This document aims to describe the Matlab code NewAnalysisCilia.m, used to measure motile cilia frequency and other properties. This code was used in (Thouvenin et al., eLife, 2020) and (Thouvenin et al., Bioprotocol, 2020).

A first description of the analysis workflow is provided, as well as a line by line walkthrough of the code.

Cilia analysis workflow (From (Thouvenin et al., BioProtocol, 2020)):

> *The program first loads the imaging data with cilia dynamics versus time (Figure 1A), and applies a local average filter (of size 4 by default) to increase the cilia SNR. For each pixel in the filtered data, the time series is extracted and Fourier transformed (Figure 3B). The 5 maximal peaks of the Fourier spectrum are extracted, but, by default, only the first one is used. The frequency of the other peaks can be used for validation (e.g., if sampling errors are made, the sum of the frequencies of the first and second peak is the acquisition frequency). A 2D image with the main frequency found at each pixel is thus created (Figure 3C). In noisy regions, it outputs a random frequency, but in cilia regions it draws regions of interests of a given frequency that we considered to be single cilium. Each of these regions of interest containing more than 40 pixels (7.5 μm²) is finally segmented and analyzed. The parameters frequency, diameter, eccentricity, area, angle, and major axis length are extracted and associated to their corresponding cilia parameters.*
>
> *If a comparison between dorsal and ventral cilia is of interest (Thouvenin et al., eLife, 2020), the program allows to manually draw a line at the center of the central canal and classify cilia as dorsal or ventral in respect of their relative position to the central line. This procedure is not described further here, but can be found in the manual associated with the code.*

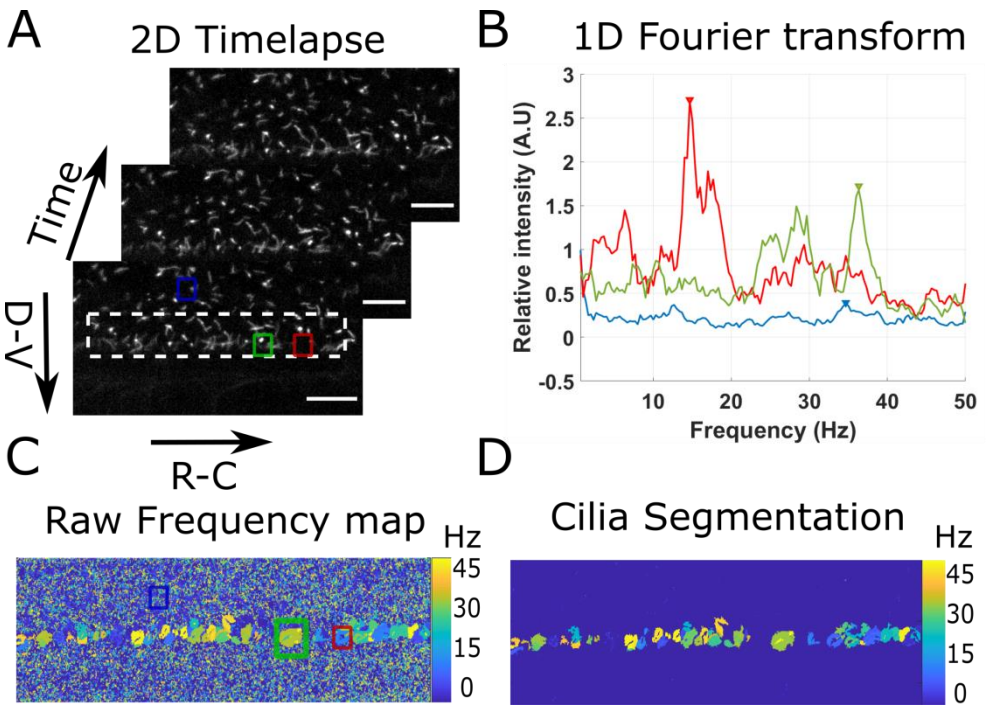


Figure 3. Principle of the beating cilia analysis. 2D time lapses of cilia beating (A) are analyzed by measuring the time Fourier transform at each pixel (B). 3 spectra corresponding to the pixel at the center of the three regions drawn in (A) are plotted. The frequency of the peak of maximum amplitude (arrows in (B)) is extracted for all spectra, to form a frequency map (C), showing regions of constant frequency corresponding to individual cilia. All cilia are then segmented by keeping only the largest regions of constant frequency (of area above 40 pixels) (D). Scale bar is 15 µm.

Program description :

In the software folder, there is only the main script, which sometimes calls functions that are stored in the SubFunctions folder. We will first describe the main script, and will provide detail description of the subfunctions below.

I. Main program : New Analysis.m

a) Initialisation and adjust parameters :

```
clear % Clear all Variables in workspace

% Add the program directory and subdirectory to memory
CurrentPath=pwd; %% Finds the script folder
addpath(genpath(CurrentPath)) % Adds to the path the folder and all subfolders - so
that all the functions in the SubFunction folder will be found.
Definition of parameters
```

```
%%%%%%%%%%%%%%%%%%%%Adjustable parameters%%%%%%%%%%%%%%
AcqFreq=100;%Hz Acquisition frequency of the cilia beating time lapse
DrawCentralLine=0; % Boolean related to the location of cilia. If 1, asks the user
to draw the central line to distinguish between ventral and dorsal cilia.
SzAvg=4; %Size of local spatial average to increase SNR

AllFreq=zeros(9,10); The AllFreq matrix will contain all the parameters (9) kept
for each cilia. The value 10 is here just to initialize the matrix, which size
increases as more and more cilia are segmented. A proper initialization is not
possible as we don't know in advance how many cilia will be detected.
Compt=0; A counter for the number of cilia.
```

b°) Define experiment folder: The idea is to store all multipage .tif files, each of which corresponding to one experiment (e.g. 300 images of cilia acquired at 100 Hz) , corresponding to a same condition (WT embryos). The code will analyse all files sequentially and will store cilia parameters in a single matrix/table, with all the data aggregated at the same place.

```
DataPath=uigetdir('','Choose Start Folder'); Finds the path of the folder with the
experimental data stored as multipage .tif files
cd(DataPath) Change the current folder to the folder DataPath (with experimental
data)

%Find and analyze every .tif file one after the other
list=ls('*.tif'); Finds all .tif files in the current folder (DataPath) and stores
them into the structure list.
```

c°) Initialize the matrix Central Line, with the coordinates of the central line for all experiments

```
if DrawCentralLine If DrawCentralLine value is changed to 1, the CentralLine matrix
will be initialized
    CentralLine=cell(1,size(list,1)); CentralLine is a cell type with a number of
cells equal to the number of experiments found in the folder. For each cell
(experiment), CentralLine will store the coordinates of the central line manually
drawn below.
end
```

The next line starts a for loop, which will repeat the rest of the procedure for each .tif file (Acquisition).

```
1.28 for nn=1:size(list,1)

 1.147 end
```

d°)Open .tif file and defines the name from the saved files: For each acquisition, the name of the .tif file is found. For each acquisition we save one png file with the central canal with the frequency map of the segmented cilia, and a .mat file that stores the cilia parameters.

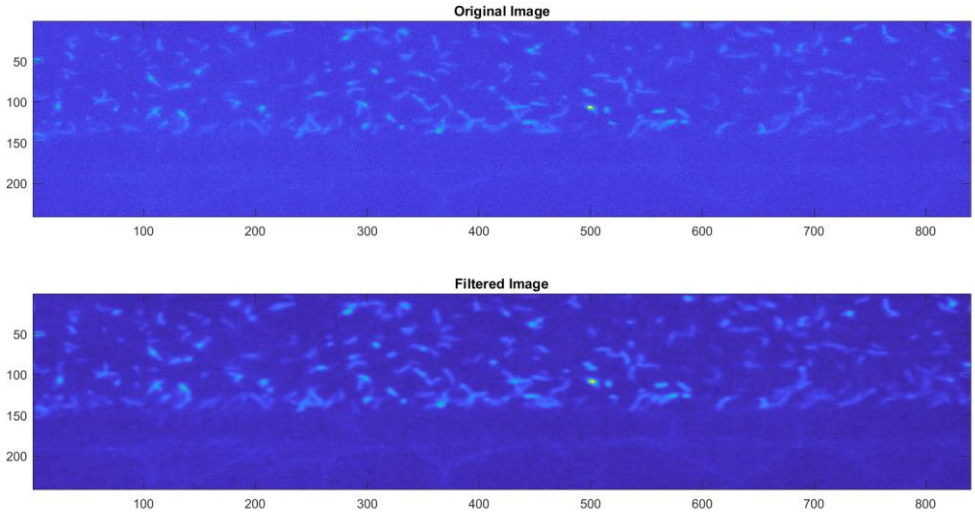
```
FileTif=strtrim(list(nn,:)); %e.g. WT4_1_Middle_100Hz.tif. Finds the name of the
nn-th .tif file. Strtrim deletes the blank spaces in the name (because the ls
function generates many blank spaces if all file names don't have the same length.

    SaveFigure=strcat(FileTif(1:end-4),'.png'); %Name for the png file where the
segmented cilia will be saved. From the name of the TifFile, removes the .tif
extension and changed it to .png
    SaveFreq=strcat(FileTif(1:end-4),'.mat'); %Name for the mat file where the
AllFreq matrix will be saved at the end. From the name of the TifFile, removes the
.tif extension and changed it to .mat


    %%Open .tif file
    [Ima,Size]=OpenTif(FileTif); Calls the subfuntion OpenTif that opens the .tif
file and stores it in the Matlab workspace as the matrix Ima of Size (Size.Nlines,
the height of the image, Size.NCol, the width of the image, and Size.NIma, the
number of images.)
```

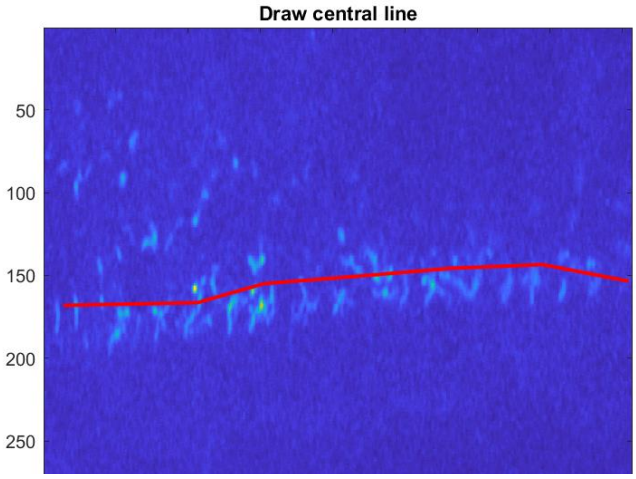
e°)Performs local average to increase SNR.

```
  %Local Average (Smooth)to increase the SNR
    FilteredIma=Local2DAverage(Ima,SzAvg); Calls the subfuntion Local2DAverage that
filters the matrix Ima with a spatial scale SzAvg
    imagesc(FilteredIma(:,:,ii))
```



Local average filter. For each pixel, the local average filter calculates the average value of the given pixel and its neighbours with a spatial scale of SzAvg defined at the beginning. If SzAvg is equal to 4, the code calculates the average value on a square centered on each pixel and of length 4 pixels, meaning the average of 16 pixels, reducing the random noise in the fluorescence images.

f°) Manual drawing of the central line : If the central line was asked (by changing the value of DrawCentralLine to 1 at the beginning of the code), the user is asked to draw a line (with as many changes of direction as required) at the center of the central canal to distinguish between ventral (below the line) and dorsal cilia (above the line).



User interface to draw the central line. The central line can be a succession of line segments to follow the central canal geometry.

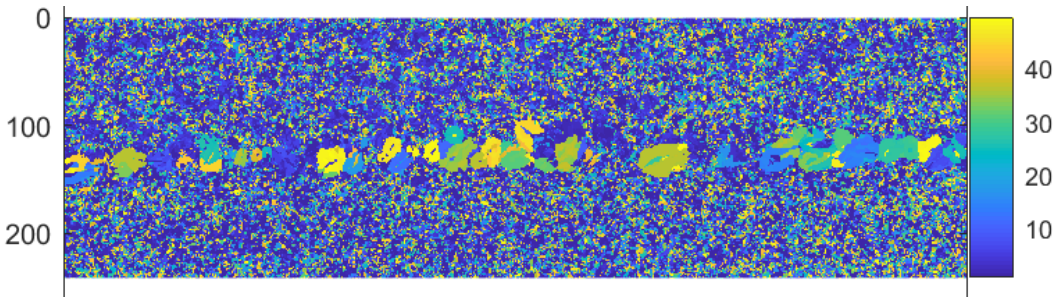
The line should be drawn from left to right in this code (or the code should be modified). To draw the line, first perform a left click at the first point (e.g. left of the image), and another left click for every line segment you need (if the central canal is not flat, the center line should adapt to the central canal geometry). The line is drawn with a black dashed line until the function getline ends, by pressing any key, a right click or a double click. In this case, the rest of the program draws the line in red on the figure. Before the function ends, it is delete, or return arrow.

```
if DrawCentralLine
        %Write a function
        imagesc(FilteredIma(:,:,1)) %Plots the 1st filtered image.
        title('Draw central line') %Changes the title
        [x, y] = getline; %Ask to draw the line manually
X, y are points coordinates the user selected manually. The rest of the code allows
to fit line segments between all selected points and will draw the corresponding
line in the figure.
        SeparationLine=zeros(2,1+ceil(x(end))-floor(x(1))); Initialize
SeparationLine. SeparationLine is the matrix that contains the coordinates of the
line segments
        SeparationLine(1,:)=(floor(x(1)):ceil(x(end))); %The abscissa of the line
segments are all the pixels starting from the abscissa of the 1st point to the
abscissa of the last point.
% Then calculates the  corresponding ordinates
        Compt2=0;
        for zz=1:(length(x)-1) %For each selected point
            Length=sum(SeparationLine(1,:)<x(zz+1)&SeparationLine(1,:)>x(zz)); %
Finds the length of the line segment between the point and the next one
            SeparationLine(2,1+Compt2:Compt2+Length)=(y(zz):(y(zz+1)-
y(zz))/(Length-1):y(zz+1)); Fits the ordinates along the length of the line segment
            Compt2=Compt2+Length;
        end
        SeparationLine(2,Compt2:end)=y(end); %Fill the ordinates corresponding to
the last selected point
        hold on
```

```
        plot(SeparationLine(1,:),SeparationLine(2,:),'r','linewidth',2) %Plots the
line in red in the figure with the first filtered image.
        pause(0.1)
        CentralLine{nn}=SeparationLine; %Save the coordinates of the line
    end
```

g°) Performs Fourier calculation and calculates the Frequency map (Detailed description below): For every pixel in the Filtered image, it calculates the Fourier transform and the power spectral density. For both spectra, the 5 peaks of maximal intensity are saved respectively in Freq and Freq_FromPSD.

```
[Freq,Freq_FromPSD]=CalculateFreqAndPSDMap(FilteredIma,Size,nn,AcqFreq);
```



h°) Cilia segmentation: To allow easier segmentation, the Frequency map is divided into 9 frequencies bands. Therefore, if two cilia are close to each other and difficult to segment, but beat at different frequencies, it will be easy to separate them. In this case, we performed the segmentation using only the first frequency component (from the Fourier spectrum).

```
FrequencyMap=Freq(:,:,1)*0;

for ff=1:9 % I divide the frequency into 9 bands of 5 Hz
        FreqBand=AcqFreq/20; %Frequency bandwidth : We divide by 2 to have the max.
frequency and by 10 to divide into 9 frequency bands (while leaving out the first
one)
        %Create the binary image
        Freq2=Freq(:,:,1); %Here we only used the first component of the Fourier
spectrum
        Freq2=Freq2>ff*FreqBand&Freq2<(ff+1)*FreqBand; We create a map with all
frequencies within a bandwidth. Freq2 is a binary map

        Freq2=bwlabel(Freq2, 8); %Finds connected pixels in the binary map, and
labels them
        Freq2=imfill(Freq2); %Fills the gaps if there are holes in the connected
components.

        %Find Regions of interest in the image
        blobMeasurements = regionprops(Freq2, Freq(:,:,1), 'all');  %Image
processing tool: Finds regions of connected pixels in the Frequency map. It creates
a structure with many parameters (Area, eccentricity, etc…) for each detected ROI.
        %Keep only regions with Area > 40 pixels
```

```
        EffectiveCilia=blobMeasurements([blobMeasurements.Area]>40); Only selects
ROIS containing more than 40 pixels. To exclude
        keeperIndexes = find([blobMeasurements.Area]>40);
        keeperBlobsImage = ismember(Freq2, keeperIndexes);
        FrequencyMap=FrequencyMap+keeperBlobsImage.*Freq(:,:,1);
        keeperBlobsImage= bwlabel(keeperBlobsImage, 8);
```

i°) Measure Cilia parameters:

```
        NAreas=size(EffectiveCilia,1);% Number of cilia
        %Position of cilia

PosCilia(1:2,Compt+1:Compt+NAreas)=reshape([EffectiveCilia.Centroid],2,NAreas);
        %Frequency
        AllFreq(1,Compt+1:Compt+NAreas)=[EffectiveCilia.MeanIntensity];
        %Diameter
        AllFreq(2,Compt+1:Compt+NAreas)=[EffectiveCilia.EquivDiameter];
        %Eccentricity
        AllFreq(4,Compt+1:Compt+NAreas)=[EffectiveCilia.Eccentricity];
        %Area
        AllFreq(5,Compt+1:Compt+NAreas)=[EffectiveCilia.Area];
        %Major Axis Length
        AllFreq(6,Compt+1:Compt+NAreas)=[EffectiveCilia.MajorAxisLength];
        %File number
        AllFreq(8,Compt+1:Compt+NAreas)=nn;


        if DrawCentralLine %If the central line is drawn, it saves the orientation
depending on the position of the cilia + Saves if dorsal or ventral + Saves if too
far from central canal (to be discarded)

            SeparationLine=CentralLine{nn};
            AngleLine=atan(diff(SeparationLine(2,:),1,2))*180/pi;

            CiliaAngle=[EffectiveCilia.Orientation];
            IsDorsal=reshape([EffectiveCilia.Centroid],2,NAreas);

            for zz=1:NAreas
                [~,Idx]=min(abs(round(SeparationLine(1,:)-IsDorsal(1,zz))));
                AllFreq(3,Compt+zz)=CiliaAngle(zz)+AngleLine(max(1,Idx-1));
                AllFreq(3,Compt+zz)=90-AllFreq(3,Compt+zz);% Je calcule l'angle par
rapport à la verticale
                AllFreq(7,Compt+zz)=(IsDorsal(2,zz)>SeparationLine(2,Idx)); %1 if
ventral 0 if dorsal
                AllFreq(9,Compt+zz)=abs(IsDorsal(2,zz)-SeparationLine(2,Idx))<35;%1
if not too far (6.5um away) from central canal cetral line
            end
        else
            AllFreq(3,Compt+1:Compt+NAreas) =[EffectiveCilia.Orientation];
            AllFreq(7,Compt+1:Compt+NAreas)=-1*ones(NAreas,1);
            AllFreq(9,Compt+1:Compt+NAreas)=ones(NAreas,1);
```

```
        end

        Compt=Compt+NAreas;
    end
```

j°) Save the data and cilia frequency map and ends program.

```
  imagesc(FrequencyMap,[5 AcqFreq/2])

    colorbar

    axis equal

    set(gca,'FontSize',24)
    saveas(gcf,SaveFigure)
    %Save 2D map
    save(SaveFreq,'Freq','Freq_FromPSD','AllFreq','FrequencyMap')

end
```

II. Subfunctions:

a. OpenTif : Takes the name of the .tif file as input and opens the file into the matrix Ima. The structure Size gives the size of the .tif file.

```
function [Ima,SizeIma]=OpenTif(FileTif)
Infos=imfinfo(FileTif);
SizeIma.NCol=Infos(1).Width;
SizeIma.NLines=Infos(1).Height;
SizeIma.NImages=numel(Infos);
Ima=zeros(SizeIma.NLines,SizeIma.NCol,SizeIma.NImages);

for ii=1:SizeIma.NImages
    Ima(:,:,ii) = double(imread(FileTif,'Index',ii,'Info', Infos));
end
end
```

b. LocalAverage2D

```
function FilteredIma=Local2DAverage(Ima,BinValue)
FilteredIma=Ima*0;
for ii=0:BinValue-1
    for jj=0:BinValue-1
        FilteredIma=FilteredIma+circshift(circshift(Ima,ii,1),jj,2);
```

```
    end
end
FilteredIma=FilteredIma/(BinValue^2);

end
```

c. CalculateFreqAndPSDMap

```
function [Freq,Freq_FromPSD]=CalculateFreqAndPSDMap(Ima,Sz,nn,AcqFreq)

Freq=zeros(Sz.NLines,Sz.NCol,5);
Freq_FromPSD=zeros(Sz.NLines,Sz.NCol,5);
h=waitbar(0,strcat('Frequency Calculation File #',num2str(nn)));
% It calculates the Fourier transform of the time profile of each pixel in
% the image + finds the maxima of the Fourier transform. If peaks are
% found, the peaks are sorted and saved in a FrequencyMap

for ll=1:Sz.NLines
    waitbar(ll/Sz.NLines,h);
    for cc=1:Sz.NCol
        F=fft(squeeze(Ima(ll,cc,:)/max(Ima(ll,cc,:))));
        PSD=F.*conj(F);%Power Spectral density
        Abs_F=smooth(abs(F(2:floor(end/2))));


[~,LocalFreq]=findpeaks(Abs_F,'MinPeakDistance',10,'NPeaks',5,'SortStr','descend');

[~,LocalFreq2]=findpeaks(PSD(2:floor(end/2)),'MinPeakDistance',10,'NPeaks',5,'SortS
tr','descend');
        LocalFreq=LocalFreq*AcqFreq/(length(F)-1);
        LocalFreq2=LocalFreq2*AcqFreq/(length(F)-1);
        Freq(ll,cc,:)=LocalFreq;
        Freq_FromPSD(ll,cc,:)=LocalFreq2;



    end
end
close(h)
end
```