HANDLE

Developmental pathway towards autonomy and dexterity in robot in-hand manipulation

importDatasetTB

MATLAB toolbox

Version 1.0: Draft

Classification: Restricted

Grant Agreement Number: 231640

Contract Start Date: February 2, 2009

Duration: 48 Months

Project coordinator: UPMC

Partners: UPMC, SHADOW, UC3M, FCTUC, KCL, ORU, UHH, CEA, IST

Project website address: www.handleproject.eu

Version Management

Version	Date	Status	Author	Modification
V1.0	17/02/2010	Draft	FCTUC	Creation of the document.
V2.0	08/07/2010	Draft	FCTUC	Update of the document.

Acronyms

MATLAB	MATrix LABoratory software
SAX	Simple API for XML
DOM	Document Object Model

Table of Contents

1 Introduction

This toolbox was developed in order to promote an easy integration on MATLAB software scripts of the data contained on the datasets stored in the web platform database (http://paloma.isr.uc.pt/DataCollectionDB/handle/) of HANDLE project and constructed following the XML files approach and guidelines described on Deliverable 4 – Protocol for the corpus of sensed grasp and handling data. The description of the context and meaning of each XML element can also be found on that document. The toolbox provides tools to automatically integrate data from a full data acquisition session or data from individual devices (Polhemus Liberty, Videre stereo camera, Tekscan Grip system, Cyberglove II, instrumented Rubik cube). This toolbox also provides a parser to get the XML file corresponding to an ASCII file exported from the Tekscan Grip system software (Grip Research 6.33).

The version 2.0 of the toolbox has been developed on top of a *Java* library (*jParserToolbox*) which implements a XML SAX (*Simple API for XML*) parser for the XML datasets structure defined on *Deliverable 4*. It was possible to use this *Java* Library on the development of this *Matlab* toolbox because *Matlab* natively includes a *Java Virtual Machine*, so that it is possible to use the *Java* interpreter via Matlab commands.

This *importDatasetTB* toolbox implementation provides a faster parsing of the datasets than the *version 1.0* as well as a better memory management during that process avoiding some out-of-memory problems that occurred in the previous version during the parsing of large XML files. SAX parsing approach is a popular alternative to the Document Object Model (*DOM*) (approach used on the previous version of the toolbox and natively implemented on *Matlab* toolboxes).

2 Toolbox Installation

In order to install the toolbox, the described steps should be followed:

1.The file *importDatasetTB.rar* should be downloaded from the project database web platform and extracted on the computer.

2.The folder *importDatasetTB* should be moved to a definitive location. It is suggested that the folder *importDatasetTB* is placed on the *Toolbox* folder under the *MATLAB* installation folder.

3.** Add the *importDatasetTB* folder and sub-folder locations to the search path of *MATLAB*. Run the MATLAB software. Go to the menu *File → Set Path*. Select the option *Add with subfolders* and then indicate the *importDatasetTB* folder location. Press *OK* button then press *Save* button.

4.** Add the folder of the *importDatasetTB* toolbox to the *static Java path* list. Go to the Command Window of *Matlab* and type edit *classpath.txt*. Add the path (….. / importDatasetTB) of the *importDatasetTB* toolbox to the file.

5. Save the file *classpath.txt*.

6. Restart *Matlab*.

7.Type of the command window *javaclasspath* . The path of the *importDataTB* toolbox should be now on the *Satic Java Path list*.

** - These steps **may require** that you run Matlab software as **administrator** user.

Note: *The version 2.0 of the toolbox requires that the paths to the data files should be specified by the full paths.*

3 Data Integration

3.1 Full data acquisition session

The toolbox automatically identifies all the different devices (Polhemus Liberty, Videre Stereo, instrumented Rubik cube, Tekscan Grip System, Cyberglove II, used on the data acquisition. The user only has to specify the root.xml file path.

```
dataset importDatasetTB(path, idSessionType)
```

Table I – Description of the `importDatasetTB` function input and output parameters to integrate full dataset data in a *MATLAB* script

Input parameters	path	String indicating the full path to the location of the root.xml file of the data acquisition session.
	idSessionType	String which identifies the type of data that should be imported. In this case idSessionType='root'
Output parameters	Dataset	*MATLAB* structure containing the datasets of each of the devices used in the referred data acquisition session.

MATLAB script example:

Note: *The version 2.0 of the toolbox requires that the paths to the data files should be specified by the full paths.*

```
%Automatic integration of the full data acquisition XML example data
%The function automatically identifies which were the devices used on
%this data acquisition
dataset=importDatasetTB('./ExampleSession/root.xml','root');

%timestamp of the second sample of the third Polhemus Liberty sensor
dataset.datasetPolhemus(3).rawdata(2).timestamp

%X value of the second sample of the third Polhemus Liberty sensor
```

```
dataset.datasetPolhemus(3).rawdata(2).X

%Y value of the second sample of the third Polhemus Liberty sensor
dataset.datasetPolhemus(3).rawdata(2).Y


%Z value of the second sample of the third Polhemus Liberty sensor
dataset.datasetPolhemus(3).rawdata(2).Z

%Yaw value of the second sample of the third Polhemus Liberty sensor
dataset.datasetPolhemus(3).rawdata(2).YAW

%Pitch value of the second sample of the third Polhemus Liberty sensor
dataset.datasetPolhemus(3).rawdata(2).PITCH

%Roll of the second sample of the third Polhemus Liberty sensor
dataset.datasetPolhemus(3).rawdata(2).ROLL

%timestamp of the second sample of the Videre stereo camera data
dataset.datasetVidereStereo(1).rawdata(2).timestamp

%Display of the second sample of the right camera of the stereo camera
figure;
imshow(dataset.datasetVidereStereo(1).rawdata(2).image_right)

%Display of the second sample of the left camera of the Videre stereo
camera
figure;
imshow(dataset.datasetVidereStereo(1).rawdata(2).image_left)
```

3.2 Polhemus Liberty Dataset

```
dataset importDatasetTB(path, idSessionType)
```

Table II – Description of the `importDatasetTB` function input and output parameters to integrate only a Polhemus Liberty dataset in a *MATLAB* script

Input parameters	path	String indicating the path to the location of the data file of the Polhemus Liberty data acquisition session.
	idSessionType	String which identifies the type of data that should be imported. In this case idSessionType='polhemus'
Output parameters	Dataset	*MATLAB* structure containing the data of the Polhemus Liberty device dataset.

MATLAB script example:

Note: *The version 2.0 of the toolbox requires that the paths to the data files should be specified by the full paths.*

```
%Integration of a Polhemus Liberty dataset
datasetPolhemus=importDatasetTB('./ExampleSession/Polhemus/data_Polhem
us_S1.xml','polhemus');


%timestamp of the second sample of a Polhemus Liberty sensor
datasetPolhemus.rawdata(2).timestamp

%X value of the second sample of a Polhemus Liberty sensor
datasetPolhemus.rawdata(2).X

%Y value of the second sample of a Polhemus Liberty sensor
datasetPolhemus.rawdata(2).Y

%Z value of the second sample of a Polhemus Liberty sensor
datasetPolhemus.rawdata(2).Z

%Yaw value of the second sample of a Polhemus Liberty sensor
datasetPolhemus.rawdata(2).YAW

%Pitch value of the second sample of a Polhemus Liberty sensor
datasetPolhemus.rawdata(2).PITCH

%Roll of the second sample of a Polhemus Liberty sensor
datasetPolhemus.rawdata(2).ROLL
```

3.3 Videre Stereo camera

```
dataset importDatasetTB(path, idSessionType)
```

Table III– Description of the `importDatasetTB` function input and output parameters to integrate only a Videre stereo camera dataset in a *MATLAB* script

Input parameters	path	String indicating the path to the location of the data file of the Videre stereo camera data acquisition session.
	idSessionType	String which identifies the type of data that should be imported. In this case idSessionType='viderestereo'
Output parameters	Dataset	*MATLAB* structure containing the data of the Videre stereo camera device dataset.

MATLAB script example:

Note: *The version 2.0 of the toolbox requires that the paths to the data files should be specified by the full paths.*

```
%Integration of a Videre stereo camera dataset
datasetVidereStereo=importDatasetTB('./ExampleSession/StereoCamera/dat
a_Videre.xml','viderestereo');

%timestamp of the second sample of the Videre stereo camera data
datasetVidereStereo.rawdata(2).timestamp

%Display of the second sample of the right camera of the Videre stereo
%camera
figure;
imshow(datasetVidereStereo.rawdata(2).image_right)

%Display of the second sample of the left camera of the Videre stereo
%camera
figure;
imshow(datasetVidereStereo.rawdata(2).image_left)
```

3.4 Tekscan Grip System

In the case of Tekscan Grip system data, the *importDatasetTB* toolbox provides three different functionalities: direct integration of an ASCII data file exported by the Tekscan Grip system software in *MATLAB*, conversion of the ASCII data file to the correspondent XML file or integration of the data contained in the XML file in the *MATLAB* software.

```
dataset importDatasetTB(path, idSessionType, msStarTime,
hzFrequency)
```

Table IV – Description of the `importDatasetTB` function input and output parameters to convert a ASCII file exported by Tekscan Grip system software to a XML file

Input parameters	path	String indicating the path to the location of the ASCII file exported by the Tekscan Grip system software.
	idSessionType	String which identifies the type of operation that should be performed. In this case idSessionType=' tekscantactileRAW2XML'
	msStartTime	Integer number indicating the millisecond timestamp of the first sample.
	hzFrequency	Integer number indicating the data sampling frequency in Hertz.
Output parameters	dataset	*MATLAB* integer. 1-XML file successfully created 0-XML file not created The XML file is created in the current workspace directory of MATLAB software with the name 'output.xml'.

```
dataset importDatasetTB(path, idSessionType)
```

Table V – Description of the `importDatasetTB` function input and output and output parameters to integrate only a TekScan Grip system dataset in a *MATLAB* script from a XML file

Input parameters	path	String indicating the path to the location of the data XML of Tekscan Grip System data acquisition session.
	idSessionType	String which identifies the type of operation that should be performed. In this case idSessionType=' tekscantactileXML2MATLAB'
Output parameters	dataset	*MATLAB* structure containing the data of the Tekscan grip system device dataset.

```
dataset importDatasetTB(path, idSessionType, msStarTime, hzFrequency)
```

Table VI – Description of the `importDatasetTB` function input and output parameters to convert a ASCII file exported by Tekscan Grip system software directly to MATLAB

Input parameters	path	String indicating the path to the location of the ASCII file exported by the Tekscan Grip system software.
	idSessionType	String which identifies the type of operation that should be performed. In this case idSessionType=' tekscantactileRAW2MATLAB'
	msStartTime	Integer number indicating the millisecond timestamp of the first sample.
	hzFrequency	Integer number indicating the data sampling frequency in Hertz.
Output parameters	dataset	*MATLAB* structure containing the data of the Tekscan grip system device dataset.

MATLAB script examples:

Note: *The version 2.0 of the toolbox requires that the paths to the data files should be specified by the full paths.*

```
%Conversion of a ASCII data file exported by the Tekscan Grip system
software to the correspondent XML file. Ms timestamp of the first
sample, 1000 ms. Sampling rate, 500Hz.
datasetTekscan=importDatasetTB('trial01_AllPadsRandomOrder.asf','teksc
antactileRAW2XML',1000,500);


%Integration of the data contained in a XML file corresponding to a
Tekscan Grip system dataset in the MATLAB software
datasetTekscan=importDatasetTB('output.xml','tekscantactileXML2MATLAB'
);

%timestamp of the tenth sample of the Tekscan Grip system dataset
datasetTekscan.rawdata(10).timestamp

%Value of the twelveth sensing element of the tDistal pad of the tenth
%sample of the Tekscan Grip system dataset
datasetTekscan.rawdata(10).tDistal(12)

%Value of the eleventh sensing element of the tProximal pad of the
tenth
%sample of the Tekscan Grip system dataset
datasetTekscan.rawdata(10).tProximal(11)

%Value of the fifth sensing element of the iDistal pad of the tenth
%sample of the Tekscan Grip system dataset
datasetTekscan.rawdata(10).iDistal(5)

%Value of the sixth sensing element of the iMedial pad of the tenth
%sample of the Tekscan Grip system dataset
datasetTekscan.rawdata(10).iMedial(6)

%Value of the twelveth sensing element of the iProximal pad of the
tenth
%sample of the Tekscan Grip system dataset
datasetTekscan.rawdata(10).iProximal(12)

%Value of the twelveth sensing element of the mDistal pad of the tenth
%sample of the Tekscan Grip system dataset
datasetTekscan.rawdata(10).mDistal(12)

%Value of the twelveth sensing element of the mMedial pad of the tenth
%sample of the Tekscan Grip system dataset
datasetTekscan.rawdata(10).mMedial(12)

%Value of the twelveth sensing element of the mProximal pad of the
tenth
%sample of the Tekscan Grip system dataset
datasetTekscan.rawdata(10).mProximal(12)

%Value of the twelveth sensing element of the rDistal pad of the tenth
%sample of the Tekscan Grip system dataset
```

```
datasetTekscan.rawdata(10).rDistal(12)

%Value of the twelveth sensing element of the rMedial pad of the tenth
%sample of the Tekscan Grip system dataset
datasetTekscan.rawdata(10).rMedial(12)

%Value of the twelveth sensing element of the rProximal pad of the
tenth
%sample of the Tekscan Grip system dataset
datasetTekscan.rawdata(10).rProximal(12)



%Value of the twelveth sensing element of the lDistal pad of the tenth
%sample of the Tekscan Grip system dataset
datasetTekscan.rawdata(10).lDistal(12)

%Value of the twelveth sensing element of the lMedial pad of the tenth
%sample of the Tekscan Grip system dataset
datasetTekscan.rawdata(10).lMedial(12)

%Value of the twelveth sensing element of the lProximal pad of the
tenth
%sample of the Tekscan Grip system dataset
datasetTekscan.rawdata(10).lProximal(12)

%Value of the twelveth sensing element of the pFingers pad of the
tenth
%sample of the Tekscan Grip system dataset
datasetTekscan.rawdata(10).pFingers(12)

%Value of the twelveth sensing element of the pInternal pad of the
tenth
%sample of the Tekscan Grip system dataset
datasetTekscan.rawdata(10).pInternal(12)

%Value of the twelveth sensing element of the pExternal pad of the
tenth
%sample of the Tekscan Grip system dataset
datasetTekscan.rawdata(10).pExternal(12)

%Direct integration of the data contained on a ASCII file exported by
%Tekscan Grip system to MATLAB. The structure of datasetTekscan is
similiar
%to the one described before in this example.
datasetTekscan=importDatasetTB('trial01_AllPadsRandomOrder.asf','teksc
antactileRAW2MATLAB',1000,500);
```

3.5 Cyberglove II

```
dataset importDatasetTB(path, idSessionType)
```

Table VII – Description of the `importDatasetTB` function input and output parameters to integrate only a Cyberglove II dataset in a *MATLAB* script

Input parameters	path	String indicating the path to the location of the data file of the Cyberglove II data acquisition session.
	idSessionType	String which identifies the type of data that should be imported. In this case idSessionType='cybergloveii'
Output parameters	Dataset	*MATLAB* structure containing the data of the Polhemus Liberty device dataset.

MATLAB script example:

Note: *The version 2.0 of the toolbox requires that the paths to the data files should be specified by the full paths.*

```
%Integration of a Cyberglove II dataset
datasetCybergloveii=importDatasetTB('Cyberbloveii.xml','cybergloveii')
;

%timestamp of the second sample of the Cyberglove II device
datasetCybergloveii.rawdata(2).timestamp

%Data of the tTMJ flexure sensor in the second sample of the
Cyberglove II device
datasetPolhemus.rawdata(2).tTMJ

%Data of the iMPJ flexure sensor in the second sample of the
Cyberglove II device
datasetPolhemus.rawdata(2).iMPJ

%Data of the mPIJ flexure sensor in the second sample of the
Cyberglove II device
datasetPolhemus.rawdata(2).mPIJ

%Data of the rmAbd flexure sensor in the second sample of the
Cyberglove II device
datasetPolhemus.rawdata(2).rmAbd


%The data of the remaining flexure sensor can be accessed using the
%names of the data elements referred on the XML file, following the
%logic described on these examples
```

3.6 Instrumented Rubik cube

```
dataset importDatasetTB(path, idSessionType)
```

Table VIII – Description of the `importDatasetTB` function input and output parameters to integrate only an instrumented Rubik cube dataset in a *MATLAB* script

Input parameters	path	String indicating the path to the location of the data file of the instrumented Rubik cube data acquisition session.
	idSessionType	String which identifies the type of data that should be imported. In this case idSessionType='rubikcube'
Output parameters	Dataset	*MATLAB* structure containing the data of the instrumented Rubik cube device dataset.

MATLAB script example:

Note: *The version 2.0 of the toolbox requires that the paths to the data files should be specified by the full paths.*

```
%Integration of a instrumented Rubik cube dataset
datasetRubikCube=importDatasetTB('instrumentedRubikCube.xml','rubikcub
e');

%timestamp of the second sample of the instrumented Rubik cube device
datasetRubikCube.rawdata(2).timestamp

%Data of the fifth sensing element of the red side of the cube, on the
%second sample of the instrumented Rubik cube dataset
datasetRubikCube.rawdata(2).R(5)

%Data of the X axis componente of the acceleration of the red side of
%the cube, on the second sample of the instrumented Rubik cube dataset
datasetRubikCube.rawdata(2).Rx

%Data of the Y axis componente of the acceleration of the red side of
%the cube, on the second sample of the instrumented Rubik cube dataset
datasetRubikCube.rawdata(2).Ry

%Data of the Z axis componente of the acceleration of the red side of
%the cube, on the second sample of the instrumented Rubik cube dataset
datasetRubikCube.rawdata(2).Rz

%Data of the second sensing element of the green side of the cube, on
%the second sample of the instrumented Rubik cube dataset
datasetRubikCube.rawdata(2).G(2)
```

```
%Data of the fourth sensing element of the white side of the cube, on
%the second sample of the instrumented Rubik cube dataset
datasetRubikCube.rawdata(2).W(4)

%Data of the third sensing element of the orange side of the cube, on
%the second sample of the instrumented Rubik cube dataset
datasetRubikCube.rawdata(2).O(3)

%Data of the eight sensing element of the blue side of the cube, on
%the second sample of the instrumented Rubik cube dataset
datasetRubikCube.rawdata(2).B(8)

%Data of the first sensing element of the yellow side of the cube, on
%the second sample of the instrumented Rubik cube dataset
datasetRubikCube.rawdata(2).Y(1)
```

3.7 Data integration performance test

The data integration performance of the importDatasetTB toolbox has been tested in the dataset Instrumented Rubik Cube Displacement - Trial 01, on a Intel Core Duo CPU P8700 @2.53GHz @2.53GHz, 4GB RAM memory, Windows Vista 32-bit.

Table IX – Data integration performance test results

Dataset	Number Data Elements	Data Element Description	Time (seconds)
Polhemus	114	timestamp 6 data elements	0.054
Videre Stereo Camera	124	timestamp 2 images (320x240)	1.095
Unibrain Monocular Camera	146	timestamp 1 image (640x480)	2.207
Cyberglove	199	timestamp 22 data elements	0.080
Tekscan	3743	timestamp 361 data elements	30.889
Instrumented Rubik Cube	45136	Timestamp 72 data elements	29.794

4 *jParserToolbox* Library

Although the main objective of this document is to describe the functionalities of the *importDatasetTB* toolbox. during the development of the version 2.0, a *Java* library (*jParserToolbox* package) implement a SAX XML parsing approach of the files of the datasets has been developed. The following tables provide a brief description of the classes, parameters and methods of that *Java* library, in order to support the integration of the web platform database datasets on *Java* applications if required.

Some examples of the utilization of this *jParserToolbox* Library can be found on the *Matlab* implementation functions of the *importDatasetTB* toolbox.

4.1 *root.xml* file

Table X – `public class RootDeviceFrame` main elements

`public RootDeviceFrame()`	Class constructor.
`private String DeviceAliasName`	String with the identification of the type of the device.
`private String DataPath`	Path to the device element data XML
`public void setDeviceAliasName(String)`	Assign `String` to `DeviceAliasName`.
`public void setDataPath(String)`	Assign `String` to `DataPath`
`public String getDeviceAliasName()`	Return the `String` content of `DeviceAliasName`
`public String getDataPath()`	Return the `String` content of `DataPath`.

Table XI – `public class jRootParser` main elements

`Public jRootParser(String)`	Class constructor. Receives the full path to the file to be parsed.
`Public RootDeviceFrame[] parseFile()`	Returns an array of `RootDeviceFrame` objects .

4.2 *Polhemus Liberty* dataset

Table XII – `public class PolhemusFrame` main elements

`public PolhemusFrame()`	Class constructor.
`private double timestamp`	Data sample timestamp
`private double[] position`	3 elements position data array. 0-X, 1-Y, 2-Z
`private double[] orientation`	3 elements orientation data array. 0-Yaw, 1-Picth, 2-Roll
`public void setTimestamp(double value)`	Assign `value` to `timestamp`.
`public double getTimestamp()`	Return the value of `timestamp`.
`public void setPosition(int index, double value)`	Assign `value` to the `position` array `index` element
`public double[] getPosition()`	Return the `position` array.
`public void setOrientation(int index, double value)`	Assign `value` to the `orientation` array `index` element.
`public double[] getOrientation()`	Return the `orientation` array.

Table XIII – `public class jPolhemusParser` main elements

`Public jPolhemusParser(String)`	Class constructor. Receives the full path to the file to be parsed.
`Public PolhemusFrame[] parseFile()`	Returns an array of `PolhemusFrame` objects .

4.3 Videre Stereo Camera dataset

Table XIV – `public class VidereFrame` main elements

`public VidereFrame()`	Class constructor.
`private double timestamp`	Data sample timestamp.
`private String pathRightImage`	Data sample path to the right image
`private String pathLeftImage`	Data sample path to the left image
`public void setTimestamp(double value)`	Assign `value` to `timestamp`
`public double getTimestamp()`	Return the value of `timestamp`
`public void setPathRight(String path)`	Assign `path` to `pathRightImage`
`public void setPathLeft(String path)`	Assign `path` to `pathLeftImage`
`public String getPathRight()`	Return the value of `pathRightImage`
`public String getPathLeft()`	Return the value of `pathLeftImage`

Table XV – `public class jVidereParser` main elements

`Public jVidereParser(String)`	Class constructor. Receives the full path to the file to be parsed.
`Public VidereFrame[] parseFile()`	Returns an array of `VidereFrame` objects.

4.4 Unibrain Monocular Camera dataset

Table XVI– `public class UnibrainFrame` main elements

`public Unibrain()`	Class constructor.
`private double timestamp`	Data sample timestamp
`private String pathImage`	Data sample path to the image
`public void setTimestamp(double value)`	Assign `value` to `timestamp`
`public double getTimestamp()`	Return the value of `timestamp`
`public void setPath(String path)`	Assign `path` to `pathImage`
`public String getPath()`	Return the value of `pathImage`

Table XVII – `public class jUnibrainParser` main elements

`Public jUnibrainParser(String)`	Class constructor. Receives the full path to the file to be parsed.
`Public UnibrainFrame[] parseFile()`	Returns an array of `UnibrainFrame` objects .

4.5 Cyberglove dataset

Table XVIII– `public class CybergloveFrame` main elements

`public CybergloveFrame()`	Class constructor.
`private double timestamp`	Data sample timestamp
`private double[] data`	Data sample 22 flexure data elements array.
`public void setTimestamp(double value)`	Assign `value` to `timestamp`
`public double getTimestamp()`	Return the value of `timestamp`
`public void setData(int position,double value)`	Assign `value` to the `position` element of `data`.
`public double[] getData()`	Return the array `data`.

Table XIX – `public class jCybergloveParser` main elements

`Public jCybergloveParser(String)`	Class constructor. Receives the full path to the file to be parsed.
`Public CybergloveFrame[] parseFile()`	Returns an array of `CybergloveFrame` objects .

4.6 *Instrumented Rubik cube* dataset

Table XX– `public class RubikFrame` main elements

`public RubikFrame()`	Class constructor.
`private double timestamp`	Data sample timestamp.
`private int[] data`***X*** ***X***`=Green,White,Orange, Blue, Red, Yellow`	Data sample array (12 elements) of the ***X*** side of the instrumented cube.
`public void setTimestamp(double value)`	Assign `value` to `timestamp`
`public double getTimestamp()`	Return the value of `timestamp`
`public void setData`***X***`(int position,int value)` ***X***`=Green,White,Orange, Blue, Red, Yellow`	Assign `value` to the `position` element of `data`***X***
`public int[] getData`***X***`()` ***X***`=Green,White,Orange, Blue, Red, Yellow`	Return the `data`***X*** array .

Table XXI – `public class jRubikParser` main elements

`Public jRubikParser(String)`	Class constructor. Receives the full path to the file to be parsed.
`Public RubikFrame[] parseFile()`	Returns an array of `RubikFrame` objects.

4.7 *Tekscan Tactile sensing* dataset

Table XXII– public class TekscanFrame main elements

public TekscanFrame()	Class constructor.
private double timestamp	Data sample timestamp.
private int[] ***X***Distal ***X***=t,i,m,r,l	Data sample array of the ***X***Distal region of the Tekscan tactile array.
private int[] ***X***Medial ***X***=i,m,r,l	Data sample array of the ***X***Medial region of the Tekscan tactile array.
private int[] ***X***Proximal ***X***=i,m,r,l	Data sample array of the ***X***Proximal region of the Tekscan tactile array.
private int[] p***X*** ***X***=Fingers,Internal,External	Data sample array of the p***X*** region of the Tekscan tactile array.
public void setTimestamp(double value)	Assign value to timestamp
public double getTimestamp()	Return timestamp value
public void setData***X***Distal(int position,int value) ***X***=t,i,m,r,l	Assign value to the position element of ***X***Distal
public void setData***X***Medial(int position,int value) ***X***=i,m,r,l	Assign value to the position element of ***X***Medial
public void setData***X***Proximal(int position,int value) ***X***=t,i,m,r,l	Assign value to the position element of ***X***Proximal
public void setDatap***X***(int position,int value) ***X***=Fingers,Internal,External	Assign value to the position element of p***X***
public int[] getData***X***Distal() ***X***=t,i,m,r,l	Return the ***X***Distal array.
public int[] getData***X***Medial() ***X***=i,m,r,l	Return the ***X***Medial array.
public int[] getData***X***Proximal() ***X***=t,i,m,r,l	Return the ***X***Proximal array.
public int[] getDatap***X***() ***X***=Fingers,Internal,External	Return the p***X*** array.

Table XXIII – `public class jTekscanParser` main elements

`Public jTekscanParser(String)`	Class constructor. Receives the full path to the file to be parsed.
`Public TekscanFrame[] parseFile()`	Returns an array of `TekscanFrame` objects.