Improving Web2cHMI Gesture Recognition Using Machine Learning

Reinhard Bacher
DESY, Hamburg, Germany

HELMHOLTZ RESEARCH FOR GRAND CHALLENGES

DESY.

Table of Contents

- **Web2cToolkit**:
  - Brief overview
- **Web2cHMI**:
  - Brief overview
- **Supervised ML Use Case**:
  - Movement recognition
- **Summary**:
  - Status
  - Open questions

Web2cToolkit Web Service Collection

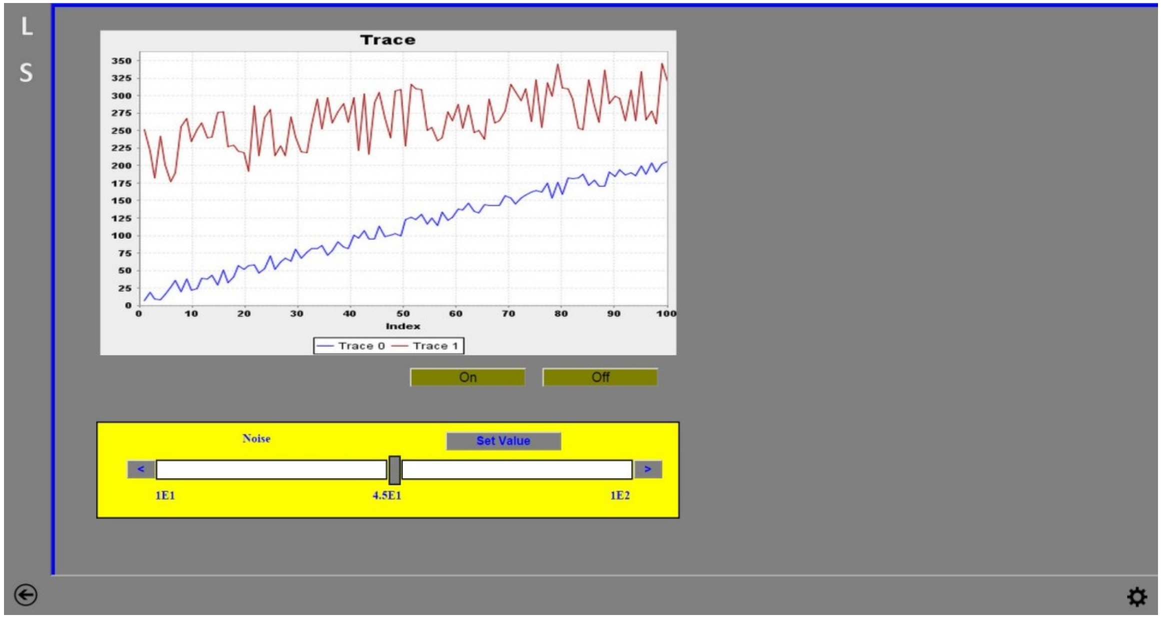
The *Web2cToolkit* is a collection of Web services, i.e. servlet applications and the corresponding Web browser applications, including

Viewer:
- *Web2cViewerWizard*
- *Web2cArchiveViewer*

Graphical GUI-Bilder:
- *Web2cViewerWizard*
- *Web2cArchiveViewer*

Mobile Framework:
- *Web2cToGo*

Miscellaneous:
- *Web2cGateway*
- *Web2cManager*



The *Web2cToolkit* provides interfaces to major accelerator and beam line control systems including TINE, DOOCS, EPICS (via DOOCS) TANGO and STARS.

Web2cHMI (1/2)

***Web2cHMI* is a Web-based, platform-neutral, single-user human machine interface which seamlessly combines actions based on various modalities provided by input devices commonly available from the consumer market.**

Web3cToGo is a test environment for *Web2cHMI*.

Web2cHMI is implemented in JavaScript.

Web2cHMI supports various user input devices including

- Mouse
- Touch-sensitive display
- Leap motion controller
- Myo gesture control armband
- Epson Moverio BT-200 smart glass
- Vuzix M100 smart glass
- Microphone

Web2cHMI (2/2)

All input modalities supported can be used simultaneously including

- 1D/2D at gestures including single- and multi-finger
- gestures,
- 2D/3D spatial gestures including hand and arm gestures,
- 3D head movements including yaw, pitch and roll
- English spoken commands.

The speech recognition system knows a *Web2cHMI*-specific vocabulary listed in a dictionary such as “Browse Up" or “Lot More".

Gestures

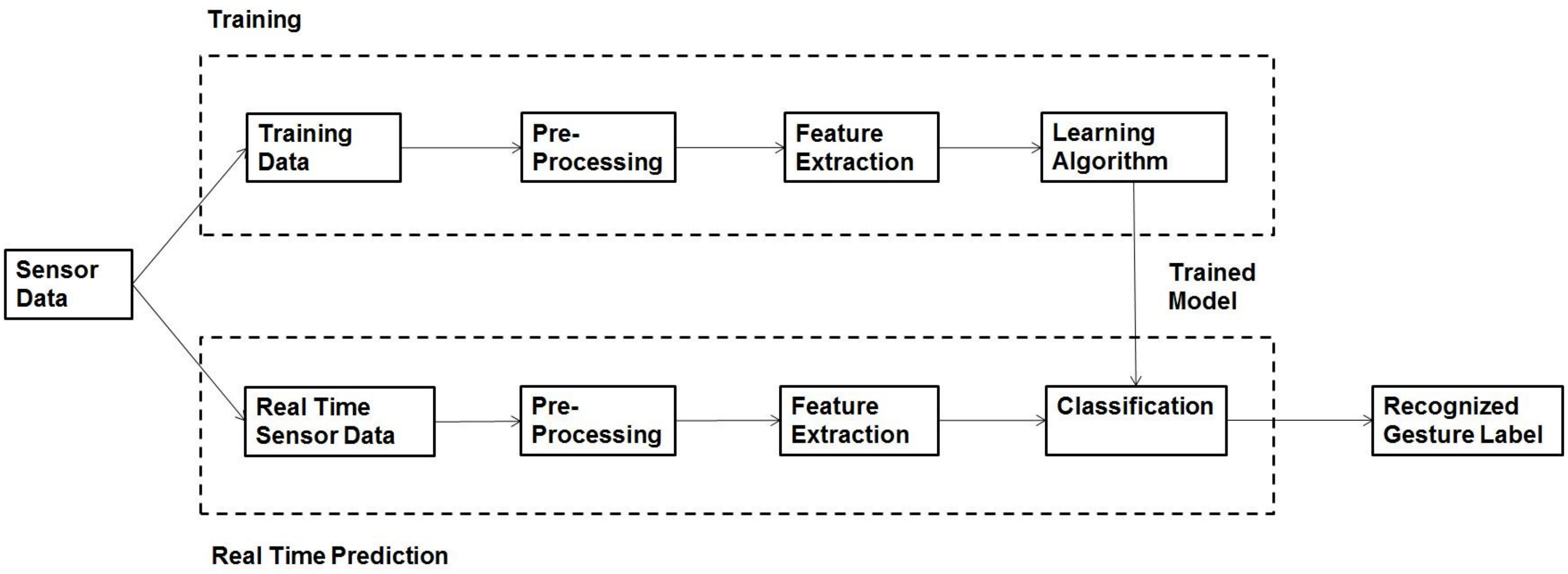
Web2cHMI recognizes various **primitive gestures**, i.e. input device-specific gestures including

- Mouse: Click, Move
- Touch-sensitive display: Tap, Move / Swipe, Pinch (two fingers)
- Leap motion controller: Key-Tap, Swipe, Open-Hand, Closed-Hand, Circle
- Myo gesture control armband: Double-Tap, Wave-Out / Wave-In, Fingers-Spread, Fist
- Smart glass: Move-Fast / Move-Slow, Roll

In addition, **enriched gestures** formed by primitive gestures followed by linear movements or rotations etc. are supported. Examples are Fingers-Spread & Clockwise Rotation or Sideward-Left Long Swipe.

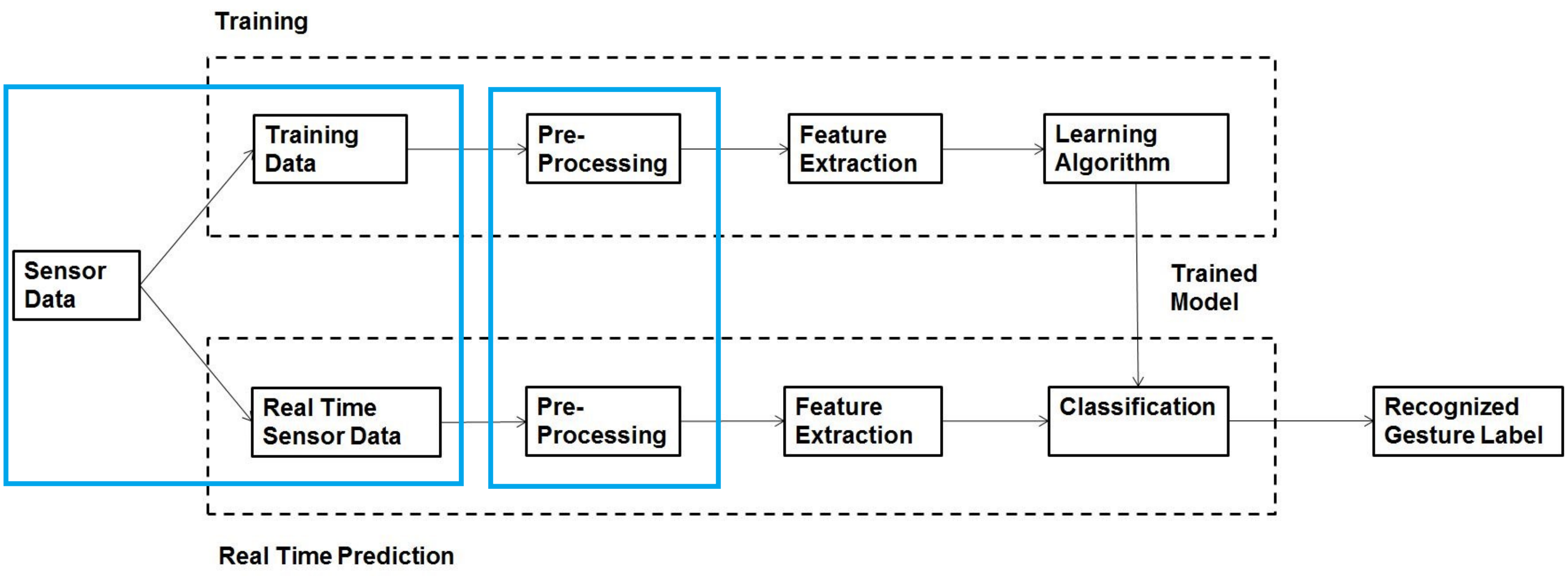
Problem: Reliable detection of arm, hand, finger or head movements of enriched gestures

Supervised ML Gesture Recognition Workflow

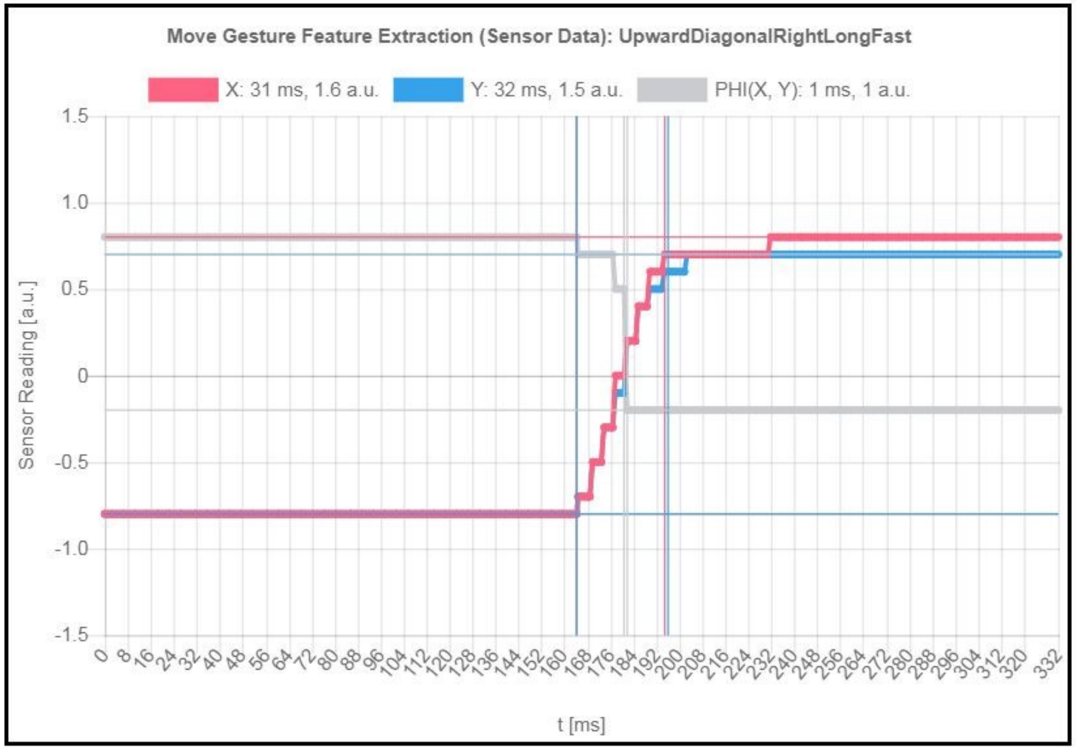


Important: Use the same pre-processing and feature extraction algorithms for both training and real time prediction.

Workflow Steps



Recording Sensor Data / Pre-Processing



Recording Sensor Data:

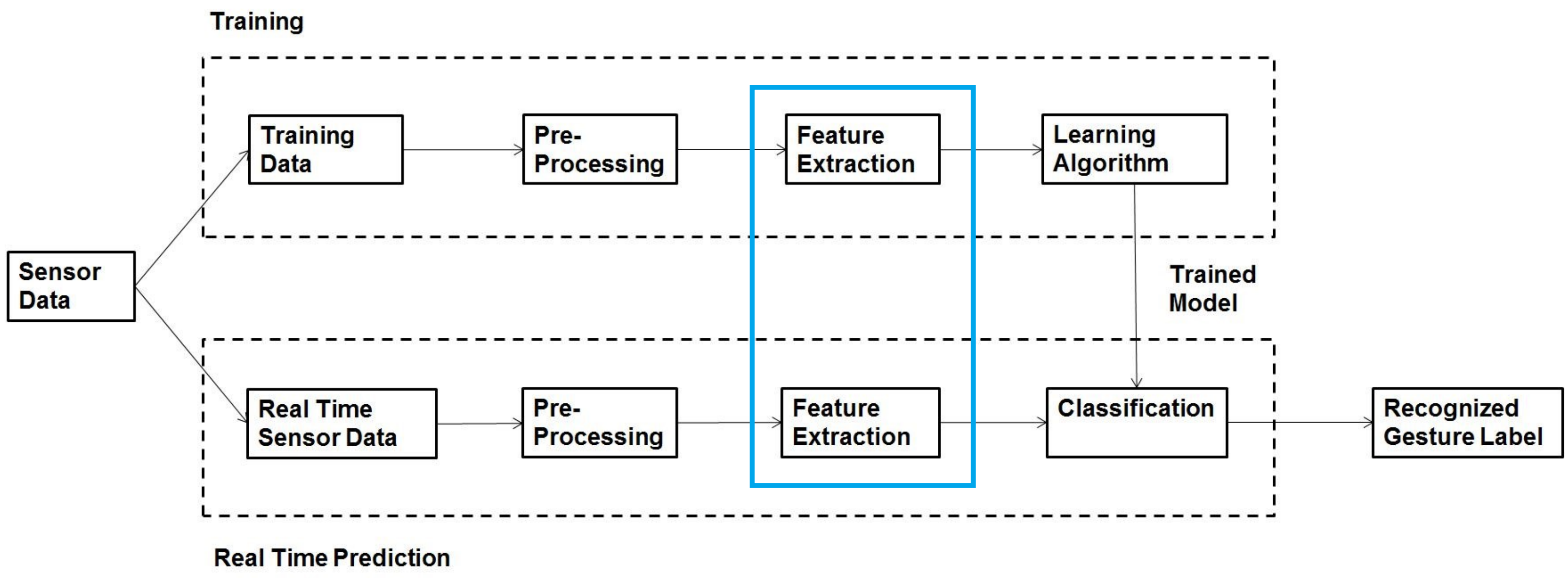
Record continuously the position of the input device in both cartesian (X, Y) and polar (R, PHI) coordinates resulting from

- Finger movements,
- Hand movements,
- Arm movements or
- Head movements

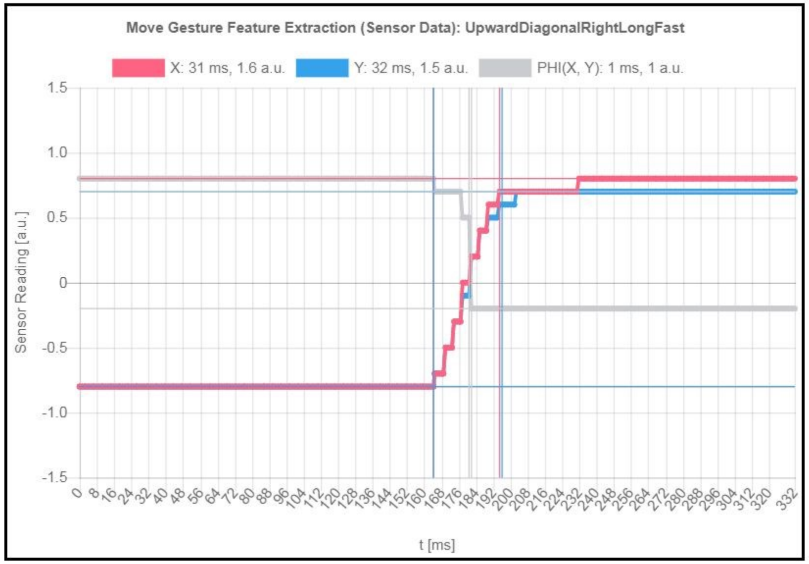
Pre-Processing:

Cleanse the sensor data (reducing noise, removing outliers, …)

Workflow Steps



Feature Extraction



Reduces the dimension of the gesture recognition task.

For each coordinate orientation (horizontal, vertical , polar angle)

- Implement a non-linear regression (**Nelder-Mead Simplex method**) to fit consecutive 333 ms long time sequences of sensor data to a pre-defined mathematical model
- Find start / end time and start / end position of movement (**features**),
- Calculate duration and length of gesture if feature values are confined within reasonable limits.

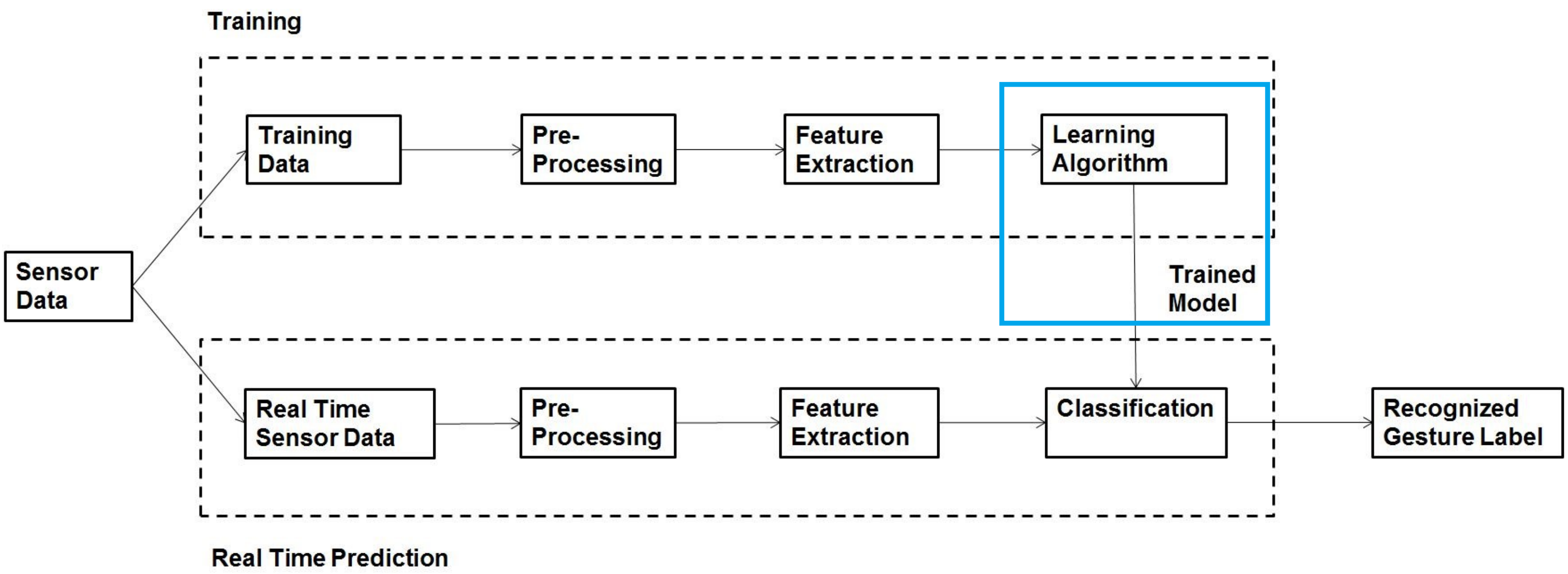
Non-Linear Regression Model

$t \leq t_{start}$: $f(t) = s_{start}$

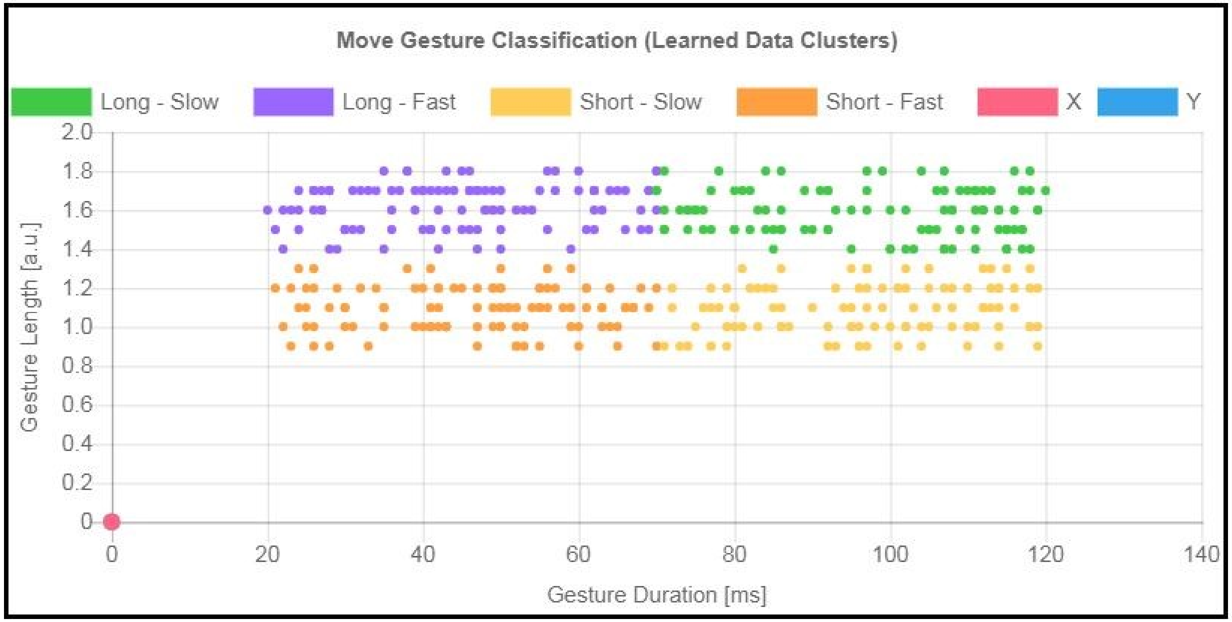
$t_{start} < t < t_{end}$: $f(t) = s_{start} + ((s_{end} - s_{start}) / (t_{end} - t_{start}) * (t - t_{start}))$

$t \geq t_{end}$: $f(t) = s_{end}$

Workflow Steps



Trained Model

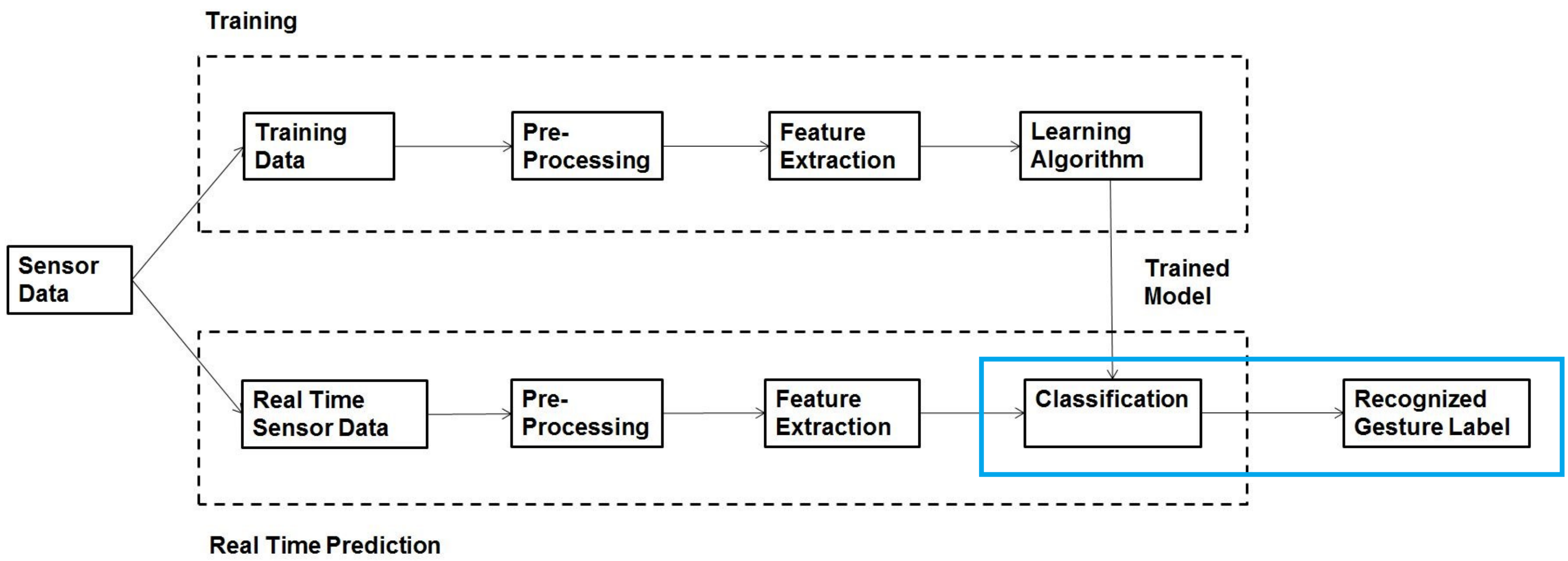


Training the Algorithm by Supervised Learning

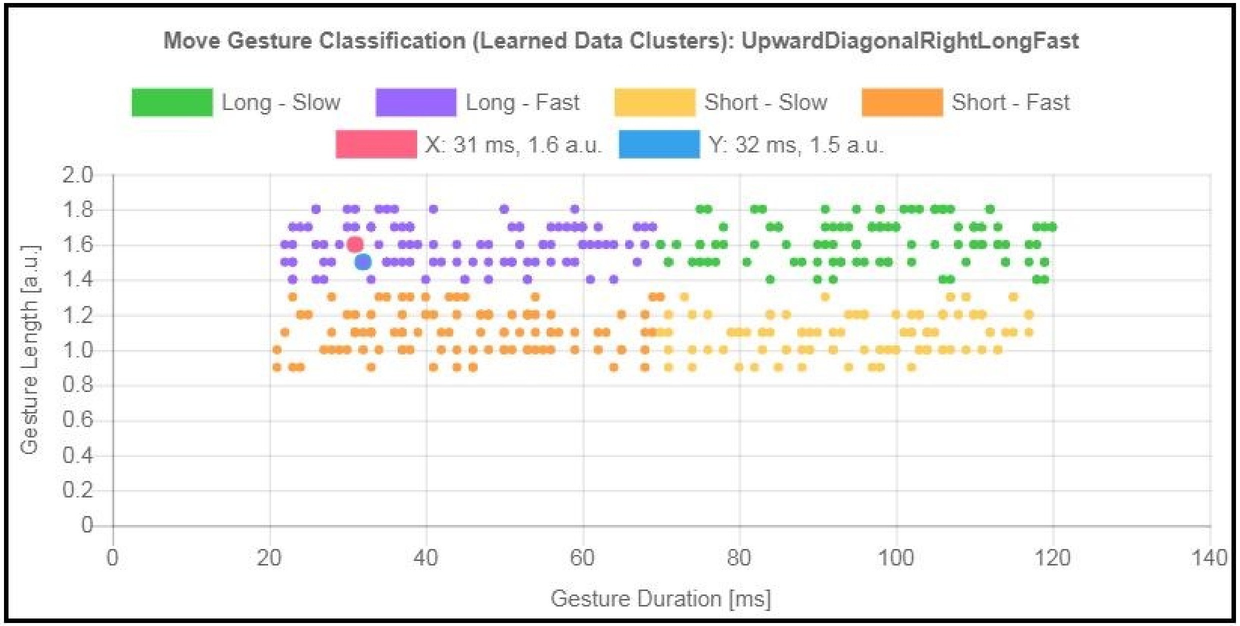
Generate sets (clusters) of learned data (real or simulated) representing

- *Long-Slow* movements,
- *Long-Fast* movements,
- *Short-Slow* movements,
- *Short-Fast* movements.

Workflow Steps



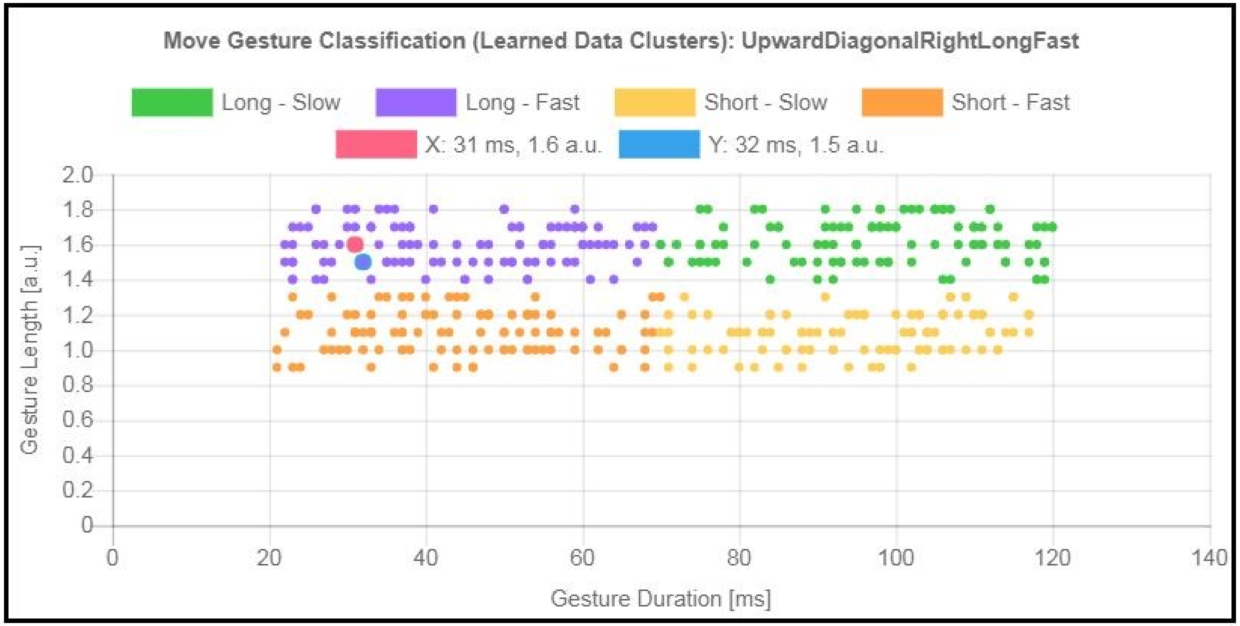
Classification (1/2)



Gesture Orientation

Define gesture orientation (left / right linear movement, up / down linear movement, clockwise / counter clockwise circular movement) using feature values (t_{start}, t_{end}, s_{start}, s_{end}) found.

Classification (2/2)



Gesture Type (**k-Nearest Neighbor Analysis**).

Define proper gesture label (long-slow, long-fast, short-slow, short-fast) by classifying the gesture duration ($t_{end} – t_{start}$) / gesture length ($s_{end} – s_{start}$) values found (red / blue dots) using the memorized learned data sets (k = 3)

Summary

Status:

- Is working pretty well.
- Is still work in progress.

Open questions:

- Input data:
  - How does the quality (i.e. pointing stability) of sensor data affects the prediction quality?
- Feature extraction:
  - Is the non-linear regression the appropriate method?
- Classification:
  - Is the k-nearest neighbor analysis the right method?
  - Which prediction accuracy can be achieved?
- Trained Model:
  - How does the prediction perform with real instead of simulated trained data?

http://web2ctoolkit.desy.de

Thank you

Sun rise at Mt. Jade (3.952 m), October 10th 2018