nature protocols

https://doi.org/10.1038/s41596-024-00955-7

Protocol

Check for updates

Profiling native pulmonary basement membrane stiffness using atomic force microscopy

In the format provided by the authors and unedited

Supplementary Information (SI)

Profiling native pulmonary basement membrane stiffness using atomic force microscopy

Bastian Hartmann, Lutz Fleischhauer, Monica Nicolau, Thomas Hartvig Lindkær Jensen, Florin-Andrei Taran, Hauke Clausen-Schaumann, Raphael Reuten

Topics:

- Force Curve Corrections
- Implementing Additional File Types
- Young's Modulus Histogram

Force Curve Corrections

The described procedures are applied in this order to every loaded raw curve in our *Force Curve Analysis* application to transform it into a force-indentation curve before a modified Hertz model is applied to determine the Young's modulus. The procedures are as follows:

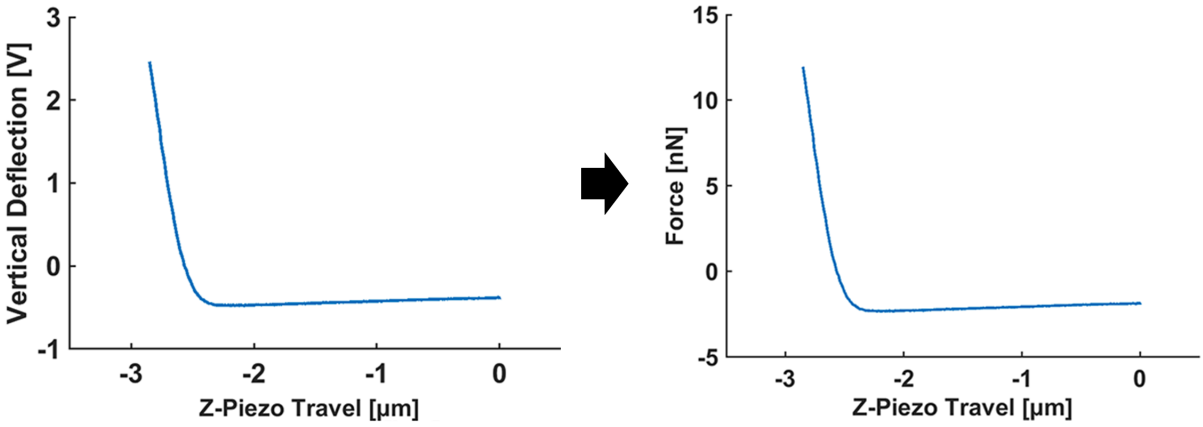
1. Transformation of the recorded vertical cantilever deflection to the applied force
2. Detection and correction of the baseline
3. Detection of the contact point x position and shifting it to 0
4. Converting the z-piezo travel to the vertical tip position

In the following, each procedure is described in detail and the applied corrections are visualized on an exemplary curve.

1) Transformation of the recorded vertical cantilever deflection to the applied force.
(for more details see: https://github.com/CANTERhm/CANTER_Processing_Tool/wiki/Converting-deflection-to-force)

To transform the recorded vertical deflection of the cantilever (*VDefl in* [V]) into the applied force (*F* in [N]) on the sample, it is multiplied with the determined inverted optical lever sensitivity (*InvOLS* in [m/V]) and the determined spring constant of the cantilever (k in [N/m]).

$$F(VDefl) = VDefl \cdot InvOLS \cdot k$$



Supplementary Fig. 1 | Transformation of the vertical deflection in volts of a raw curve (left) to the applied force in newtons (right).

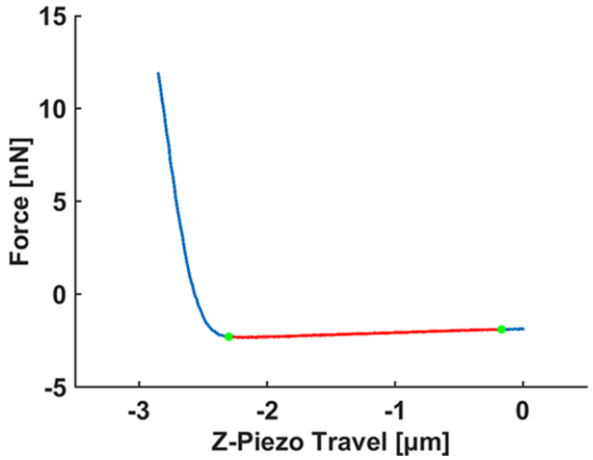
2) Detection and correction of the baseline.
(for more details see: https://github.com/CANTERhm/CANTER_Processing_Tool/wiki/Baseline-detection-and-correction)

The baseline of a force-indentation curve is the section of the curve where the probe is not in contact with the sample but moves toward its surface. Our software automatically detects and corrects the baseline of the force curve. Per default, the offset and tilt of the baseline are both transformed to 0.

The baseline detection algorithm of our *Force Curve Analysis* application conducts the following general steps to detect the baseline in a force curve automatically:

a) A fit window (20% of force curve x-length) is moved in 1% steps along the x-axis of the force curve and the slope of each curve section is determined.
b) The fit window with the lowerest observed absolute slope is determined and used as the initial baseline estimation.
c) To refine the baseline estimation, now the force curve is fitted using polynomials with orders ranging from 3 to 50 with a stepsize of 1.
d) Of the polynomial giving the highest adjusted R^2 value, the second derivative is calculated.
e) The roots of the second derivative are calculated to obtain the deflection points of the force curve.
f) Starting from the previously determined baseline window (step b), the left baseline edge is shifted to the last deflection point closest to the contact point of the force curve.
g) The right edge is either the first recorded point of the force curve or the deflection point (if it exists) closest to the previously determined right window edge from step b.
h) A linear function is fitted to the force curve between the determined left and right baseline edges and it is checked if the linear fit is in agreement with the force curve with respect to the baseline noise. If not, the range is slightly decreased until this criterion is fulfilled.

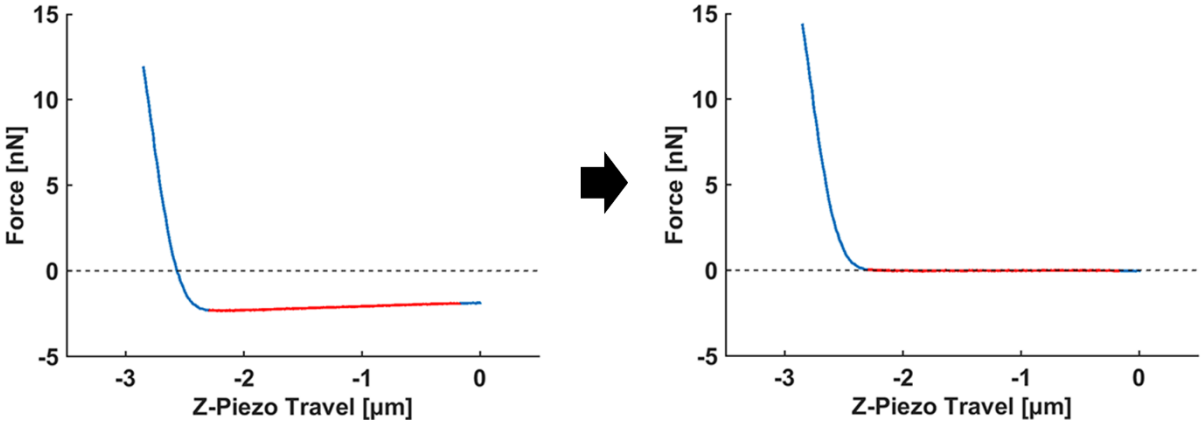
The automatically detected baselined edges using this algorithm are shown on the example force curve in Supplementary Fig. 2 as green dots. The resulting baseline region, used for the subsequent baseline correction is highlighted in red.



Supplementary Fig. 2 | Force curve with automatically detected baseline.

After the detection of the baseline, offset and tilt are corrected. This means the tilt is eliminated from the force curve by subtracting a first-order polynomial fitted on the detected baseline. Subsequently,

the offset is corrected by shifting the mean of the detected baseline to 0 N. The result of the baseline correction can be seen in Supplementary Fig. 3.

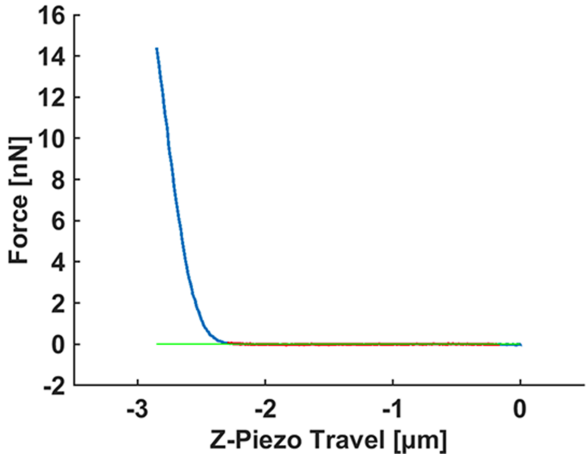


Supplementary Fig. 3 | Exemplary force curve before (left) and after (right) the correction of the baseline tilt and offset.

3) Detection of the contact point x position and shifting it to 0.
(for more details see: https://github.com/CANTERhm/CANTER_Processing_Tool/wiki/Contact-point-correction)

After the baseline of the force curve has been corrected, the next step is to determine the contact point. The contact point is the z-position of the piezo where the tip first got in contact with the sample surface. After the contact point has been determined, the corresponding value on the x-axis of the curve is shifted to 0 µm. Algorithms to detect the contact point implemented in our *Force Curve Analysis* application are "*via Intersect*" and "*via Hertz model*". When "*via Hertz* model" is selected, the "*via Intersect*" algorithm is applied first on the force curve. In the following, the two contact-point determination algorithms are explained in detail.

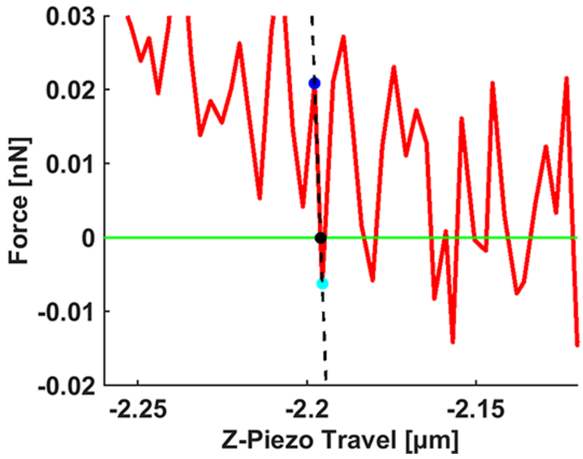
a) *Via Intersect*
In the previous correction step, the baseline of the curve has already been determined. The first step of this algorithm is to fit a linear function (green line in Supplementary Fig. 4) on the baseline (red highlighted curve section).



Supplementary Fig. 4 | Linear function (green line) fitted on the detected baseline (red section of the curve).

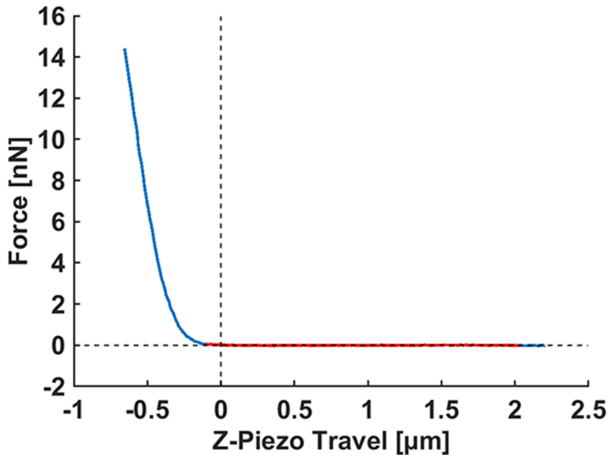
Subsequently, the last point (on the left side of the baseline) of the force curve lower than the fitted green line is determined (see the cyan solid circle in Supplementary Fig. 5). Additionally,

the next (one step to the left) point of the force curve is determined (dark blue dot in Supplementary Fig. 5), Subsequently, the intersect between the fitted baseline (green line) and the line defined by two selected curve points (black dashed line) is calculated. The determined intersect point (black solid dot) is the contact point of the "via Intersect" algorithm.



Supplementary Fig. 5 | Zoom in on the contact point region of the force curve.

After shifting the x-value of the determined contact point to 0 µm, the resulting force curve is shown in Supplementary Fig. 6.



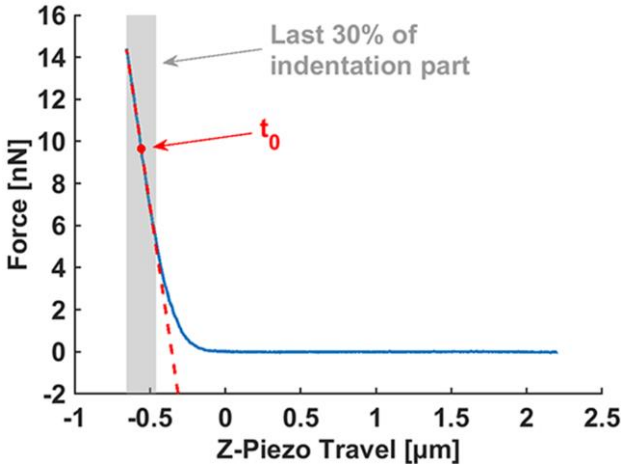
Supplementary Fig. 6 | Force curve with determined and corrected contact point using the *via Intersect* algorithm.

b) *Via Hertz model*

Even when "*via Hertz model"* is selected in our application, the initial step is always the contact point correction using the "*via Intersect*" algorithm (see section 3a). Thus, the "*via Hertz model*" algorithm starts with a force curve where the contact point is already pre-defined, as shown in Supplementary Fig. 6.

The idea of the "*via Hertz model*" algorithm is to determine the contact point using a linear fit on a defined end range (defined as a percentage value) of the contact part ($x < 0$) of the force curve and calculating the contact point of a modified Hertz model using the determined slope and y-axis intercept. This approach is faster compared to directly fitting a Hertz model with the boundary condition for the contact point of $y = 0$ and it is independent of the quality of the first contact between tip and sample.

In our *Force Curve Analysis* application, the user defines a minimum and maximum indentation depth, the Hertz fit for determining the sample Young's modulus is fitted to. Additionally, the user defines a percentage value when using the "*via Hertz model*" algorithm which defines the x range for the linear fit in the indentation part (see the grey area for an exemplary value of 30% in Supplementary Fig. 7). This linear fit range is determined as the percentage (user selection) of the indentation part starting from the defined maximum indentation depth to the pre-defined contact point (x = 0). If the maximum indentation depth is higher than the depth reached with the current force curve, the range is calculated as the percent of the depth between the last (most left) force curve data point to 0, as shown in the representative curve (Supplementary Fig. 7).
For the determination of the contact point from a linear fit, the modified Hertz model was linearized using the Taylor series on the evaluation point t_0 which is defined in the middle of the linear fit range (see Supplementary Fig. 7).



Supplementary Fig. 7 | Visualization of the fit range for the linear estimation (grey area) and the evaluation point (t_0) for the Taylor series used to linearize the modified Hertz model.

In the following, the linearization of the modified Hertz model is exemplarily described for a four-sided pyramid with a half-opening angle to an edge α_e. The modified Hertz model for this indenter geometry and an unconstrained contact point position is:

$$F(d) = \frac{1}{2}\,\frac{E \cdot \tan(\alpha_e)}{(1-\nu^2)} \cdot (d - d_0)^2.$$

The parameters of this expression are:

F	The applied force on the sample as a function of the indentation depth d.
d	The indentation depth of the indenter into the sample.
E	Young's modulus of the sample.
ν	Poisson's ratio of the sample.
α_e	The half-opening angle to an edge of the four-sided pyramid.
d_0	The contact point position on the x-axis where the tip starts to indent the sample.

Supplementary Table 1 | Parameters of the modified Hertz model for a four-sided pyramid.

For the linearization, we only use the first two terms (power of 0 and 1) of the Taylor series on the evaluation point t_0:

$$f_{lin}(t_0) = f(t_0) + \frac{f'(t_0)}{1!}(x - t_0)$$

$$=> F(d) = \frac{E \cdot \tan(\alpha_e)}{2(1-\nu^2)} \cdot (t_0 - d_0)^2 + \frac{E \cdot \tan(\alpha_e)}{(1-\nu^2)} \cdot (t_0 - d_0) \cdot (d - t_0)$$

In the next step, we transform this linearized function into a standard linear equation of the form $y = m \cdot x + t$. First, we summarize the constants into C:

$$C = \frac{E \cdot \tan(\alpha_e)}{(1-\nu^2)}$$

$$=> F(d) = \frac{C}{2} \cdot (t_0 - d_0)^2 + C \cdot (t_0 - d_0) \cdot (d - t_0)$$

Transformation into the standard linear equation:

$$\begin{aligned}
F(d) &= \frac{C}{2} \cdot (t_0 - d_0)^2 + C \cdot (t_0 - d_0) \cdot (d - t_0) \\
&= \frac{C}{2} \cdot \left({t_0}^2 - 2t_0 d_0 + {d_0}^2\right) + C \cdot (t_0 d - {t_0}^2 - d_0 d + t_0 d_0) \\
&= \frac{C}{2} \cdot \left({t_0}^2 - 2t_0 d_0 + {d_0}^2\right) + \frac{C}{2} \cdot (2t_0 d - 2{t_0}^2 - 2d_0 d + 2t_0 d_0) \\
&= \frac{C}{2} \cdot \left(2t_0 d - 2d_0 d + {t_0}^2 - 2{t_0}^2 + {d_0}^2 - 2t_0 d_0 + 2t_0 d_0\right) \\
&= \frac{C}{2} \cdot \left(2t_0 d - 2d_0 d - {t_0}^2 + {d_0}^2\right) \\
&= C t_0 d - C d_0 d - \frac{C \cdot \left({t_0}^2 - {d_0}^2\right)}{2} \\
&= C \cdot (t_0 - d_0) \cdot + \frac{C \cdot \left({d_0}^2 - {t_0}^2\right)}{2}
\end{aligned}$$

Relating linearized model to slope m and y-axis intercept t of the standard linear equation:

$$=> F(d) = \underbrace{C \cdot (t_0 - d_0)}_{m} \cdot d + \underbrace{\frac{C \cdot \left({d_0}^2 - {t_0}^2\right)}{2}}_{t}$$

$$=> \; m = \; C \cdot (t_0 - d_0) \qquad \& \qquad t = \frac{C \cdot \left({d_0}^2 - {t_0}^2\right)}{2}$$

In the next step, we use the derived relations of the modified Hertz model parameters with the slope and y-axis intercept of the linear fit to calculate the contact point position of the corresponding Hertz model d_0. Therefore, we convert both expressions to C

$$=> C = \frac{m}{(t_0 - d_0)} \qquad \& \qquad C = \frac{2t}{\left({d_0}^2 - {t_0}^2\right)}$$

and equate them which results in

$$\frac{m}{(t_0 - d_0)} = \frac{2t}{\left({d_0}^2 - {t_0}^2\right)}.$$

To solve this equation for d_0, we transform the expression above to the form of the quadratic standard form $ax^2 + bx + c = 0$:

$$\frac{m}{(t_0 - d_0)} = \frac{2t}{\left({d_0}^2 - {t_0}^2\right)}$$

$$=> m\left({d_0}^2 - {t_0}^2\right) = 2t(t_0 - d_0)$$

$$=> m{d_0}^2 - m{t_0}^2 = 2tt_0 - 2td_0$$

$$=> \underbrace{m}_{a} \cdot {d_0}^2 + \underbrace{2t}_{b} \cdot d_0 \underbrace{-m{t_0}^2 - 2tt_0}_{c} = 0.$$

Now we use the quadratic formula

$$x_{1,2} = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

with the determined coefficients

$$a = m$$
$$b = 2t$$
$$c = -m{t_0}^2 - 2tt_0$$

to determine the d_0 fulfilling the quadratic expression

$${d_0}_{1,2} = \frac{-2t \pm \sqrt{4t^2 - 4m(-m{t_0}^2 - 2tt_0)}}{2m}$$

$$= \frac{-2t \pm \sqrt{4t^2 + 4m^2{t_0}^2 + 8mtt_0}}{2m}$$

$$= \frac{-t \pm \sqrt{t^2 + 2tmt_0 + m^2{t_0}^2}}{m}$$

$$= \frac{-t \pm \sqrt{(t + mt_0)^2}}{m}$$

$$= \frac{-t \pm (t + mt_0)}{m}$$

$$=> d_0 = \begin{cases} -\frac{t}{m} + \frac{t}{m} + t_0 = t_0 & \rightarrow \text{not a reasonable solution!} \\ -\frac{t}{m} - \frac{t}{m} - t_0 = -\frac{2t}{m} - t_0 & \rightarrow \text{reasonable solution!} \end{cases}$$

Thus, the result is an equation to determine the contact point position d_0 using the slope m and y-axis intercept t derived from a linear fit and the evaluation point of the Taylor series t_0:

$$\boxed{d_0 = -\frac{2t}{m} - t_0}.$$

For the example curve, the linear fit on the last 30% gives the following results for the parameters:

$$m = -0.0479 \ \frac{\mathrm{N}}{\mathrm{m}}$$

$$t = -16.99\ \text{nN}$$

$$t_0 = -0.557\ \mu\text{m}$$

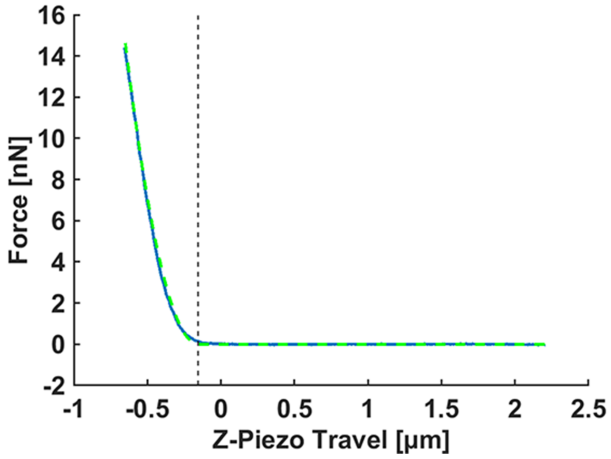
Using these parameters, the calculated contact point of a modified Hertz model which has the slope m on the x-value of t_0 is

$$d_0 = -\frac{2 \cdot (-16.99\ \times 10^{-9}\ \text{N})}{-0.0479\ \frac{\text{N}}{\text{m}}} - (-0.557 \times 10^{-6}\ m) = -1.524 \times 10^{-7}\ \text{m}$$

$$d_0 = -0.1524\ \mu\text{m}$$

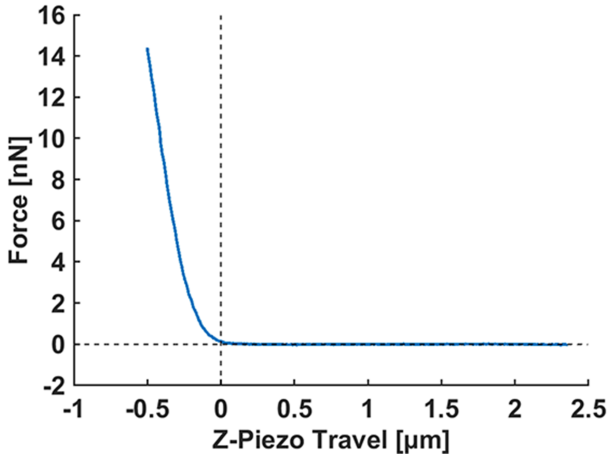
In Supplementary Fig. 8 the contact point position determined with this algorithm is shown as a vertical dashed black line. Additionally, the resulting modified Hertz model with the calculated contact point d_0 and the slope m at the x position t_0 is visualized as a green dashed line.

Important to note here is that in our software, this resulting modified Hertz model is not used to determine the Young's modulus of the sample. It is only used to determine the contact point in this algorithm. The Young's modulus of the sample is determined after the corrections described in this SI by fitting a modified Hertz model to the indentation depth range defined by the user in the edit fields *Fit start* and *Fit depth* in the Force Curve Analysis application.



Supplementary Fig. 8 | Visualization of the determined contact point by the *„via Hertz model"* algorithm.

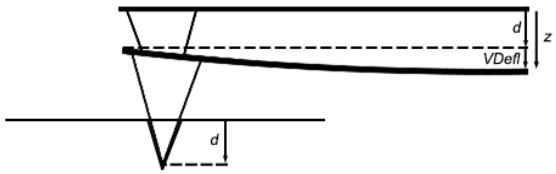
Finally, the detected contact point position is shifted to zero. Thus, the resulting force curve after the contact point correction „via Hertz model" algorithm is shown in Supplementary Fig. 9.



Supplementary Fig. 9 | Resulting force curve after the detection of the contact point and shifting ist x-position to 0.

4) Converting the z-piezo travel to the vertical tip position.
(for more details see: https://github.com/CANTERhm/CANTER_Processing_Tool/wiki/Calculating-vertical-tip-position)

The last correction step, before any mechanical model is applied to the curve to extract sample parameters like the Young's modulus, is to correct the z-piezo travel (*z*) using the known upwards deflection of the cantilever in meters (*VDefl[m]*) to achieve the vertical tip position. This transformation gives the real depth of indentation into the sample (*d*).



Supplementary Fig. 10 | Schematic to visualize the vertical z-piezo travel (*z*), the vertical deflection of the cantilever (*VDefl*), and the indentation depth (*d*).

This can be done by substracting for each recorded data point of the force curve (*i*) the vertical deflection from the z-piezo travel:

$$d(i) = z(i) - VDefl[m](i).$$

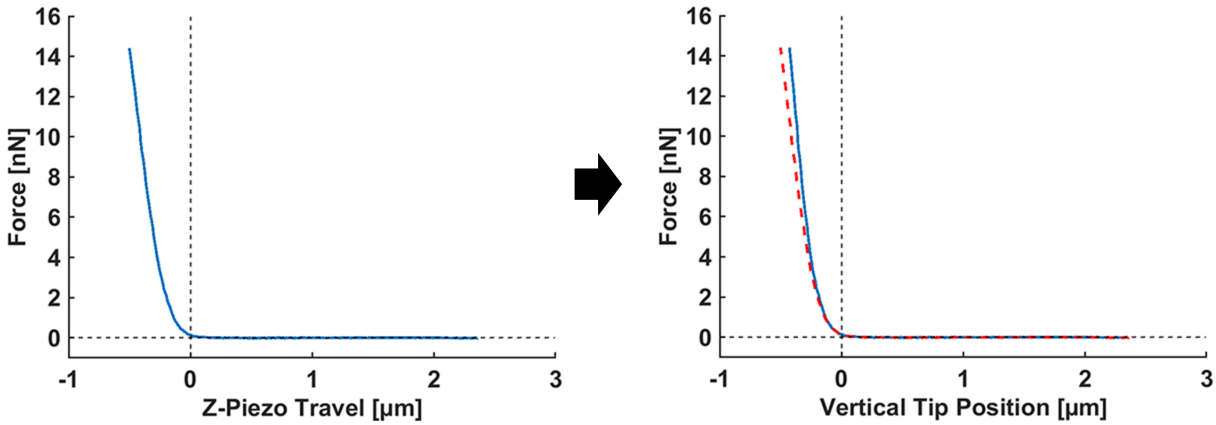
The vertical deflection in meters can be calculated from the recorded vertical deflection signal in volts *(VDefl[V]*) by multiplication with the determined inverted optical lever sensitivity (*InvOLS*):

$$VDefl[m] = VDefl[V] \cdot InvOLS.$$

Taken together, the vertical tip position for each data point of the force curve is calculated using the following equation:

$$d(i) = z(i) - (VDefl[V](i) \cdot InvOLS)$$

In the figure below, the force-indentation curve before (left) and after (right) the correction is displayed (blue solid line). Additionally, in the right figure the curve before the z-travel has been converted into the vertical tip.



Supplementary Fig. 11 | Example curve before (left) and after (right) the transformation of the z-piezo travel to the vertical tip position.

Implementing Additional File Types

To enable the import of additional force volume files into the *Force Curve Analysis* application of our *CANTER Processing Toolbox,* the following changes/insertions in the *bihertz_gui_App.mlapp* file located in the folder *GUI-files\bihertz_fit* have to be made:

1. Insert the new file extension in the filter strings of the *uigetfile* function of the *button_file_Callback* (see Supplementary Fig. 12).

```
% Button pushed function: button_file
function button_file_Callback(app, event)
    % Create GUIDE-style callback args - Added by Migration Tool
    [hObject, ~, handles] = convertToGUIDECallbackArguments(app, event); %#ok<ASGLU>

    % hObject    handle to button_file (see GCBO)
    % eventdata  reserved - to be defined in a future version of MATLAB
    % handles    structure with handles and user data (see GUIDATA)

    [file,path,ext] = uigetfile({'*.jpk-force-map;*.jpk-qi-data','JPK-Data-Maps';...
        '*.txt','Mach-1 text-file'},'Select a File',handles.last_load_path);
```

Supplementary Fig. 12 | Highlighted filter strings of the *„uigetfile"* function inside the „button_file_Callback" function.

Currently, *.jpk-force-map*, *.jpk-qi-data*, and a special kind of text file (Mach-1 text file) can be selected using the file button.

2. Add an additional case to the *Fext* switch for your new file type inside the *file*-case of the *handles.loadtype* switch located in the *button_load_data_Callback* (see Supplementary Fig. 13).

```
switch handles.loadtype
    case 'file'
        if handles.filefilter == 1
           handles.loaded_file_type = 'jpk-force-map';
           % if a jpk-force-map is loaded reset the fit start and
           % fit depth values
           if handles.load_status == 0
               handles.hertz_fit_depth.String = '1';
               handles.hertz_fit_start.String = '0';
           end
           [~,~,Fext] = fileparts(handles.edit_filepath.String);
           switch Fext
               case ".jpk-force-map"
                    [x_data,y_data, ~, ~, Forcecurve_label,~,~,name_of_file,map_images,~,handles.map_info_array] = ReadJPKMaps_App(app,handles.edit_filepath.String);
               case ".jpk-qi-data"
                   ProgBar = uiprogressdlg(app.figure1,"Message","Loading QI Map Data ...","Title","Loading Procedure","Indeterminate","on");
                   [x_data,y_data, ~, ~, Forcecurve_label,~,~,name_of_file,map_images,~,handles.map_info_array] = ReadQIMaps(handles.edit_filepath.String);
                   close(ProgBar);
           end
```

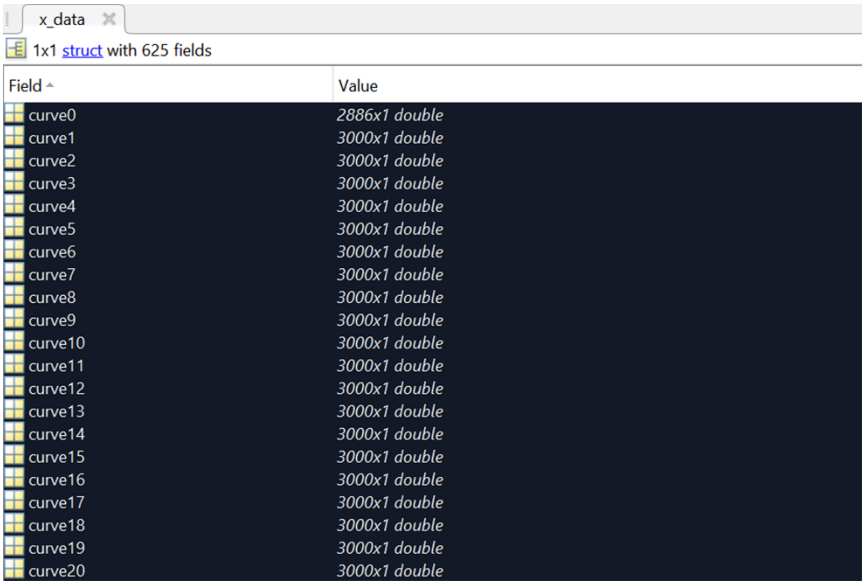
Supplementary Fig. 13 | The „Fext" switch where the import function of the new file type has to be added located inside the „button_load_data_Callback" function.

In the screenshot above, you can see the already existing cases for the file types *.jpk-force-map* and *.jpk-qi-data*.

3. Write an import function for your custom file type. This function need to fulfill the following criteria:
   -accept the full path to your new file as string or character variable as input parameter. Give the function for this imput parameter the user selected path stored in the "app.edit_filepath.String" variable.

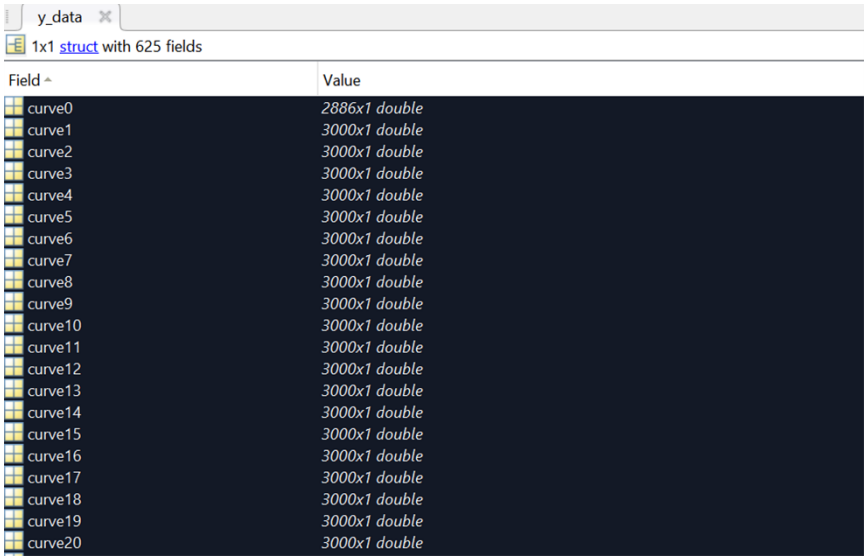
-must have the following 11 output parameters in the corresponding output order on the specified position number:

1. **x_data:** The extend x data of the imported force curves as a struct variable containing a field for each force curve named “curveX”, where X is the index of the corresponding force curve starting at 0 (see Supplementary Fig. 14). Each field contains the x data of the corresponding curve as a double vector in m.



Supplementary Fig. 14 | Example of the first 20 fields of the „x_data“ struct.

2. **y_data:** The extend y data of the imported force curves as a struct variable containing a field for each force curve named “curveX”, where X is the index of the corresponding force curve starting at 0 (see Supplementary Fig. 15). Each field contains the y data of the corresponding curve as a double vector in V:



Supplementary Fig. 15 | Example of the first 20 fields of the „y_data“ struct.

3. and 4. are not used in the *Force Curve Analysis* application.

5. **Forcecurve_label:** This output varualbe has to contain the labels of the fields in the data structs (output parameters 1 - 4) as a one-column cell array where each cell contains the character sequence of one curve label like shown in Supplementary Fig. 16.

Forcecurve_label

625x1 cell

	1	2	3
1	curve0		
2	curve1		
3	curve2		
4	curve3		
5	curve4		
6	curve5		
7	curve6		
8	curve7		
9	curve8		
10	curve9		
11	curve10		
12	curve11		

Supplementary Fig. 16 | Example of the first 12 entries of the „Forcecurve_label“ cell array.

6. And 7. are not used in the *Force Curve Analysis* application.

8. **name_of_file:** Name of the selected file as character array.

9. **map_images:** A struct containing all loaded channel images from the force volume file. An example resulting from a QI-Map file is shown in Supplementary Fig. 17.

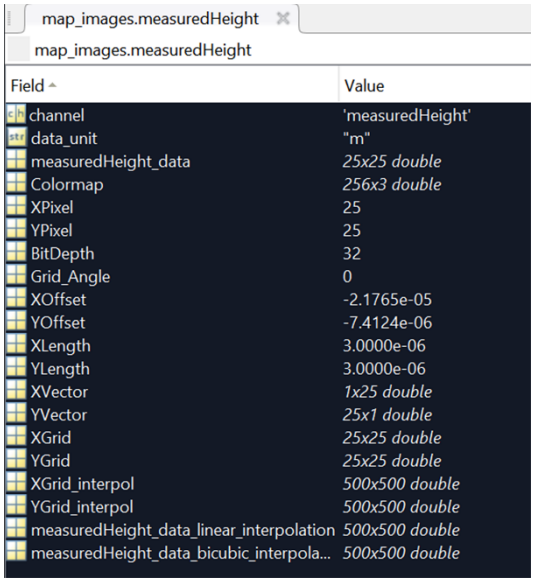
map_images

1x1 struct with 6 fields

Field	Value
thumbnail	*1x1 struct*
vDeflection	*1x1 struct*
measuredHeight	*1x1 struct*
slope	*1x1 struct*
height	*1x1 struct*
adhesion	*1x1 struct*

Supplementary Fig. 17 | Example of the „map_images“ struct fields after loadin a .jpk-qi-data file.

Each field is again a struct containing all image information for the corresponding channel. Example of the *measuredHeight* channel of a 3 µm x 3 µm QI-Map containing 25x25 force-indentation curves is shown in Supplementary Fig. 18.

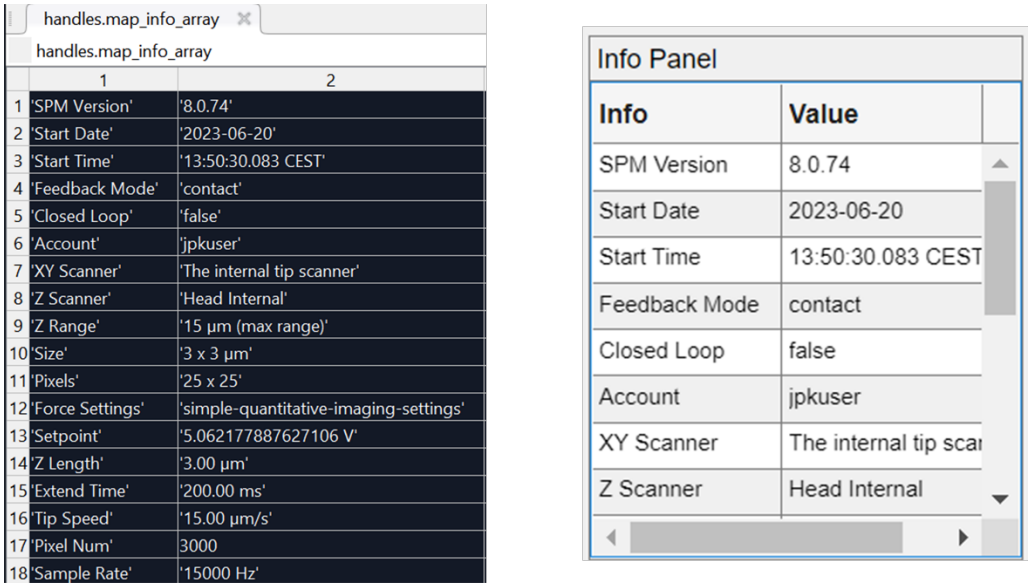


Supplementary Fig. 18 | Strunct contained in the „measuredHeight“ field of the „map_images“ struct.

The code to import images from JPK map files can be found in the function “ForceMapImageData” located in the folder “IO-functions” → ”Read_functions” of our CANTER Processing Toolbox.

10. Not used in the *Force Curve Analysis* application.

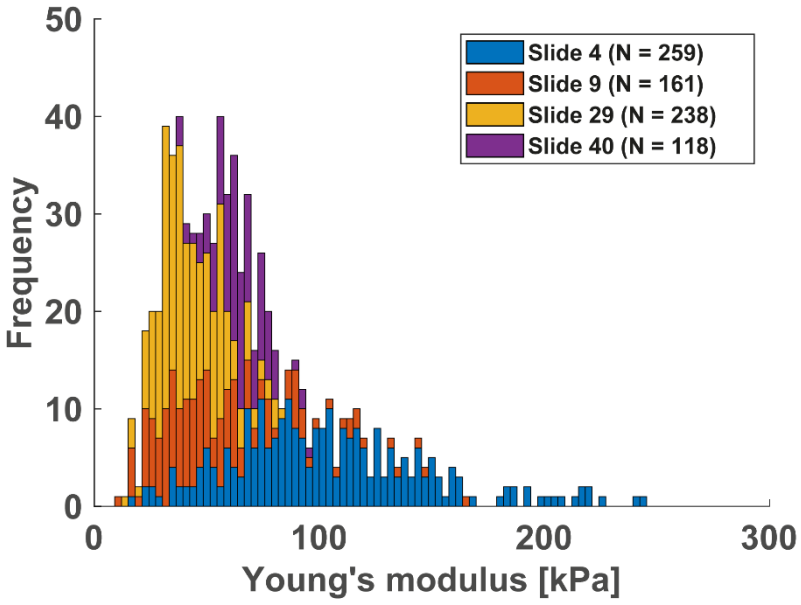
11. **handles.map_info_array:** A two-column cell array (left image of Supplementary Fig. 19) containing map information that will be displayed in the Info-panel (right image of Supplementary Fig. 19) of the *Force Curve Analysis* application:



Supplementary Fig. 19 | (left) „handles.map_info_array“ cell array containing the map information read from the header files of a .jpk-qi-data file. (right) The information stored in the „handles.map_info_array“ cell array presented in the „Info Panel“ of the *Force Curve Analysis* application.

Young's Modulus Histogram

In Supplementary Fig. 20, the combined histogram of the Young's modulus results from 4 human pulmonary BMs (Slides 4, 9, 29, and 40) is shown. Because the Young's modulus is a quantity that can only attain positive values, without transformation it is not normally distributed. Thus, before determining summary statistics using a normal distribution like the mean, the values must be normalized using a log transformation.



Supplementary Fig. 20 | Combined Young's modulus histogram of the results of four human pulmonary BMs showing a right-skewed distribution.