Georg-August-Universität Göttingen Sommersemester 2023

Lecture Series in Physics for Data Scientists

Timo Betz

Problem Set – Image Analysis in Biophysics

Due 22.06.2023 (23:59)

Solve at home, submit code the evening before tutorial, discuss in tutorial 23.06.2023

Problem 1: Determine the deformation fields in an image series. (5 Points)

Generate the deformation field of the image sets provided on the StudIP server.

- Download the zip-File: TFM_Example.zip
- In the unzip folder you will find the folders: “beads_2D”, “cells_2D”, “3D”.
- In the beads_2D folder you will find a file called reference.tif, in which the reference bead position (without cells) is stored. The images of the beads with the cells on are called beadsX.tif, where X marks the time series going from 0 to 6. Difference between images is 15 minutes, but you do not need this information. In the cells_2D folder you find the images of the cells. For your info the cells are endothelial cells (HUVECs) making normally up the inner wall of blood vessels. They are fluorescently marked to express Lifeact-GFP, which allows to see the actin cytoskeleton.
- For your Info. The cells are growing on an elastic substrate that contains the labeled beads. When the cells pull on the substrate, they can deform it. In the first exercise, your task is to determine the deformation field of the beads. This is like the vector fields generated by PIV, optical Flow and Elastix as used in the course.

Task: Use the three approaches PIV (openpiv), optical flow (opencv) or the elastix(itp) discussed in class to determine the displacement fields. You should generate the displacement with respect to the reference. One pixel corresponds to 0.1 µm in reality. (µm per pixel = 0.1)

- You should try to save the displacement fields as ux and uy (displacement in x and y direction).
- Tips: The demo settings of each library already not far from what will work. If it does not look nice, try playing with the parameters. Important is always the window size!
- Question: Which approach did you like best, and why?

Bonus (not required for credits, but cool to do): (2 extra Points)

- Load the 3D data of the beads.
- Which of the approaches can be extended to 3D image stacks?
- Compare the x,y and z displacement with the 2D version.

OpenPIV: http://www.openpiv.net/openpiv-python/

OpenCV: https://pypi.org/project/opencv-python/

Elastix: https://github.com/InsightSoftwareConsortium/ITKElastix

Problem 2: Calculate the traction forces that led to the deformation measured in Problem 1 (5 Points)

- For the inversion, you need to generate the Fourier Transformation of the displacement in x and y.
- Use a 2D fft implementation (in Python it is part of numpy or scipy, in Matlab directly available), to Fourier Transform ux and uy. (use the fftshift function. What does it do? Why should you use it?)
- What are k_x, k_y, k ?
- To construct k_x, k_y you need a linear spaced array in the x and y direction respectively that is repeated along the transverse direction.
- k is then simply: $k = \sqrt{k_x^2 + k_y^2}$
- Simply apply the formula used in class to get the Fourier transforms of the force. The inverted Fourier Transformation of the Boussinesq solution I presented had a sign error (sorry), which was also in the original paper. If you are interested to check for this error, you can invert the matrix given in the paper I put in the zip file.

Please use the following inverted Boussinesq solution in k space:

$$\tilde{\vec{F}}(\vec{k}) = \tilde{K}^{-1} * \tilde{\vec{u}}(\vec{k})$$

$$\tilde{K}^{-1} = \frac{E}{2(1-\nu^2)k}\begin{bmatrix}(1-\nu)k^2 + \nu k_x^2 & \nu k_x k_y \\ \nu k_x k_y & (1-\nu)k^2 + \nu k_y^2\end{bmatrix}$$

where $\nu = 0.5$ is the Poisson Ratio and E = 10 000Pa is the stiffness of the substrate.

- To obtain the Forces, calculate the inverse Fourier Transform (using ifftshift), and only consider the real part.
- Finally plot the forces in x and y.
- Ideally also generate an image with the absolute forces as image intensity (use a nice colormap) and plot the directions of the forces over the image (use quiver).

This is the simplest approach. It should already give reasonable results. What can one do to improve this approach? Where had you had trouble in implementing?

What you need for next Friday (23.6.2023):

1) Try to solve as much as possible. Send me your code the evening before the lecture. You can use any programming language you wish.
2) In class, I will ask each of you to present at least part of the code. From my side, you can work together on the problems, but everybody needs to submit his/her own code. You need to be able to explain the code.
3) You are allowed to use any information you can find. There are many pages and projects implementing this. You still need to be able to explain the code, and you should have played with it to understand what you are doing.
4) Ideally you have answers to the additional questions I posed.
5) To pass, you need at least 5 points. (I will also give points if you show that you tried but failed!)