**Fall Detection Documentation**

The current document presents a description of the dll’s public variables and functions which are available to the user.

**FallDetect.h**

**Class: FallDetect**

Public Variables:

bool m\_bVectorsAreUpdated:

Bool variable that informs us whether the vectors of movement are updated.

std::ofstream m\_ofs\_data:

Stream for printing the values of R, Δθ, Δφ of a recording.

std::ofstream m\_ofs\_timestamp:

Stream for printing the corresponding timestamps, of R, Δθ, Δφ, of a recording.

Public Functions:

**FallDetect**();

Description: Creates an object of class **FallDetect**.

~**FallDetect**();

Description: Destroys the object of class **FallDetect**.

bool **detectFall**(std::string storedMLPPath)

Description: Function that returns true if a fall happens, false otherwise. The function checks the 120 last values returned by the neural prediction and if 60, or more, of them exceeds the threshold returns true, otherwise false.

**Parameters: 1. storedMLPPath** –path where the signature (multilayerPerceptron.dat) of the neural network is stored.

bool **checkUsersStatus**(double xVal, double yVal, double zVal, Timestamp tmpStmp)

Description: Function that updates the x, y, z and the current timestamp to the passed values. Calculates the R, Δθ, Δφ, SMA and computes through the doFIS(m\_current\_sma, m\_current\_deltaTheta, m\_current\_deltaPhi) (Fuzzy Inference System) the current move and pushes, the smoothed by an average filter, value to a queue.

**Parameters: 1. xVal** – X accelerometer value

**2. yVal** –Y accelerometer value

**3. zVal** – Z accelerometer value

**4. tmpStmp** – The corresponding timestamp for xVal, yVal, zVal.

std::deque<double> **calcAll\_Offline**(

std::deque<std::deque<double>>inputData\_deq, std::deque<std::deque<double>> inputTime\_deq )

Description: Function that takes as arguments the inputData and the inputTime and returns a deque of double produced by the Fuzzy Inference System, which can be used as data for the processedAngles.txt (which can be used for training or testing the neural network).

**Parameters: 1. inputData\_deq** –A 2-dimensional deque of double which includes the values of R, Δθ, Δφ.

**2. inputTime\_deq** –A 2-dimensional deque of double which includes the corresponding timestamps of the values of inputData\_deq.

fl::flScalar **doFIS**(fl::flScalar in\_sma, fl::flScalar in\_deltaTheta, fl::flScalar in\_deltaPhi)

Description: Responsible function for computing the movement of a timestamp. We pass arguments to the function the SMA, Δθ, and Δφ and function maps the values to a movement using a Fuzzy Inference System (FIS).

**Parameters: 1. in\_sma** –The value of SMA(Signal Manitude Area).

**2. in\_deltaTheta** –The value of Δθ.

**3. in\_deltaPhi** – The value of Δφ.

std::deque<std::deque<double> > **readDataFromFile\_1Ddeq**(

std::string filenamePath, bool bPopBackLastCol=true)

Description: Function that takes as argument a file (ex allTrainingDataResults.txt) and returns the contents of the file as a 1-dimensional deque.

**Parameters: 1. filenamePath** –The path of the input file.

**2. bPopBackLastCol** – Bool variable setted true by default.

std::deque<std::deque<double> > **readDataFromFile\_2Ddeq**(

std::string filenamePath, bool bPopBackLastCol=true)

Description: Function that takes as argument a file (ex angles.txt, index.ini) and return the contents of the file as a 2-dimensional deque

**Parameters: 1. filenamePath** –The path of the input file.

**2. bPopBackLastCol** – Bool variable setted true by default.

std::deque< std::deque<double> > **convertRawsToColumns**(

std::deque< std::deque<double> > inputMatrix\_deq)

Description: Function that converts the Raws of a matrix to columns.

**Parameters: 1. inputMatrix\_deq** – a 2-dimensional matrix.

void **storeAngles2D\_deq**(

std::deque< std::deque<double>> input\_vec,

std::string outputPath\_cstr,

bool bInsertIndex=false)

Description: Stores the values of a 2-dimensional deque at a file, with path the outputPath\_cstr.

**Parameters: 1. input\_vec** – a one dimensional deque

**2. outputPath\_cstr** – the path to the printing file

**3. bInsertIndex** – bool variable, false by default

void **storeAngles1D\_deq**(

std::deque<double> input\_vec,

std::string outputPath\_cstr,

bool bInsertIndex=false)

Description: Stores the values of a 1-dimensional deque at a file, with path the outputPath\_cstr.

**Parameters: 1. input\_vec** – a one dimensional deque

**2. outputPath\_cstr** – the path to the printing file

**3. bInsertIndex** – bool variable, false by default

bool **isMovementUpdated** ()

Description: Returns true if the deque of movement, produced by the Fuzzy Inference System (FIS), was updated, false otherwise.

std::deque<double> **getMovement\_deq**( )

Description: Function that returns a deque with the current movement (as list of doubles), produced by the Fuzzy Inference System (FIS).

double **getR** ()

Description: Returns the current value of r.

double **getDeltaPhi** ()

Description: Returns the current value of Δφ.

double **getDeltaTheta** ()

Description: Returns the current value of Δθ.

Timestamp **getTimestamp()**

Description: Returns the current value of timestamp.

**MLPClass.h**

**Class: MLPClass**

Public Functions:

**MLPClass**();

Description: Creates an object of class **MLPClass**.

~ **MLPClass** ();

Description: Destroys the object of class **MLPClass**.

bool checkPattern(std::deque<double> inPattern, double &mlpVal, double threshold)

Description: Runs the neural network prediction for the given data (inPattern) and gives a value to mlpVal. If mlpVal >= threshold returns true, else returns false. We use checkPattern to detect possible falls.

**Parameters: 1. inPattern** – a deque with the movements produced by doFIS

**2.** &**mlpVal** – a reference to mlpVal. We pass the value of the neural prediction to mlpVal

**3. threshold** – a double value

int performMLPTraining(std::string inputDataFilePath, std::string inputResultDataFilePath, std::string logFilePath)

Description: Trains the MLP and saves the results to the multilayerperceptron.dat

**Parameters: 1. inputDataFilePath** – path to allTrainingData.txt, which contains the movement values for every recording, produced by the FIS. Every column of the file represents a recording.

**2.** **inputResultDataFilePath** – path to allTrainingDataResults.txt, which contains a list of ones and zeros. Every value at the allTrainingDataResults.txt corresponds to a column of allTrainingData.txt. If the recording of a column (of allTrainingData.txt ) describes a fall we annotate 1 to the corresponding row (of allTrainingDataResults.txt), otherwise we annotate 0.

**3. logFilePath** – path where the produced file will be stored.

bool loadNewMLP(std::string storedMLPPath)

Description: Loads the MLP from multilayerperceptron.dat

**Parameters: 1. storedMLPPath** – path where the multilayerperceptron.dat file is stored.