hint: Your functions should be callable in this manner:

```
img = create_image(n,m)
dist = compute_distances(img)
visualize_results(dist)
```

0.3 Exercise 3 [1 point]

Using the image `stopturnsigns.jpg` from ILIAS, try to find the threshold values t_{min}, t_{max}, such that a mask m contains only pixels of the stop sign. An example of such a boolean binary mask can be seen below.

0.4 Exercise 4 [2 points]

Considering two one-dimensional arrays, calculate the root mean square error between them following the steps below. Check your function for arrays of known difference.

- Define two arrays x, y of length 100, and assign random values to them.
- Write a function `rmse(a,b)` that calculates the RMSE (root mean square error) between two 1-d arrays of size N. The RMSE between two arrays x and y is defined as $\text{RMSE}(\mathbf{x}, \mathbf{y}) = \sqrt{\frac{1}{N}\sum_{i=0}^{N-1}(x_i - y_i)^2}$. Can you think of a way to calculate the RMSE without using any for loop (hint: check numpy's dot product function)?
- Calculate the RMSE between arrays x and y.
- Check your function through the degenerate case of the RMSE between array x and itself.
- Define an array with an offset of 2 from the values of x. What do you expect the RMSE to be? Confirm by using your function.
- Assume you have the function of exercise 0.1 ($f(x) = \cos(x \cdot e^{\frac{-x}{100}}) + \sqrt{x} + 1$), for $x \in [0, 100]$ and you want to approximate it with the line $g(x) = a \cdot x$, where $a = 0.3$. Plot the two lines on the same figure, and visually check if this is a good approximation.
- Calculate the RMSE between f(x) and g(x) for the given range.
- Try to tune the parameter a of the function g(x) so that you achieve a lower error (hint: try different values of a in an interval that makes sense for you, e.g. $a \in [0, 0.5]$). Plot the RMSE for the different values of a and use the numpy.min() function to choose the optimal a with respect to the RMSE. Print the minimum RMSE value, and the value of a for which it was achieved.