ECG Interpolation - Detailed Documentation

This JavaFX-based application enables interactive visualization and filtering of ECG signals using various signal processing algorithms. It supports segmented filtering based on R-peaks and provides an interactive graphical user interface built with JavaFX.

Key Features

- Reading ECG signals from XML files
- Graphical rendering using JavaFX
- Filtering algorithms:
    - Gaussian Moving Average
    - Savitzky-Golay
    - LOESS
    - Cubic Spline
    - Wavelet (experimental, not yet implemented)
- Segmented filtering around R-peaks
- Interactive user interface:
    - Filter settings and visibility control
    - Zoom and signal navigation
    - R-peak visualization

System Requirements

- Java JDK 17 or later
- Maven 3.6+

Installation

1. Clone the project
2. Build the project
3. Run it using Maven

Project Structure

```
ecginterpolation/
├── pom.xml                    # Maven configuration
├── src/
│   ├── main/
│   │   ├── java/hu/ujvari/      # Java source code organized in modules
│   │   └── resources/xml/       # ECG test files (ecg1.xml, ecg2.xml, ecg3.xml)
│   └── test/                  # JUnit tests
├── .vscode/                   # Optional VS Code settings
└── .gitignore
```

Example Files

Three example ECG XML files are available in `src/main/resources/xml/` (ecg1.xml, ecg2.xml, ecg3.xml) to test the application.

Main Class

Entry point: `hu.ujvari.ECGMenuApp`

This JavaFX-based main menu allows:

- Launching the ECG plotter
- Analyzing XML files (structure, full content view)
- Exiting the application

Developed using Java 17+ and JavaFX 21.

Project Documentation

Configuration Files

- `.vscode/launch.json`: VS Code run configurations
- `.vscode/settings.json`: Project-specific settings
- `pom.xml`: Maven configuration file

Source Structure

```
ecginterpolation/
├── .gitignore
├── pom.xml
├── struktura.txt
├── src/
│   ├── main/java/hu/ujvari/
│   │   ├── ECGMenuApp.java
│   │   ├── ecgmodel/ → Signal.java
│   │   ├── ecgplotter/
│   │   │   ├── controller/ → FilterController.java, ViewController.java
│   │   │   ├── filter/ → FilterInterface.java, GaussianFilter.java, LoessFilter.java,
SegmentedFilterAdapter.java, SgFilter.java, SplineFilter.java, WaveletFilter.java
│   │   │   ├── model/ → FilterParameters.java, SignalData.java
│   │   │   ├── view/ → FilterControlPanel.java, FilterVisibilityPanel.java,
NavigationPanel.java, SignalCanvas.java, StatusPanel.java
│   │   │   └── unused/ → CanvasEcgPlotter.java, EcgPlotter.java,
EcgPlotterApplication.java, FastEcgPlotter.java
│   │   ├── ecgprocessor/ → CubicSplineFilter.java, ECGSegmenter.java,
GaussianMovingAverage.java, LoessFilter.java, SavitzkyGolayFilter.java, WaveletFilter.java
│   │   └── ecgreader/ → XmlEcgReader.java
│   └── resources/xml/ → ecg1.xml, ecg2.xml, ecg3.xml
│   └── test/java/hu/ujvari/ → AppTest.java
├── target/                    # Compiled files – do not modify manually
```

Resources

- `resources/xml/`: Contains ECG data files

Main Components

ECGMenuApp.java

JavaFX GUI with a main menu containing:

- ECG Plotter
- XML Analyzer
- Exit

EcgPlotterApplication.java

Responsible for:

- Loading ECG signal from XML via static `setData()`
- Registering and applying multiple filters (Gaussian, Savitzky-Golay, LOESS, Spline)
- Building GUI with SignalCanvas, FilterControlPanel, NavigationPanel, FilterVisibilityPanel, StatusPanel

SignalData.java

Stores and manages:

- Original signal
- Filtered signals
- Minimum/maximum value tracking
- Viewport control: `moveViewport()`, `setZoomLevel()`, `resetViewRange()`
- Thread-safe design with synchronized access

Filters

FilterParameters.java

Abstract class storing filter parameter types:

- GaussianParameters
- SavitzkyGolayParameters
- LoessParameters

- SplineParameters
- WaveletParameters
- SegmentFilterParameters

FilterInterface.java

Defines methods:

- `getName()`
- `filter()`
- `getParameters()` / `setParameters()`

Adapter Classes

- SgFilter, LoessFilter, SplineFilter
- Use underlying implementations from `hu.ujvari.ecgprocessor`

SegmentedFilterAdapter.java

- Applies filters segment-by-segment based on detected R-peaks

Segmentation Logic

ECGSegmenter.java

- R-peak detection using windowed thresholding
- Filtering each segment separately
- Smooth transitions using linear interpolation

GUI Components

SignalCanvas.java

- Renders signals with:
    - Distinct color/transparency for each filter
    - Zoom and drag support
    - R-peak markers
- Methods:
    - `redrawChart()`
    - `drawSignal()`
    - `drawRPeaks()`
    - `setupCanvasEvents()`

FilterControlPanel

- Adjusts filter parameters
- Applies filters and refreshes canvas

FilterVisibilityPanel

- Toggles visibility of each filter output

NavigationPanel

- Handles zoom and navigation

StatusPanel

- Displays messages and progress

Segmented Filtering Example

1. ECGMenuApp loads signal from XML
2. Data is passed to EcgPlotterApplication
3. SignalData object is created
4. Filters are registered in FilterController
5. Segmented Savitzky-Golay is instantiated and registered
6. User selects the segmented filter from FilterControlPanel
7. R-peak detection via ECGSegmenter
8. Signal is split into segments
9. Each segment is filtered
10. Results displayed on SignalCanvas with R-peaks highlighted

Key Components:

- ECGSegmenter
- SegmentedFilterAdapter
- FilterController

This adapter pattern enables flexible use of segmented filters across various algorithms while preserving diagnostic integrity of the signal.