FilamentSensor Git

The FilamentSensor is grouped into different modules to separate functionality.

FAThresholdOptimizator: this one contains GUI and functionality since its a small and simple tool for checking the “best” automatic thresholding algorithm for thresholding focal adhesions(the best from the IJ thresholding algorithms).

FXControls: this module contains UI-Custom-Controls created for Filamentsensor GUI and FA-GUI

GUI: this module contains the whole FilamentSensor GUI functionality, the data is mostly contained by objects from package “model”, the GUI itself is designed in the “view” package (fxml files) and the functionality is contained by the package control. Additional styling can be done by css which is found in the package “style”

GUIFocalAdhesionOnly: this module contains the whole FocalAdhesion-Tool GUI, it makes also use of the module GUI as it contains a basic FilamentSensor in the application. This module has its own main method to start the application, found in “fa.view.MainWindow”. The basic structure is similar to the GUI-module. Model, View, Control.

Main: this module contains the main entry point for the FilamentSensor since it was also used for fast testing of newly produced method (development class was used for this).

Sensor: this module is the core of the FilamentSensor it contains the whole processing functionality and data storage also the image/csv/xml export functionality. It is no standalone application, it is used more like an API to be integrated in other applications. Like done in “GUI” module and GUIFocalAdhesionOnly module.

The UserInterface parts of the modules follow some rough Model, View, Controller pattern. Models handle the Data, Controller the program flow and View the visual aspect.

The filament sensor makes a lot of use of multi-threading, which also consumes a good amount of RAM, due to the nature of parallel computation.

There are some memory “optimizations” done like not storing binary images as boolean[][] since java does not use 1bit per boolean value. These 2D boolean arrays are mostly encoded as Bitset which uses datastructures like int[] to encode the boolean[] into the integer values. Makes access a bit slower (since encode/decode) but safes a lot of memory.

There was memory testing done and Bitset seem to be the most appropriate option for use.
Its internally handled by the class BinaryImage.

Class Diagrams can be found in the “doc” directory.