Philip Morris: Data Analysis Improvement of Ciliary Beating of 3D Epithelial Tissue

Final report

Simon Jenni & Laurent Hayez

June 21, 2016

Contact:	Simon Jenni	– simujenni@students.unibe.ch
	Laurent Hayez:	– laurent.hayez@unine.ch
Supervisor:	Patrice Leroy:	– Patrice.Leroy@pmi.com

Table of contents

1 Project Description

1.1 Project Context

Philip Morris International (PMI) is a global cigarette and tobacco company with headquarters in Lausanne. The research and development program of PMI focuses on the development of products with the potential to reduce the risk of tobacco related diseases. To this end, new products are tested against ordinary cigarettes by exposing human tissue cultures to smoke or aerosol of both products. The effect of the exposure is then analysed by observing different features of the tissue, one of which is the ciliary beating.

1.2 Goals and Objectives of the Project

The goal of this project is to implement a tool for the automatic analysis of ciliary beating in tissue movies. Concretely the objectives are the following:

- Allowing batch processing of video-data contained in a folder (including subfolders).
- Pre-processing of the video data in order to remove noise by smoothing with a customisable 3D kernel.
- Scoring the tissue surface activity using simple descriptive statistics and storing the results in an activity image.
- Determining the frequency distribution given per region of interest (ROI) and extracting the dominant frequency.
- Processing should be possible on multiple scales, i.e. ROI of variable size.
- Illustrating the phase of the beating frequency.

Part of the project is also an evaluation of the performance of different techniques applied to the problem and other research objectives such as:

- Evaluating the effect of the ROI size on performance.

- Evaluating the probable shape of the beating pattern on a by beating movie basis.
- Comparing different techniques for the frequency analysis (e.g. FFT and wavelet transform).

Given the scope of this project and the relatively large amount of objectives, it should be noted that some of the objectives have been given a higher priority than others. The ultimate goal of this project is to provide a tool for frequency analysis and task-priorities were therefore weighted with this goal in mind. This means for example that de-noising, a whole subject on it's own, has not be studied and evaluated as extensively as techniques for frequency analysis.

2 Methodology

The implementation of the tool was carried out in Matlab. The decision to use Matlab has been taken in agreement with the client and is based on the ease of handling image and video processing and the relatively fast development time that Matlab provides. Git has been used as version control tool.
Development has been done in an incremental and iterative fashion roughly following the SCRUM framework. The objectives have been distributed across sprints of two weeks each. To keep track of the progress and help manage the project we have been using Taiga, an open source project management platform similar to JIRA. Both the client and project stakeholders were given access to Taiga and have been able to follow the progress.
Testing of our implementation using synthetic test-data has been an integral part of the development process to ensure correctness of the implementation. In order to maintain a high code-quality, code reviews by the other respective developer have been performed for every task.

2.1 State of the art

The arguably most accurate method for analysing the ciliary beating frequency is the direct measurement from high-speed video recordings. This is of course very time-consuming and therefore several automated methods have been proposed. The most commonly used approaches for the automated analysis of ciliary beating are based on using the Fast Fourier Transform (FFT) to analyse intensity-signals in a region of interest and has been the principal approach and starting point for further exploration used in this project. Other methods such as photomultiplier and modified photodiode techniques rely on different hardware and inputs and are therefore not considered.

3 Realisation

3.1 Frequency Analysis

Listing 1: Function performing the frequency analysis using the FFT.

```
function [ power, f, domFreqs, domPhase ] = performFFT( data, fs )
%PERFORMFFT performs frequency analysis using the fast fourier transform
% Input:
%   data - 3D array of extracted video data (width x height x frames)
%   fs - Sampling frequency of the input data
% Output:
%   power - Power of resulting discrete fourier transform
%   f - Frequency range
%   domFreqs - Dominant frequency per region of interest
%   domPhase - Phase corresponding to dominant frequency
% See also BATCHANALYSEFOLDER, WTANALYSIS.
```

3.2 Activity Analysis

Listing 2: Function performing the activity analysis using entropy.

```
function entropy = computeEntropy(data)
%COMPUTEENTROPY Computes pixel-wise entropy of the provided video-data.
% Input:
%   data - 3D array of extracted video data (width x height x frames)
% Output:
%   entropy - 2D array of size (width x height) where entropy(i,j) is the
%             entropy along data(i,j,:)
% See also PLOTRESULTS, ACTIVITYFROMPOWER.
```

3.3 Smoothing for Noise-Removal

Listing 3: Function performing the noise removal.

```
function [ data ] = filter3d( data, dims, sigmas, type )
%FILTER3D Filters the provided 3D-array data with a 3D-gaussian kernel with
%dimensions specified in dims and gaussians with standard-deviations
%specified by sigmas. Filtering in 'time' can optionally be performed using
%median-filtering.
% Input:
%   data - 3D array (width x height x frames)
%   dims - Vector (hx, hy, hz) of filter dimensions
%   sigmas - Vector (sx, sy, sz) parameters for gaussian filters
%   type - 'gaussian' or 'median' defining type of 3rd dimension filter
% Output:
%   data - filtered input
% See also BATCHANALYSEFOLDER.
```

3.4 Extracting Probable Beating Pattern

Listing 4: Function for extracting the probable beating pattern.

```
function [ shape ] = probableShapeFromData( dominantFreqs, phase, data,...
    activity, fs )
%PROBABLESHAPEFROMDATA Computes the probable shape of the beating pattern
%from the results of the FFT and the underlying data.
%Note: The FFT should be computed using ROI-size=1 for use with this
%function.
% Input:
%   dominantFreqs - 2D array of size (w x h) with dominant frequencies per
%                   pixel
%   phase - 2D array of corresponding phases per pixel
%   data - 3D array of extracted video data of size (w x h x t)
%   activity - 2D array of activities phases per pixel
%   fs - Sampling frequency of the input data
% Output:
%   shape - Curve fitted to the data using smoothing-splines
% See also PERFORMFFT, COMPUTEENTROPY, ACTIVITYFROMPOWER.
```

3.5 Batch Processing of the Data

Listing 5: Function for extracting data from videos.

```
function batchExtractFolder( extractFolder, storeFolder, reload )
%BATCHEXTRACTFOLDER Extracts data of all the videos in the folder
%extractFolder (including subfolders) and saves it to storeFolder for later
%analysis. Note that if reload is set to false and the video has already
%been extracted, the video will not be extracted again.
% Input:
%   folderPath - String indicating the path of the video-data
%   storeFolder - Folder where the extracted data should be stored
%   reload - Boolean value (already extracted videos skipped if false)
% See also EXTRACTVIDEODATA, SAVEEXTRACTEDDATA and ALREADYEXTRACTED.
```

Listing 6: Function for analysing extracted data.

```
function batchAnalyseFolder( folderPath, fs, roiSize, resultsDir,...
    preProcess, method, waveletType )
%BATCHANALYSEFOLDER Anlayses all files in the given folder using the
%specified parameters and saves the results to the folder resultsDir.
% Input:
%   folderPath - Path to folder of exctracted video files to be analysed
%   fs - Sampling frequency of the data to be analysed
%   roiSize - Integer or vector [sx, sy] specifying the ROI-size
%   resultsDir - Path to directory where results should be stored
%   preProcess - Function handle for pre-processing function of extracted
%                data (e.g. denoising)
%   method - String indicating which method to use for analysis ("wt",
%            "fft" or "both")
%   waveletType - Specifies which wavelet to use for the wavelet transform.
%                 'morl' is used by default
% See also BATCHEXTRACTFOLDER, FILTER3D.
```

3.6 GUI

4 Results

4.1 Effect of ROI size on performance

We measured the computational time of the FFT analysis and the WT analysis for different ROI sizes on the initial cylia beating movie. The ROI sizes tested were 1, $[3, 4]$ and $[6, 8]$. The results are plotted in Figure 1.
Interestingly, computing the FFT pixel wise takes approximately 2 minutes and 40 seconds, while it takes 25 minutes for the WT. However, as soon as we increase the ROI size, the computational time for the WT decreases drastically. It should be noted that the computations were run with `batchAnalyseFolder`, hence the times presented here

FFT	WT
164.517	1,525.203
27.674	146.551
24.087	55.623

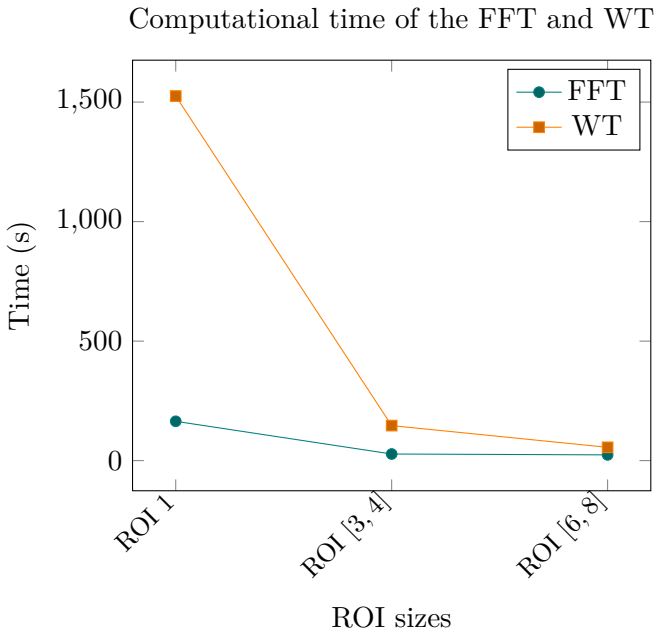

Figure 1: Computational time of the FFT and WT for different ROI sizes

are greater than if we had only used `performFFT` and `WTAnalysis` instead.

4.2 Effects of wavelet types

The code currently supports only two types of wavelet types (two of the functions used support almost exclusively distinct wavelet types):

- `morl` (Morlet wavelets),
- `mexh` (Mexican hat wavelets).

However, there are significant differences between the results we obtain with these two wavelets. You can compare Figure 2 with Figure 3. Although the spectrums are similar, the results we obtain with the Morlet wavelet are much closer to the results we obtain with the fast Fourier transform.

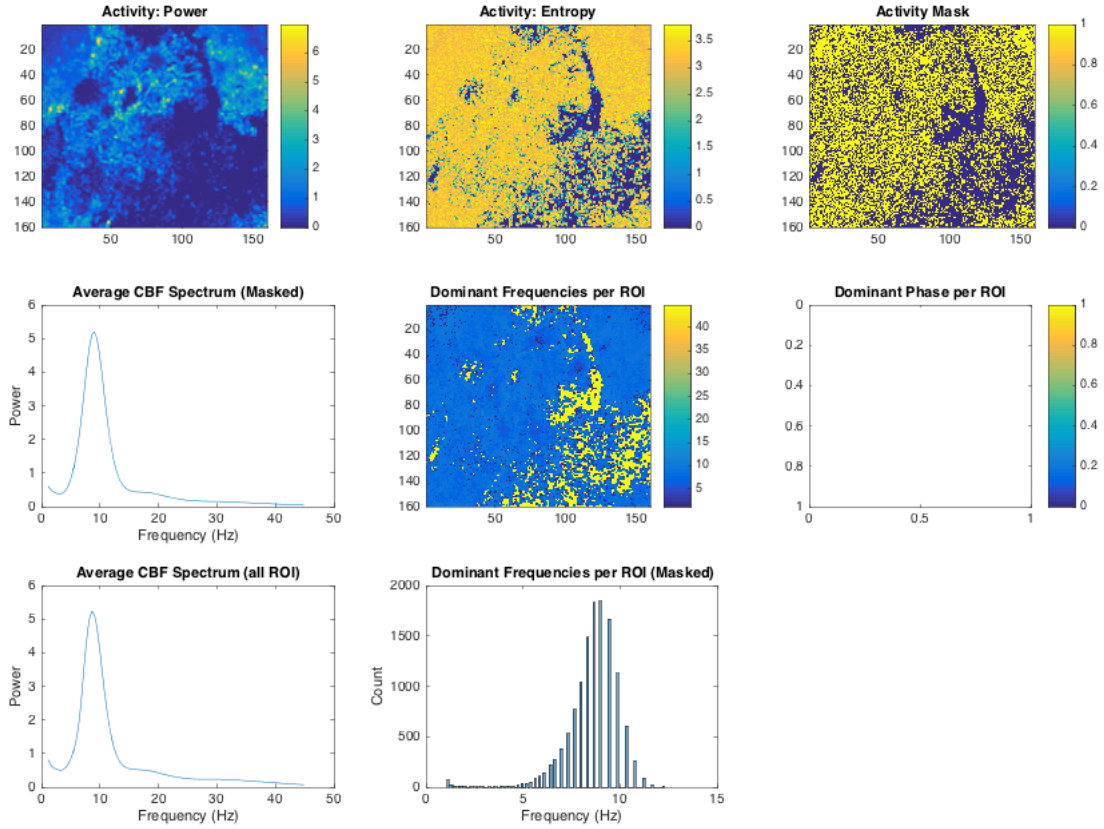

Figure 2: Results with `morl` wavelets

4.3 Comparison of the techniques used

5 Recommendations

5.1 Statement of Recommendations

5.2 Limitations

5.3 Outstanding Issues and Perspective for Future Work

6 [Other relevant section]

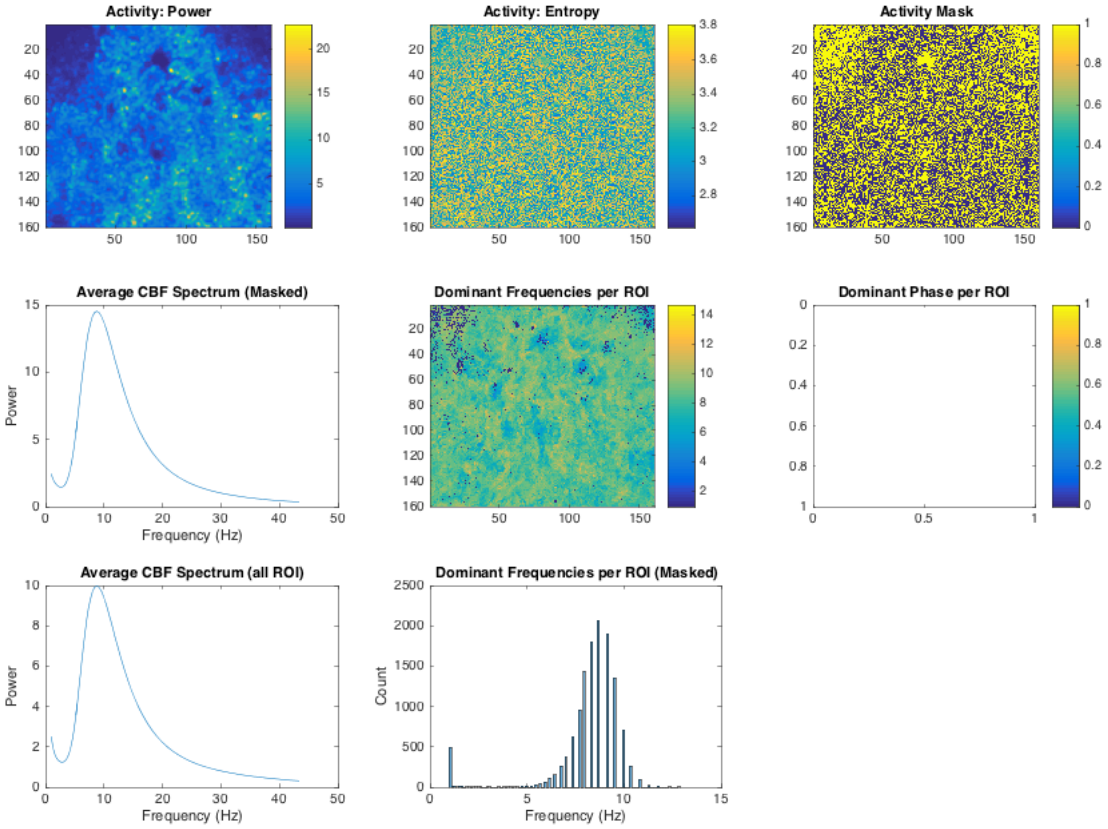

Figure 3: Results with `mexh` wavelets