Assignment 2

Basic Image Processing Algorithms
Fall 2024

General rules

This is the last assignment of the semester.

The assignment will be graded on a **30 point** scale.
The contribution is **50% to the assignment grade**.

Deadline: **November 27**, 2024 23:59:59 (grace period ends on the **29th**)

The exercise is unguided. You are expected to work on the assignment alone and come up with a solution based on the labs and materials in the course.

The main task is to provide a good, reasonable solution. Minimal restrictions do apply on the naming on filenames, but feel free to code “freely”.

Problem formulation

You have to create a custom image recovery and texture based segmentation program that can remove artefacts from drone camera footage based on additional sensor data. The goal is to segment the images into semantic classes like grass, trees, etc.

Input:	a distorted drone camera image and measurements from the IMU
Output:	a segmented map of the captured image

Task outline

Write an algorithm that can process each captured image as follows:

1. loads the image and the assumed motion blur kernel (from the IMU)
2. uses an iterative deconvolution method (Richardson-Lucy)
3. applies local contrast enhancement (Wallis filter)
4. identifies the regions using a texture matching algorithm (Laws filter)
5. filters/enhances the result based on majority voting
6. returns a segmented image showing the clustered region map
7. count blobs of specific object

Key results to be presented:

You may code freely, as there are not so many restrictions on what to use.

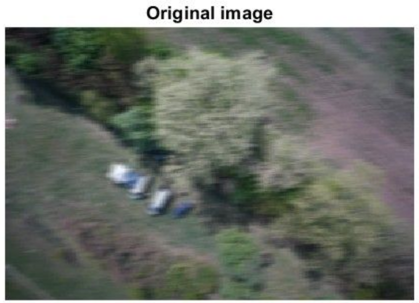
However, you should create a script which solves the exercise and

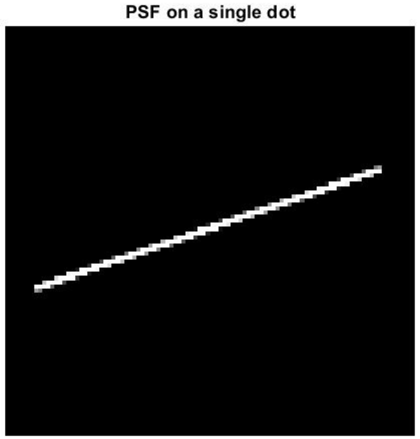
presents the following outputs:

Figure 1	shows the degraded (original) and restored image
Figure 2	shows the result of the Wallis filtering
Figure 3	shows the segmented image
Figure 4	highlights the blobs of selected objects on the segmented images with bounding boxes

Console output that shows the location (x,y) of each instance of the found object.

Figure 1


Original image


PSF on a single dot

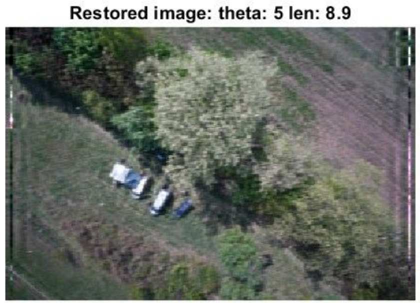
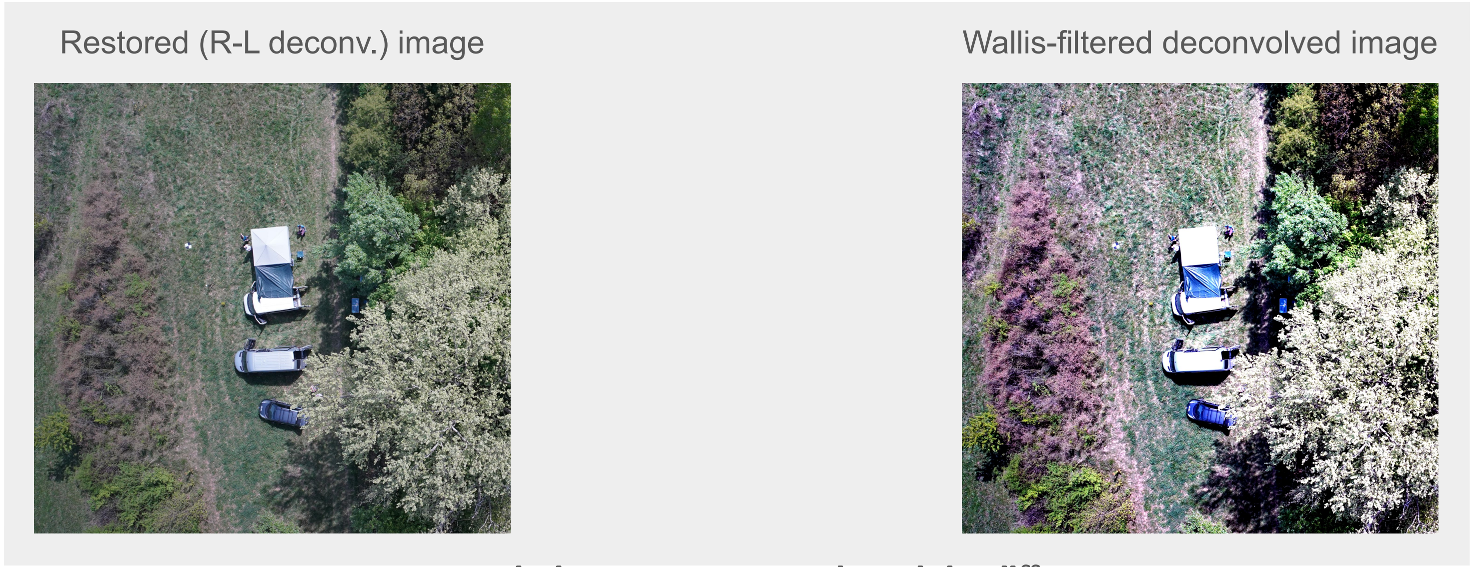

Restored image: theta: 5 len: 8.9

Figure 2


Restored (R-L deconv.) image
Wallis-filtered deconvolved image

example image, your results might differ

Figure 3

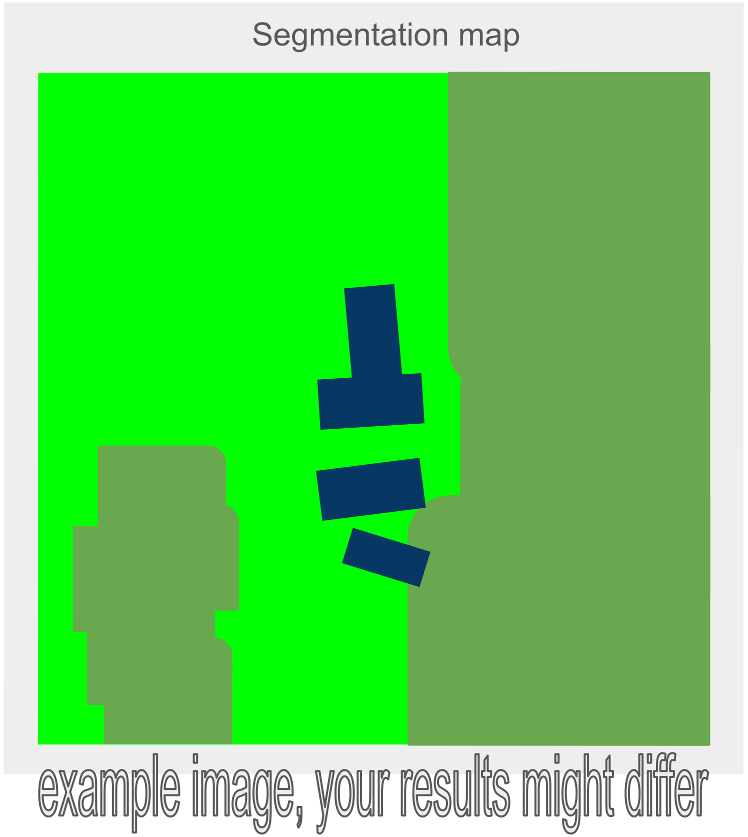
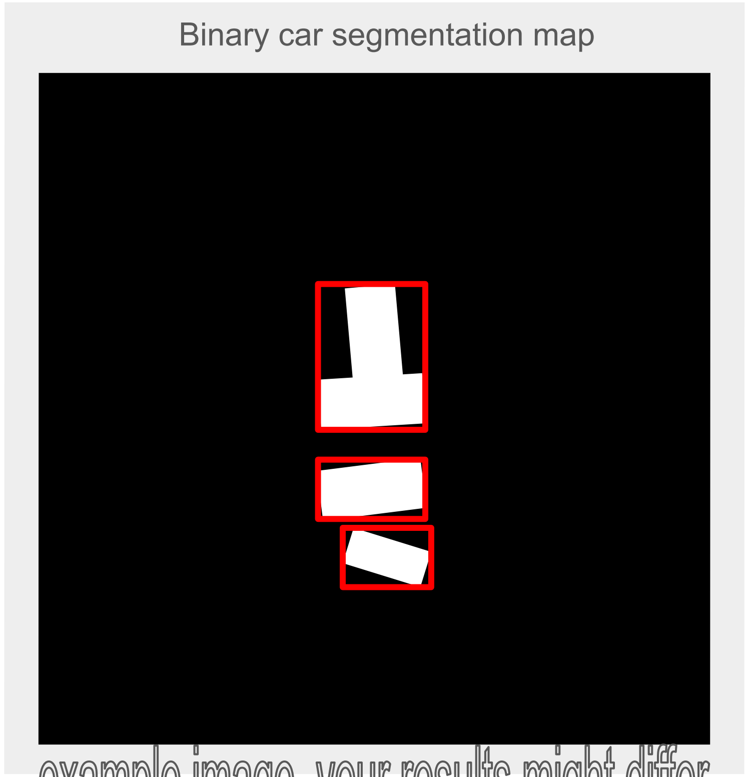


Figure 4



There is no code package for this assignment.

All scripts and functions must be written by you.
Collaboration and using code from previous years
is **STRICTLY** prohibited.

Download the images to be processed from moodle.

Submission & Hints

You should create a script named a03_NEPTUN.m where the NEPTUN part is your Neptun ID. This has to be the main script; running that must be able to solve the problem.

You are allowed to create other files (e.g. additional functions) too, if necessary. Usage of any built-in high level restoration functions (e.g. wallis, deconvlucy etc.) is **prohibited**.

Please submit the source files only (**WITHOUT THE TEST IMAGES**) in a **ZIP** file via the Moodle system.

Check the upcoming slides for hints!

Hints contd. (1)

1. For each image an IMU measurement is provided in the form of a len and theta measurement.
    a. due to the movement of the drone you can not assume that the kernel is the same for all images.
    b. the measurements are stored in a struct for each capture. The structs are stored in a cell. ;)

2. The Richardson-Lucy reconstruction method is described in details in multiple articles [1,2]. The key idea is to apply the following iteration rule:

$$\hat{u}^{(t+1)} = \hat{u}^{(t)} \cdot \left(\frac{d}{\hat{u}^{(t)} \otimes P} \otimes P^* \right)$$

    a. where d is the degraded image and P is the PSF of the imaging system.
    b. we recommend you the following article: https://en.wikipedia.org/wiki/Richardson%E2%80%93Lucy_deconvolution
    c. we recommend that you scale down the images **before** restoration (also scale the len parameter of the IMU measurement)

Hints contd. (2)

3. There should be a stopping criteria for the iteration. It can be hard-coded or it can be based on some measures (e.g. the maximum pixel intensity in the result of the Laplacian-of-Gaussian filtering is above a threshold).

4. The R-L deconvolution uses convolution. Please DO NOT use convolution in the spatial domain (which is a slow operation). Transform everything to the frequency domain (DFT) where the convolution is just a multiplication.

5. The Wallis filter [3] was described in detail at the beginning of Lecture 4. We recommend that you experiment with the parameters of the filter, some suggestions for starting values are:
   - Desired local mean: ½ of the top value of the dynamic range
   - Desired local contrast: ⅕ of the top value of the dynamic range
   - Amax: from the [1, 5] interval
   - p: something small (e.g. 0.2)

Hints contd. (3)

6. Texture samples of the Laws filtering should be fabricated by hand using small samples from each region of the input (restored, filtered) image.

7. Since the images are color, one can implement an RGB Laws filter which incorporates color info in the decision making. You can implement this by using 3D texture samples or by applying Laws filter on each color layer separately.

8. We recommend you to downscale the image when using laws filters.
    a. If you did this beforehand, it is not necessary

9. You can use multiple samples from each texture.

Hints contd. (4)

10. The regions on the segmented image should be represented by their corresponding cluster index (1, 2, 3).
    Use the built-in function imagesc to visualize the result.
11. For counting the object instances you should remove the background and create a binary mask then apply regionprops.

Grading rubric

The script filename is correct, it’s a script	2 pts
Script loads the image	1 pt
Figure 1 exists and looks good	2 pt
R-L deconvolution idea is understood, formula is OK	2 pts
R-L deconvolution stopping criteria is OK	1 pt
R-L deconvolution parameters are appropriate	1 pt
Figure 2 exists and looks good	2 pts

Grading rubric contd.

Wallis filter local average and contrast calculation is OK	2 pts
Wallis operator implementation is OK	2 pts
Wallis parameters are appropriate	2 pts
Figure 3 exists and looks good	2 pts
Laws kernels are OK	1 pt
Laws sample textures are OK	1 pt
Laws implementation is working, result is acceptable	2 pts
Some kind of voting / post-processing is applied and makes sense	2 pts
The cars have their own binary mask, fig 4 is ok	1 pt
Console output exists, shows the number of cars correctly and looks good	1 pt
Images with missing textures (e.g. no cars) are processed correctly	1 pt
Code quality (readability, understandability, good comments and structure)	2 pts

References

[1] Richardson, William Hadley (1972). "Bayesian-Based Iterative Method of Image Restoration". JOSA. 62 (1): 55–59. doi:10.1364/JOSA.62.000055

[2] Lucy, L. B. (1974). "An iterative technique for the rectification of observed distributions". Astronomical Journal. 79 (6): 745–754. doi:10.1086/111605

[3] Wallis, R., (1976). "An approach to the space variant restoration and enhancement of images". In: Proc. IEEE Conference on Computer Vision and Pattern, Naval Postgraduate School, Monterey, CA, USA