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 **2**
- 1.1 Project Context . . . 2
- 1.2 Goals and Objectives of the Project . . . 2

2 Methodology **3**
- 2.1 State of the art . . . 3

3 Realisation **3**
- 3.1 Frequency Analysis . . . 4
- 3.2 Activity Analysis . . . 4
- 3.3 Smoothing for Noise-Removal . . . 5
- 3.4 Extracting Probable Beating Pattern . . . 5
- 3.5 Batch Processing of the Data . . . 6
- 3.6 GUI . . . 7

4 Evaluation **7**
4.1 Effect of ROI-Size on Performance 7
4.2 A Comparison of the Different Techniques 8

5 Recommendations **10**
5.1 Statement of Recommendations 10
5.2 Limitations 10
5.3 Outstanding Issues and Perspective for Future Work 10

6 [Other relevant section] **10**

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

In this section we give a brief overview of the main functionalities of the tool. The usage of these functions is demonstrated in the `demo.m` script.

3.1 Frequency Analysis

We implemented two techniques to perform the frequency analysis:

- Fast Fourier Transform (FFT): Implemented in the function `performFFT.m`
- Wavelet Transform (WT): Implemented in the function `WTAnalysis.m`

The interfaces of the two functions are identical with the exception that `WTAnalysis.m` has an additional optional parameter specifying the type of wavelet to use and `performFFT.m` has one more return value (the dominant phases per ROI). The header of `performFFT.m` is shown in Listing 1.

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

To compute the tissue activity we explored two possibilities. The computation based on the mean power of the FFT is done in `activityFromPower.m` and the computation based on ROI-wise entropy can be performed with `computeEntropy.m`. The arguments to these two functions are different (`activityFromPower.m` uses the output from `performFFT.m` while `computeEntropy.m` uses the video-data) but the return values are interchangeable. The header of `computeEntropy.m` is shown in Listing 2.

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

As a pre-processing step we can use smoothing for noise-removal. This functionality is implemented in `filter3d.m`. As the name suggests, the smoothing is performed via a 3D filtering of the video data with a customisable 3D smoothing-kernel. We can apply 3D filtering because we deal with videos taken from a static scene (except the cilia) with a fixed camera. The 3rd dimension of the filter-kernel (the *time*-dimension) can either be chosen as a gaussian- or median-filter. All other dimensions are fixed gaussians. The header of `filter3d.m` is shown in Listing 3.

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

To estimate the probable shape of the beating pattern we make use of the results from the frequency analysis using the FFT. We select the ROIs with the most common dominant frequencies and high activity to regress the beating shape. Before using locally weighted linear least squares to estimate the shape we use the phase estimates to shift the samples in time such that they all have the same phase. The function implementing this procedure is `probableShapeFromData.m` and the corresponding header is shown in Listing 4.

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

The processing of the video data is possible in batches. Processing of videos takes place in two phases, first the data is extracted from videos contained in a folder with `batchExtractFolder.m` for further processing. Data is extracted for all videos contained in the specified folder including subfolders. In the second phase, the extracted data is then analysed with `batchAnalyseFolder.m`. This function coordinates the frequency and activity analysis and takes care of storing the results. Both functions can run parallel (if the required Toolbox is installed) leading to significant speed-ups on multiprocessor systems. The header of `batchExtractFolder.m` is shown in Listing 5 and the header of `batchAnalyseFolder.m` in Listing 6.

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

For a more flexible and convenient interpretation of the results we provide a GUI. Note that it is only used for the inspection of the results and not for running the analysis. Figure 1 shows the layout of the GUI. Some of the special features of the GUI are the following:

- The available videos and ROIs for inspection are accessible via the two drop-down menus in the top-left.
- The techniques used for the frequency and activity analysis can be switched via the buttons on the left.
- Clicking on a ROI in the activity plot will show the corresponding frequency spectrum in the bottom centre plot.
- In the dropdown-menus above the bottom-left and bottom-right figures you can choose which statistics or plots these figures should display.
- The activity mask (top-right) can be manipulated via the two dropdown-menus above the figure. The mask influences some of the statistics, i.e. yellow ROI are included and blue ROI excluded.

4 Evaluation

4.1 Effect of ROI-Size on Performance

We compared the computation time required for the FFT and WT analysis for different ROI sizes on a single video. We tested with ROIs of size 1, $[3, 4]$ and $[6, 8]$ on a 2.4 GHz

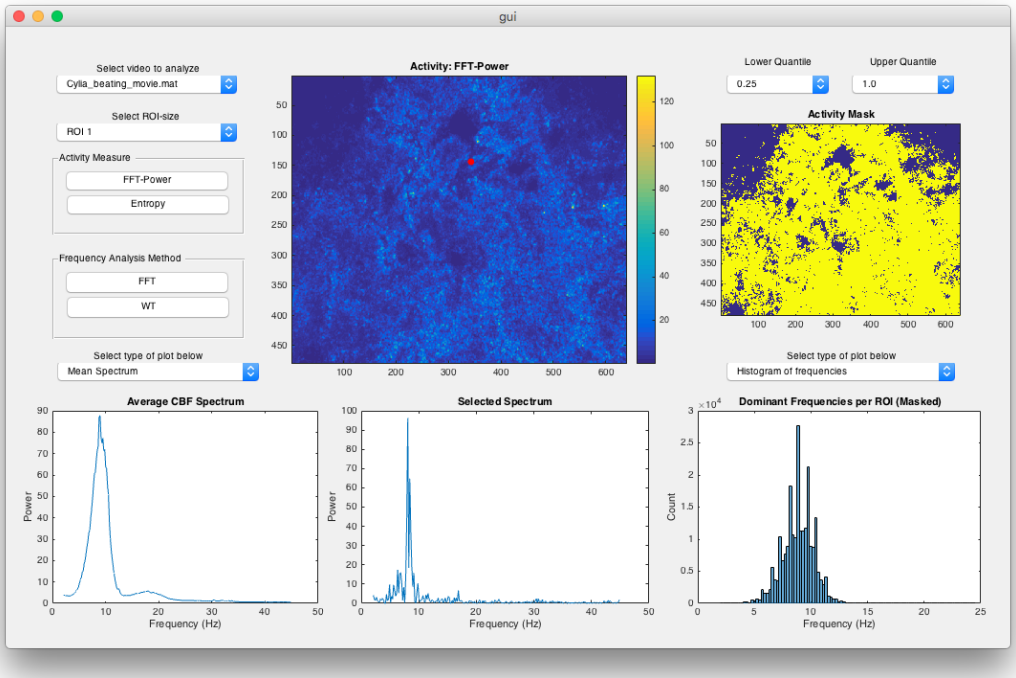

Figure 1: The GUI which can be used to inspect the results of the analysis.

Intel Core i5 MacBook Pro with 8 GB 1067 MHz DDR3 RAM. The results are plotted in Figure 2.
We observe that computing the FFT pixel-wise takes approximately 2 minutes and 40 seconds, while it takes around 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 are greater than if we had only used `performFFT` or `WTAnalysis`.

4.2 A Comparison of the Different Techniques

Frequency Analysis We compared results obtained with the FFT with results obtained with the WT on synthetic data to get an idea of the quality of the results. The synthetic example in this case is made up of high amplitude sinusoid at 10Hz and low amplitude sinusoids at 15Hz. Results are shown in Figure 3. We make the following observations:

a. The FFT is much more precise with respect to frequency analysis. We can clearly observe this in Figure 3.

b. The results of the WT are highly dependent on the type of wavelet. The Morlet for example is much closer to the FFT than the Mexican-Hat wavelet.

FFT	WT
164.517	1,525.203
27.674	146.551
24.087	55.623

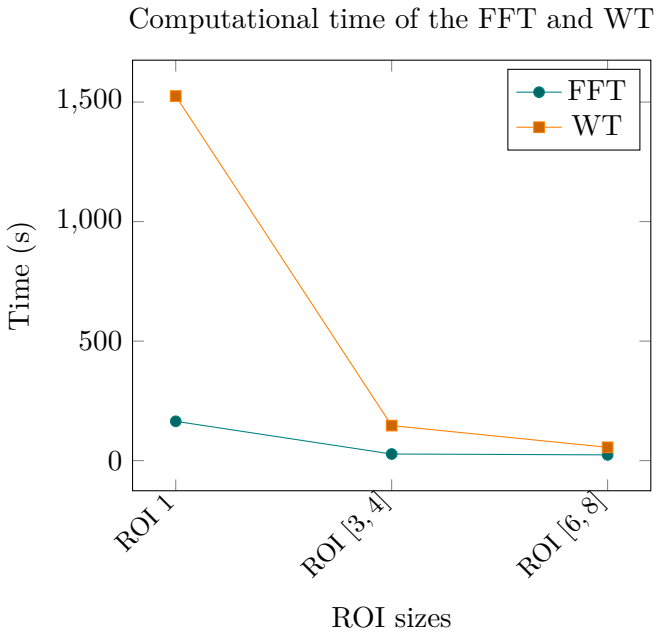

Figure 2: Computational time of the FFT and WT for different ROI sizes

c. FFT is much faster than WT especially at smaller ROI-sizes.

d. FFT provides us with phase information that we can use to compute the probable beating pattern.

e. WT allows time-frequency analysis.

f. WT is very customisable and so most likely the results with the WT can be improved through careful parameter tuning.

From these observations we can conclude that if we are not interested in a time-frequency analysis of the signal we get no benefit from using the WT.

Activity analysis: We implemented two methods to analyse the activity of the video: using the FFT power and entropy. If the FFT was previously computed, then computing the activity with the FFT is straightforward and very fast. The function computing the entropy has a $O(\texttt{heigth} \times \texttt{width} \times \texttt{time})$ complexity. The quality of the two techniques is quite different in that FFT power is much more sensible to the amplitude in the signal. If we're interested to only look at very strong beatings it is therefore more useful to look at the FFT-power. Conversely, if we are more interested in low amplitude signals the entropy seems like the better choice.

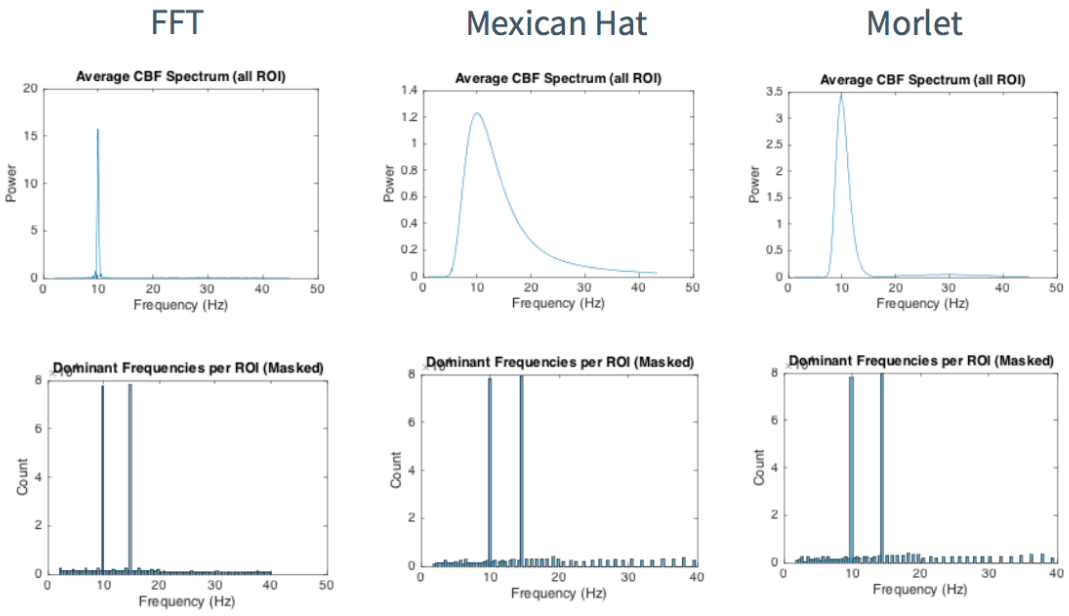

Figure 3: Comparison of results obtained with the FFT (left), WT with Mexican-Hat wavelet (middle) and WT with Morlet wavelet.

5 Recommendations

5.1 Statement of Recommendations

5.2 Limitations

5.3 Outstanding Issues and Perspective for Future Work

6 [Other relevant section]