

Index

| | |
|--|----|
| Introduction..... | 2 |
| Goals..... | 3 |
| Materials and Methods..... | 4 |
| Inertial Measurement Unit (IMU)..... | 4 |
| MTx measurement unit..... | 6 |
| Hidden Markov Model (HMM)..... | 8 |
| HMM algorithms..... | 9 |
| Definition of a Hidden Markov Model (HMM)..... | 9 |
| Forward Algorithm | 10 |
| Backward Algorithm | 11 |
| Posterior decoding | 11 |
| Viterbi algorithm | 12 |
| Baum–Welch algorithm..... | 12 |
| K-means clustering..... | 14 |
| Overview of the algorithm used..... | 14 |
| Data..... | 16 |
| Parameters..... | 18 |
| Size of the array of data (data_width)..... | 18 |
| Data dimensions used to generate the observable symbols (D)..... | 19 |
| Number of observable symbols in the alphabet (N)..... | 19 |
| Number of internal states of the model (M)..... | 20 |
| Degree of play in the left-to-right HMM transition matrix (LR)..... | 20 |
| Results..... | 21 |
| Description of the experiments..... | 21 |
| Performance of the resulting models..... | 22 |
| Successful rejection of different movements..... | 24 |
| Influence of the number of internal states..... | 25 |
| Influence of the number of output symbols (clusters)..... | 27 |
| Influence of the degree of play in the left-to-right transition matrix (LR)..... | 29 |
| Repeatability of the results..... | 30 |
| Conclusions..... | 31 |
| Performance of the resulting models..... | 31 |
| Rejection of other movements..... | 32 |
| Defining a model that would be valid for different users..... | 32 |
| Bibliography and links..... | 33 |
| Measurement of mobility-related activities:..... | 33 |
| Inertia Mobile Units (IMUs)..... | 33 |
| Hidden Markov Models..... | 34 |
| Appendix A: Representation of data for every person and kind of movement..... | 35 |
| Appendix B: Data from every person while walking..... | 46 |
| Appendix C: Clustered data from every person while walking..... | 48 |
| Appendix D: Results of HMM models for identifying walk..... | 50 |
| Rejection of other movements..... | 51 |

| | |
|--------------------------------|----|
| Averaged performance..... | 52 |
| Appendix E: Source Code..... | 53 |
| test.m..... | 53 |
| get_data.m..... | 55 |
| get_point_centroids.m..... | 56 |
| get_point_clusters.m..... | 56 |
| kmeans.m..... | 56 |
| prior_transition_matrix.m..... | 58 |
| dhmm_numeric.m..... | 59 |
| pr_hmm.m..... | 62 |
| csum.m..... | 62 |
| cdiv.m..... | 63 |
| rsum.m..... | 63 |
| rdiv.m..... | 63 |

Introduction

Mobility is essential in human being. Due to the quick evolution of technology in the last decades, the whole lifestyle and work environment of human beings has totally change, thus decreasing considerably the average physical activity carried out by people. All this implies problems and illnesses such as obesity, cardiovascular disorders, diabetes, etc.

For that reason, governments and different organizations have taken physical activity as one of the most important health indicators of a country, and a number of initiatives have been developed to measure and classify this activity on a daily basis.

Among the methods developed for that, there are the traditional and somewhat imprecise and subjective ones such as questionnaires and activity logs. However, in recent times we have seen intents of incorporating technological devices for quantifying this physical activity, mostly based on inertial sensors, because of the reduction in cost and size of these devices.

Commercially, we can find mostly two types of devices:

Pedometers, consisting in a single motion sensor and are used to measure the number of steps. They can indirectly measure distance, speed and walking cadence.

More complex devices, which allow the use of several sensors of movement placed in different parts of the body. These multisensor systems, more expensive, technologically advanced and complex to use, have the advantage of being able to provide better performance in quantifying physical activity.

The literature on the measurement of physical activity is very heterogeneous. Each researcher or research group has focused on a set of specific movements using a concrete device, and has applied wide array processing algorithms. This heterogeneity makes it really difficult to compare them objectively.

Inertial sensors seem to focus the main interest, mainly accelerometers, but also other inertial sensors such as gyroscopes, or even non-inertial ones such as heart rate measuring devices or microphones.

Similarly, the algorithms used for the detection of physical activity are very diverse. Some studies focus on the development of classification systems based on thresholds or heuristic classifiers. Other use methods such as decision trees, k-neighbors, Bayesian classifiers, support vector machines (SVM), artificial neural networks, etc. So far, no method is known to be the most appropriate to solve the problem.

In the best cases, the methods used imply that the system has to be adapted or trained to adjust to each specific individual, so they have to be subjected to various tests prior to their use in their daily lives.

Finally, it has to be noticed that an increasing number of people are carrying inertial sensors with them. As technology is becoming increasingly accessible, Inertia Measurement Units (IMUs) are becoming more widely widespread, to the point that at the moment there are many people that carry one all the time on them, maybe without them being conscious of it, as there is one already integrated in many smart mobile phones. This makes it very attractive to try use them for measuring activity.

Goals

The purpose of this work is to find out whether a system based on a conventional Hidden Markov Model (HMM) is able to detect the action that the person is carrying out, and be able to differentiate it from other similar actions. Thus, at some point in the future, a system can be developed that checks the data read from the sensors, compares it with some different models that represent a bunch of different actions, and find out if any of them correspond to what the person is actually doing.

The main idea is to be able to develop a system that knows whether a person is carrying out a certain movement type at a moment. A HMM model has to be developed for each type of movement, so in the end we will have a battery of models, each of them detecting a different kind. The models have to be previously trained with each person, although at some point in the future it would be nice to find out if some normalization of the data can be used to make it user-independent.

Some cases, like walking, have been thoroughly studied, and every stage of each step carefully described. This can only be done for a small set of movements, so the idea would be to make it as automatic as possible, so that the system doesn't need to know anything beforehand. None of the experiments carried out for this work have taken any of that knowledge into account.

Materials and Methods

Inertial Measurement Unit (IMU)

An IMU (Inertial Measurement Unit) is an electronic device that uses a combination of accelerometers, gyroscopes and, sometimes, magnetometers, to measure and log the velocity, orientation, gravitational forces and magnetic fields (digital compass). IMUs are the main components of inertial navigation systems, which are used in aircraft, spacecraft, watercraft, guided missiles, etc.

IMUs work by detecting accelerations in three orthogonal directions, and changes in the rotational attributes like pitch, roll and yaw, as well as changes in the magnetic field along the three orthogonal directions already mentioned. The inclusion of magnetometers allow for a better performance in dynamic orientation calculation in attitude and heading reference systems.

Accelerometers can be as simple as a mass attached to a spring that measures its deflection. That way, the sum of forces acting over that mass can be measured, including gravitation. To improve the dynamic interval and linearity, and also to reduce hysteresis, a control loop can be applied, keeping the mass close to its nominal position. Using 3 (or more) accelerometers we can obtain a 3D specific force measurement. Good commercial accelerometers have an accuracy in the order of 50 μg .

Gyroscopes (or simply gyros) measure the angular velocity relative to the inertial space. That can be achieved by different means:

- **Mechanical gyros:** A spinning wheel will resist any change in its angular momentum vector relative to inertial space. The idea of the mechanical gyro is to isolate the wheel from the vehicle angular movements by means of gimbals, and then to output the gimbal positions.
- **The Sagnac-effect:** The inertial characteristics of light can also be utilized, by letting two beams of light travel in a loop in opposite directions. If the loop rotates clockwise, the clockwise beam must travel a longer distance before finishing the loop. The opposite is true for the counter-clockwise beam. Combining the two rays in a detector, an interference pattern is formed, which will depend on the angular velocity. The loop can be implemented with 3 or 4 mirrors (Ring Laser Gyro), or with optical fibers (Fiber Optic Gyro).
- **The Coriolis-effect:** Assume a mass that is vibrating in the radial direction of a rotating system. Due to the Coriolis force working perpendicular to the original vibrating direction, a new vibration will take place in this direction. The amplitude of this new vibration is a function of the angular velocity. MEMS gyros (Micro-Electro-Mechanical Systems), "tuning fork" and "wineglass" gyros are utilizing this principle. Coriolis-based gyros are typically cheaper and less accurate than mechanical, ring laser or fiber optic gyros.

A categorization of the IMU technology and IMU performance can be shown in the following table:

| Class | Position performance | Gyro technology | Accelerometer technology | Gyro bias | Accelerometer bias |
|-------------------------|-------------------------------|--------------------------|---|------------------------------|--------------------|
| <i>Military grade</i> | 1 nmi / 24 h | ESG, RLG, FOG | Servo accelerometer | $< 0.005^\circ / \text{h}$ | $< 30 \mu\text{g}$ |
| <i>Navigation grade</i> | 1 nmi / h | RLG, FOG | Servo accelerometer, Vibrating beam | $0.01^\circ / \text{h}$ | 50 μg |
| <i>Tactical grade</i> | $> 10 \text{ nmi} / \text{h}$ | RLG, FOG | Servo accelerometer, Vibrating beam, MEMS | $1^\circ / \text{h}$ | 1 mg |
| <i>AHRS</i> | NA | MEMS, RLG, FOG, Coriolis | MEMS | $1 - 10^\circ / \text{h}$ | 1 mg |
| <i>Control System</i> | NA | Coriolis | MEMS | $10 - 1000^\circ / \text{h}$ | 10 mg |

As mobile phone technology is increasingly developing, we're seeing more smart phones with inertial measurement capabilities. Smart phones usually have a 3-axis accelerometer, a 3-axis gyroscope and a 3-axis magnetometer. Each of them has advantages and disadvantages: The accelerometers are very sensitive to inertial noise, the gyroscopes are fastest and the most accurate, but their main problem is the drift, and for the magnetometers, while they don't have drift and they aren't sensitive to inertial forces, they are too slow and very sensitive to electromagnetic noise.

Android accelerometers are digital. They sample acceleration using the same number of "buckets" (lets say there are 256 buckets and the accelerometer is capable of sensing from -2g to +2g). This means that the output is quantized, and will be jumping around some set of values.

Usually a Kalman filter is applied over the raw data of the IMU, to lower the errors. A Kalman filter is a recursive algorithm for estimating states in a system. Two sorts of information are utilized in this kind of filters:

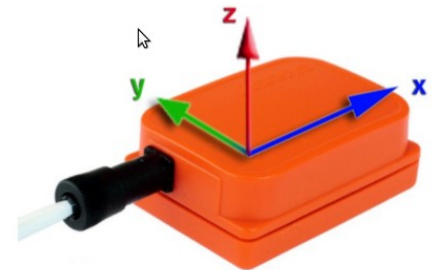
- Measurements from relevant sensors
- A mathematical model of the system (describing how the different states depend on each other, and how the measurements depend on the states)

If an IMU is placed onto a human body, and moves along with it, it can be used to keep track of the changes in position and orientation of that person. The IMU, represented as a rigid body, has 6 degrees of freedom.

MTx measurement unit

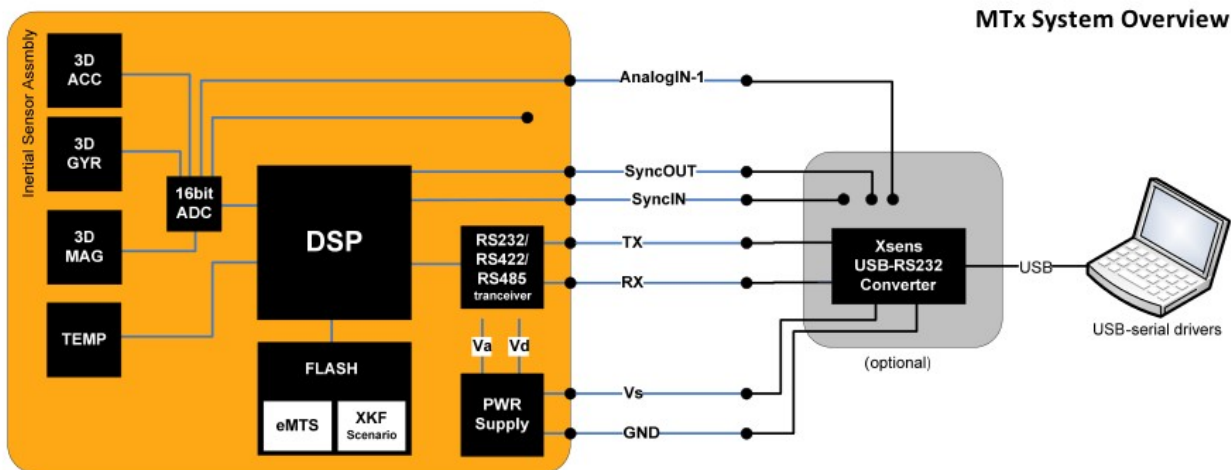
The MTx measurement unit is a small and accurate 3DOF Orientation Tracker, developed by Xsens. It provides drift-free 3D orientation as well as kinematic data: 3D acceleration, 3D rate of turn and 3D earth-magnetic field. The MTx uses 3 rate gyroscopes to track rapidly changing orientations in 3D and it measures the directions of gravity and magnetic north to provide a stable reference. The systems real-time algorithm fuses all sensor information to calculate accurate 3D orientation, with a highly dynamic response, which remains stable over time.

The orientation of the MTx is computed by an Xsens Kalman Filter for 3 degrees-of-freedom (3DoF) orientation (XKF-3). XKF-3 uses signals of the rate gyroscopes, accelerometers and magnetometers to compute a statistical optimal 3D orientation estimate of high accuracy with no drift for both static and dynamic movements.



MTx with sensor-fixed co-ordinate system overlaid

The design of the XKF-3 algorithm can be explained as a sensor fusion algorithm where the measurement of gravity (by the 3D accelerometers) and Earth magnetic north (by the 3D magnetometers) compensate for otherwise slowly, but unlimited, increasing (drift) errors from the integration of rate of turn data (angular velocity from the rate gyros). This type of drift compensation is often called attitude and heading referenced and such a system is often called an Attitude and Heading Reference System (AHRS).



The XKF-3 can be optimized according to the scenario in which the device is going to be used. Some pre-set user scenarios are also available for different applications.

Typical performance characteristics of MTx orientation output, according to the official specifications are:

- Dynamic Range: all angles in 3D
- Angular Resolution: 0.05° (1σ standard deviation of zero-mean angular random walk)
- Repeatability: 0.2°
- Static Accuracy (roll/pitch): 0.5°
- Static Accuracy (heading): 1.0°
- Dynamic Accuracy: 2° RMS (although it may depend on type of motion)
- Update Rate: user settable, max 256 Hz

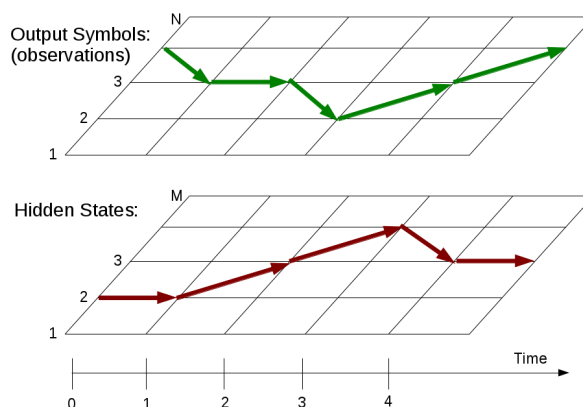
The Calibrated data performance specification for an MTx with standard configuration (with a rate gyro with a range of 1200 deg/s) are:

| Unit | Rate of turn [deg/s] | Acceleration [m/s^2] | Magnetic field [mGauss] | Temperature [$^\circ\text{C}$] |
|---|----------------------|---------------------------------|-------------------------|----------------------------------|
| Dimensions | 3 axes | 3 axes | 3 axes | - |
| Full scale [units] | ± 300 | ± 50 | ± 750 | -55 to +125 |
| Linearity [% of FS] | 0.1 | 0.2 | 0.2 | <1 |
| Bias stability [units 1σ] | 1 | 0.02 | 0.1 | 0.5 |
| Scale factor stability [% 1σ] | - | 0.03 | 0.5 | - |
| Noise density [units $\sqrt{\text{Hz}}$] | 0.05 | 0.002 | 0.5 (1σ) | - |
| Alignment error [deg] | 0.1 | 0.1 | 0.1 | - |
| Bandwidth [Hz] | 40 | 30 | 10 | - |
| A/D Resolution [bits] | 16 | 16 | 16 | 12 |

Hidden Markov Model (HMM)

A hidden Markov model (HMM) is a statistical Markov model in which the system being modeled is assumed to be a Markov process with unobserved (hidden) states. An HMM can be considered as the simplest dynamic Bayesian network. In a regular Markov model, the state is directly visible to the observer, and therefore the state transition probabilities are the only parameters. In a hidden Markov model, the state is not directly visible, but output, dependent on the state, is visible. Hidden Markov models are especially known for their application in temporal pattern recognition such as speech, handwriting, gesture recognition, part-of-speech tagging, musical score following, partial discharges and bio-informatics.

The Hidden Markov Model is a finite set of states, each of which is associated with a (generally multidimensional) probability distribution. Transitions among the states are governed by a set of probabilities called transition probabilities. In a particular state an outcome or observation can be generated, according to the associated probability distribution. It is only the outcome, not the state visible to an external observer and therefore states are "hidden" to the outside; hence the name Hidden Markov Model.



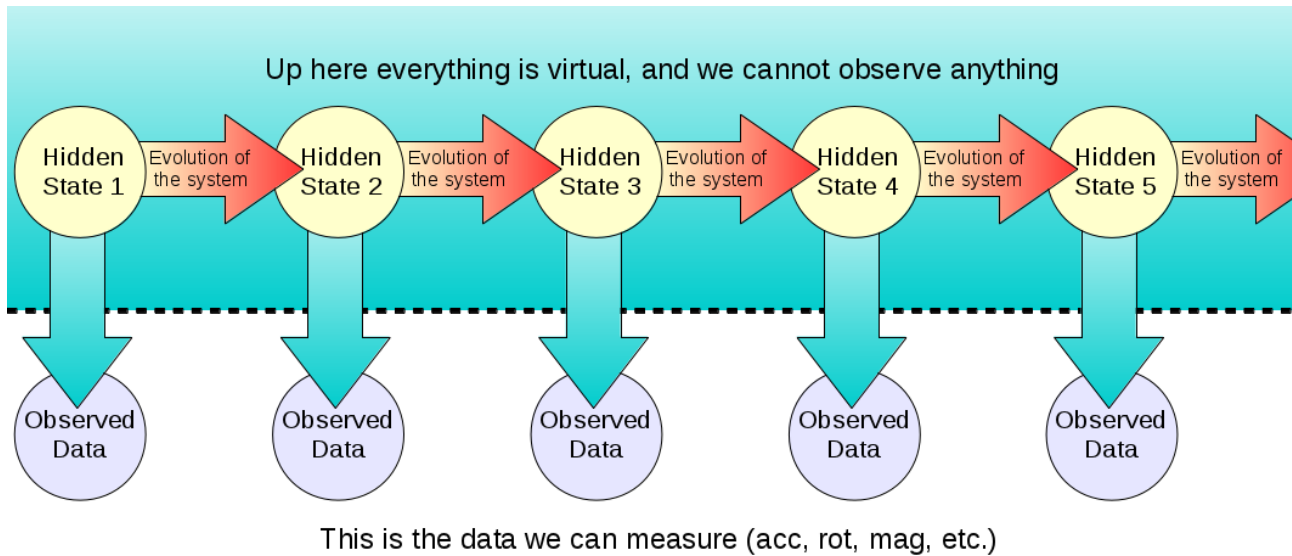
In order to define an HMM completely, following elements are needed:

- The number of internal states of the model, M
- The number of observation symbols in the alphabet, N
- A set of state transition probabilities, $A = \{a_{ij}\}$
- A probability distribution in each of the states, $B = \{b_j(k)\}$
- The initial state distribution, $\Pi = \{\pi_i\}$

Once we have an HMM, there are three canonical problems of interest:

- The Evaluation Problem: Given the model parameters, compute the probability of a particular output sequence. This problem is solved by the Forward and Backward algorithms.
- The Decoding Problem: Given the model parameters, find the most likely sequence of (hidden) states which could have generated a given output sequence. Solved by the Viterbi algorithm and Posterior decoding.
- The Learning Problem: Given an output sequence, find the most likely set of state transition and output probabilities. Solved by the Baum-Welch algorithm.

In our model of the system, at any time we have some internal state in which the person is (which is really virtual, and selected among a definite set of M possibilities). This internal state keeps changing as time passes, so we have, in fact, an ordered set of hidden states that the person goes through. As we have already said, these states are totally virtual and unobservable, but there is a statistical dependency between them and some variables we can measure (such as the acceleration or the turn rate).



The movements we're interested in are defined by a series of states that the person goes through in a certain order. Like, for example, when walking, you put up your foot, move it forward, let it drop, change your mass center, move up your other foot, and so on. If the probability that the measures we're getting come from the preset series of states we know the system has to go through, then we know that the person is doing the movement associated to that series.

$$Pr_t = Pr(S_t = i | O)$$

$$\alpha_{t+1} = \sum_{i=1}^N [\alpha_t(i) a_{i,j}] b_j(o_{t+1})$$

HMM algorithms

Definition of a Hidden Markov Model (HMM)

The Hidden Markov Model (HMM) is a variant of a finite state machine, in which there us a set of hidden states (Q), an output alphabet or observations (O), a transition matrix (A) that describes the probability of change from one hidden state to another, an output (emission) probabilities matrix (B) that describes the probabilistic relations between the hidden states and the output symbols, and initial state probabilities (Π).

The current state is not observable, that's why it's called hidden. Instead, each state produces an output with a certain probability (B). Usually the states (Q) and outputs (O) are understood, so an HMM is said to be defined by a triple $\Lambda = \{ A, B, \Pi \}$.

Hidden states $Q = \{ q_i \}, i = 1, \dots, M$

Observations (symbols) $O = \{ o_k \}, k = 1, \dots, N$

Initial state probabilities $\Pi = \{ p_i = P(q_i \text{ at } t = 1) \}$

Transition probabilities $A = \{ a_{ij} = P(q_j \text{ at } t + 1 \mid q_i \text{ at } t) \}$, where

- $P(a \mid b)$ is the conditional probability of a given b
- $t = 1, \dots, T$ is time
- and q_i is in Q .

Informally, A is the probability that the next state is q_j given that the current state is q_i .

Emission probabilities $B = \{ b_{ik} = b_i(o_k) = P(o_k \mid q_i) \}$, where

- o_k is in O

Informally, B is the probability that the output is o_k given that the current state is q_i .

Forward Algorithm

Let $\alpha_t(i)$ be the probability of the partial observation sequence $O_t = \{ o(1), o(2), \dots, o(t) \}$ to be produced by all possible state sequences that end at the i -th state:

$$\alpha_t(i) = P(o(1), o(2), \dots, o(t) \mid q(t) = q_i)$$

Then the unconditional probability of the partial observation sequence is the sum of $\alpha_t(i)$ over all M states.

The Forward Algorithm is a recursive algorithm for calculating $\alpha_t(i)$ for the observation sequence of increasing length t .

First, the probabilities for the single-symbol sequence are calculated as a product of initial i -th state probability and emission probability of the given symbol $o(1)$ in the i -th state. Then the recursive formula is applied. Assume we have calculated $\alpha_t(i)$ for some t . To calculate $\alpha_{t+1}(j)$, we multiply every $\alpha_t(i)$ by the corresponding transition probability from the i -th state to the j -th state, sum the products over all states, and then multiply the result by the emission probability of the symbol $o(t+1)$. Iterating the process, we can eventually calculate $\alpha_T(i)$, and then summing them over all states, we can obtain the required probability.

The algorithm works as shown:

- Initialization: $\alpha_1(i) = p_i \cdot b_i(o(1)), \text{ for } i = 1, \dots, M$
- Recursion: $\alpha_{t+1}(j) = \left[\sum_{i=1}^M \alpha_t(i) a_{ij} \right] b_j(o(t+1)), \text{ where } i = 1, \dots, M; t = 1, \dots, T-1$
- Termination: $P(o(1) o(2) \dots o(T)) = \sum_{j=1}^M \alpha_T(j)$

Backward Algorithm

In a similar way, we can introduce a symmetrical backward variable $\beta_t(i)$ as the conditional probability of the partial observation sequence from $o(t+1)$ to the end to be produced by all state sequences that start at i -th state:

$$\beta_t(i) = P(o(t+1), o(t+2), \dots, o(T) \mid q(t) = q_i)$$

The Backward Algorithm calculates recursively backward variables going backward along the observation sequence. The Forward Algorithm is typically used for calculating the probability of an observation sequence to be emitted by an HMM, but, as we shall see later, both procedures are heavily used for finding the optimal state sequence and estimating the HMM parameters.

The algorithm works as shown:

- Initialization: $\beta_T(i) = 1$, for $i = 1, \dots, M$ (according to the definition, $\beta_t(i)$ does not exist, so this is just a formal extension for the algorithm to work)
- Recursion: $\beta_t(i) = \sum_{j=1}^M a_{ij} b_j(o(t+1)) \beta_{t+1}(j)$, where $i = 1, \dots, M; t = T-1, T-2, \dots, 1$
- Termination: $P(o(1) o(2) \dots o(T)) = \sum_{j=1}^M p_j b_j(o(1)) \beta_1(j)$

Obviously both Forward and Backward algorithms must give the same results for total probabilities $P(O) = P(o(1), o(2), \dots, o(T))$.

Posterior decoding

There are several possible criteria for finding the most likely sequence of hidden states. One is to choose states that are individually most likely at the time when a symbol is emitted. This approach is called posterior decoding.

Let $\lambda_t(i)$ be the probability of the model to emit the symbol $o(t)$ being in the i -th state for the given observation sequence O :

$$\lambda_t(i) = P(q(t) = q_i \mid O)$$

It is easy to derive that:

$$\lambda_t(i) = \alpha_t(i) \beta_t(i) / P(O), \text{ for } i = 1, \dots, M; t = 1, \dots, T$$

Then at each time we can select the state $q(t)$ that maximizes $\lambda_t(i)$:

$$q(t) = \arg \max \{ \lambda_t(i) \}$$

Posterior decoding works fine in the case when HMM is ergodic (i.e. there is transition from any state to any other state). If applied to an HMM of another architecture, this approach could give a sequence that may not be a legitimate path because some transitions are not permitted.

Viterbi algorithm

The Viterbi algorithm chooses the best state sequence that maximizes the likelihood of the state sequence for the given observation sequence.

Let $\delta_t(i)$ be the maximal probability of state sequences of the length t that end in state i and produce the t first observations for the given model. :

$$\delta_t(i) = \max \{ P(q(1), q(2), \dots, q(t-1) ; o(1), o(2), \dots, o(t) \mid q(t) = q_i) \}$$

The Viterbi algorithm is a dynamic programming algorithm that uses the same schema as the Forward algorithm except for two differences:

1. It uses maximization in place of summation at the recursion and termination steps.
2. It keeps track of the arguments that maximize $\delta_t(i)$ for each t and i , storing them in the M by T matrix ψ . This matrix is used to retrieve the optimal state sequence at the backtracking step.

The algorithm works as follows:

- Initialization: $\delta_1(i) = p_i \cdot b_i(o(1))$; $\psi_1(i) = 0$, for $i = 1, \dots, M$ (according to the definition, $\beta_T(i)$ does not exist, so this is just a formal extension for the algorithm to work)
- Recursion: $\delta_t(j) = \max_i [\delta_{t-1}(i) a_{ij}] b_j(o(t))$; $\psi_t(j) = \arg \max_i [\delta_{t-1}(i) a_{ij}]$
- Termination: $p^* = \max_i [\delta_T(i)]$; $q^*_T = \arg \max_i [\delta_T(i)]$

Path (state sequence) backtracking: $q^*_t = \psi_{t+1}(q^*_{t+1})$, for $t = T - 1, T - 2, \dots, 1$

Baum–Welch algorithm

The Baum–Welch algorithm, also known as forward-backward algorithm, is a special case of Estimation Maximization (EM) method. It can compute maximum likelihood estimates and posterior mode estimates for the parameters (transition and emission probabilities) of an HMM, when given only emissions as training data.

Baum–Welch algorithm works by assigning initial probabilities to all the parameters. Then until the training converges, it adjusts the probabilities of the HMM's parameters so as to increase the probability the model assigns to the training set.

If no prior information is available, the parameters will be assigned random probabilities. In case domain knowledge is available, an informed initial guess can be made for the parameter values.

Once the initial values are assigned to the parameters, the algorithm enters a loop for training. In each iteration, based on the tags and corresponding probabilities that the current model assigns, probabilities are estimated. That is the parameter values are adjusted in each iteration. Training stops when the increase in the probability of the training set between iterations falls below some small value.

Forward-backward algorithm guarantees to find a locally best set of values from the initial parameter values. It works well if small amount of manually tagged corpus given.

The algorithm works as follows:

- Initialization: set $\lambda = (A, B, \pi)$ with random initial conditions. The algorithm updates the parameters of λ iteratively until convergence
- The forward procedure: We define: $\alpha_i(t) = p(O_1 = o_1, \dots, O_t = o_t, Q_t = i | \lambda)$, which is the probability of seeing the partial sequence o_1, \dots, o_t and ending up in state i at time t . We can efficiently calculate $\alpha_i(t)$ recursively as: $\alpha_i(t) = \pi_i \cdot b_i(o_1); \alpha_j(t+1) = b_j(o_{t+1}) \sum_{i=1}^M \alpha_i(t) \cdot a_{ij}$
- The backward procedure: This is the probability of the ending partial sequence o_{t+1}, \dots, o_T given that we started at state i , at time t . We can efficiently calculate $\beta_i(t)$ as

$$\beta_i(T) = 1; \beta_i(t) = \sum_{j=1}^M \beta_j(t+1) a_{ij} b_j(o_{t+1})$$

- Using α and β , we can calculate the following variables:

$$\gamma_i(t) \equiv p(Q_t = i | O, \lambda) = \frac{\alpha_i(t) \beta_i(t)}{\sum_{j=1}^M \alpha_j(t) \beta_j(t)}$$

$$\xi_{ij} \equiv p(Q_t = i, Q_{t+1} = j | O, \lambda) = \frac{\alpha_i(t) a_{ij} \beta_j(t+1) b_j(o_{t+1})}{\sum_{i=1}^M \sum_{j=1}^M \alpha_i(t) a_{ij} \beta_j(t+1) b_j(o_{t+1})}$$

- Having γ and ξ , one can define update rules as follows:

$$\bar{\pi} = \gamma_i(1)$$

$$\bar{a}_{ij} = \frac{\sum_{t=1}^{T-1} \xi_{ij}(t)}{\sum_{t=1}^{T-1} \gamma_i(t)}$$

$$\bar{b}_i = \frac{\sum_{t=1}^T \delta_{O_t, o_k} \gamma_i(t)}{\sum_{t=1}^T \gamma_i(t)} \quad (\text{note that the summation in the nominator of } b_i(k) \text{ is only over}$$

observed symbols equal to o_k)

Using the updated values of A , B and π , a new iteration is performed until convergence.

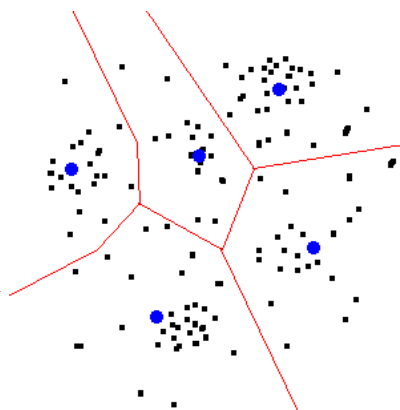
K-means clustering

k-means clustering is a method of cluster analysis which aims to partition n observations into k clusters in which each observation belongs to the cluster with the nearest mean. This results in a partitioning of the data space into Voronoi cells.

Lloyd's algorithm, also known as Voronoi iteration or relaxation, is the most common algorithm used for grouping data points into a given number of categories, used for k-means clustering.

Given an initial set of k means m_1, \dots, m_k , the algorithm proceeds by alternating between two steps:

- Assignment step: Assign each observation to the cluster with the closest mean (i.e. partition the observations according to the Voronoi diagram generated by the means).
- Update step: Calculate the new means to be the centroid of the observations in the cluster.



The algorithm is deemed to have converged when the assignments no longer change.

Commonly used initialization methods for the initial set of k means are Forgy and Random Partition:

- The Forgy method randomly chooses k observations from the data set and uses these as the initial means.
- The Random Partition method first randomly assigns a cluster to each observation and then proceeds to the Update step, thus computing the initial means to be the centroid of the cluster's randomly assigned points.

The Forgy method tends to spread the initial means out, while Random Partition places all of them close to the center of the data set.

Overview of the algorithm used

The algorithm used can be divided in 3 main steps:

- Load the data and cluster them, so that we finally have a temporal set of discrete output symbols.
- Train the HMM model with those data
- Test the HMM model with new data, which hasn't been used in the model

The data stored by the IMU is composed of 10-dimension arrays, which include a timestamp, three measures from the 3-axis accelerometers, other three measures from the 3-axis gyros, and finally three more measures from the 3-axis magnetometers. From those data, we're only using the 3-axis accelerometers at the moment (that is, we're using 3 dimensional vectors as input data).

We're not analyzing the whole bunch of data all at once. Instead, we're just using sets of 60 consecutive measurements, which is a bit greater than the number of measurements that describe a whole average cycle when walking. The data in the files is thus splitted in chunks of fixed size, which we're feeding to the algorithm.

Once the data has been extracted from the files, a k-means clustering algorithm is applied over it, so that we get a series of output symbols. The number of output symbols is predefined and, even though we're going to see that in more detail, the number of 30 output symbols seems to work OK as a reference.

Before training the HMM model, we're defining the initial transition matrix in such a way that the internal states have to be cyclically consecutive. That is, the probability of each change of state just depends on the current state and the previous one. The initial parameters can be easily changed so that we can have a higher degree of play in the left-to-right HMM transition matrix (such as 3 or 4, meaning that each state depends in the 2 previous ones, or in the 3 previous ones, respectively).

For the training of the model, a Baum-Welch algorithm is used, which is iterated until the proportional change in the log likelihood is low enough, or 50 steps have been run, whatever comes first. The number of internal states is previously fixed, and 20, for example, seems to be a good option. It is suggested that there should be more output symbols than internal states, because we don't know beforehand whether or not the test data will contain the gesture or not.

The training algorithm needs as input the clustered training data, the initial transition matrix, the internal state alphabet, the number of output symbols, the maximum number of steps to run (50, already mentioned) and the proportional change in the log likelihood that would be enough for us (0.00001). From all this data, we get the 3 elements that define the HMM model (observation emission probabilities, state transition probabilities, initial state prior probabilities) and the log likelihood curve.

Finally, for testing whether an observation belongs to the defined model, we use the Forward algorithm.

Data

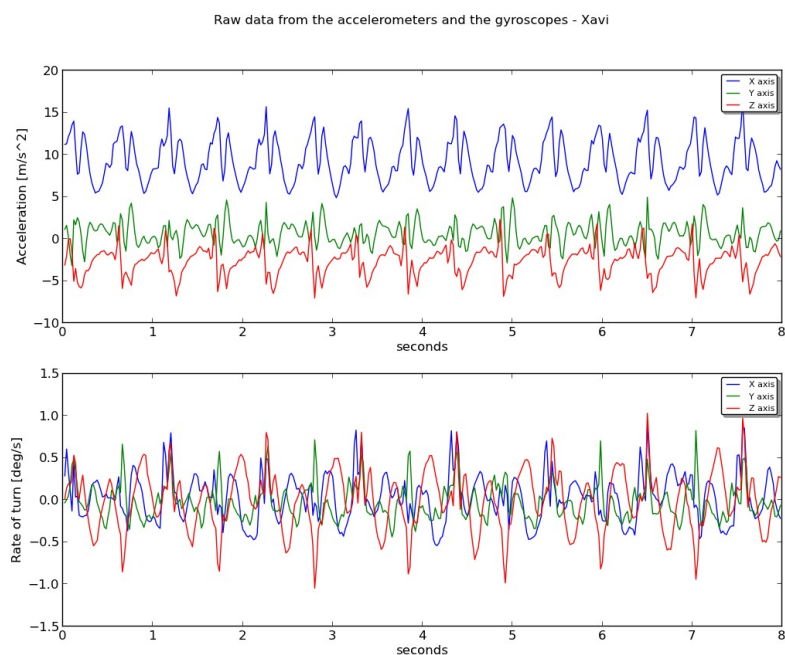
For testing if the proposed algorithm works or not, and whether it is able to differentiate between different movements, but also similar enough so that the system can be confused, 5 different situations have been measured:

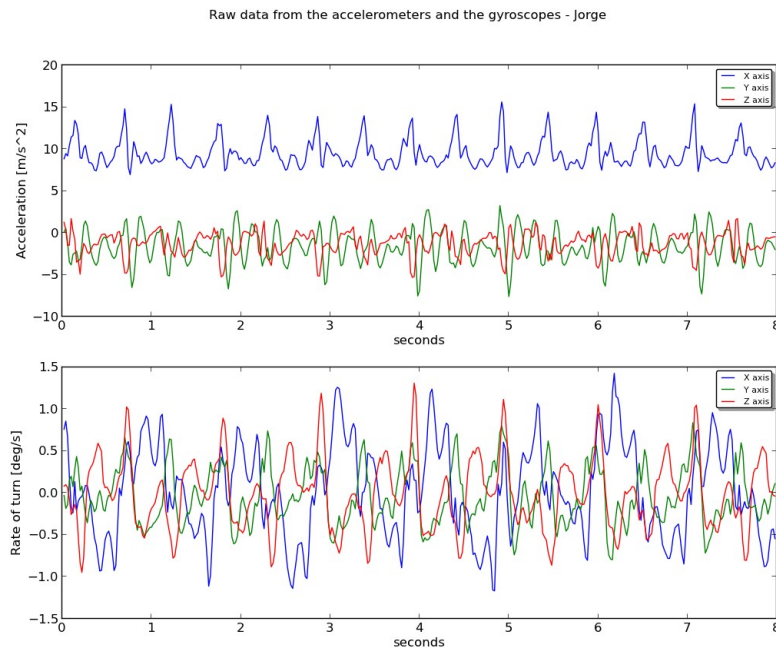
- The person is walking normally
- The person is going upstairs
- The person is going downstairs
- The person is running
- The person is walking up a ramp
- The person is walking down a ramp

Different measures have been taken from different people, so that the algorithm can be verified to be more or less independent of the person moving. The persons are randomly labeled as “Gabriel”, “Pablo”, “Jorge”, “Miguel_V”, “Xavi”, “Arturo”, “Marta”, “Silvia”, “Monica” and “Angela”, the first six corresponding to males, and the last four to females.

The data is captured at 50 times per second, and includes the three accelerometers, the three gyroscopes, the three magnetometers, and the time-stamp. It is passed through a Kalman filter so that it is improved and the drifts are minimized.

Although the main pattern of a person walking are somehow similar, there's a significant degree of difference between two people. A couple of graphs might show that, although you have all of them in the appendixes.





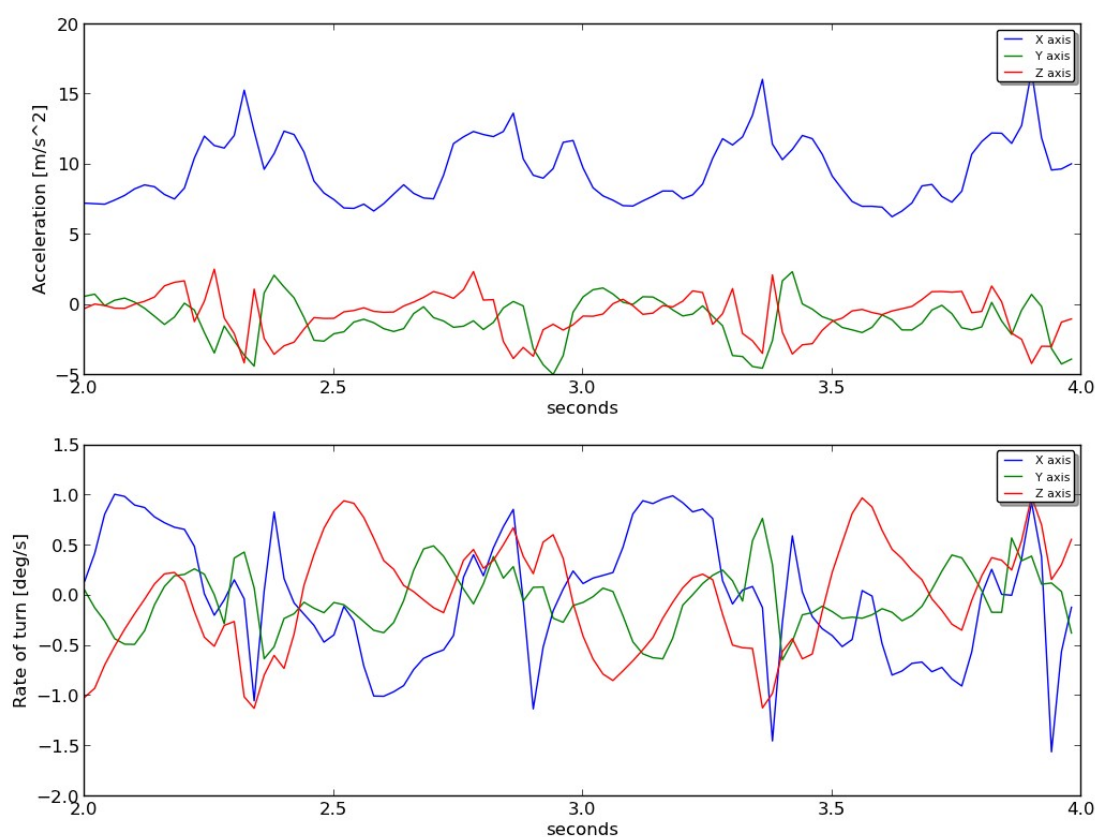
The only data we're using in our experiments for the moment is accelerometers. At a later stage we will see if including gyroscopes might improve the performance, and in which circumstances.

Parameters

Size of the array of data (data_width)

Moving is a continuous action that doesn't have a predefined length. That is, the bunch of data captured in every experiment is potentially unlimited. In real life, though, you don't usually keep doing the same movement for a long time, but you keep changing (you walk for a while, then you go upstairs, you run a bit, you go down a ramp, etc.). We want a bigger granularity than having to analyze a long series of data all at once: we want to be able to know what the person is doing at every instant.

Raw data from the accelerometers and the gyroscopes - Pablo Walking



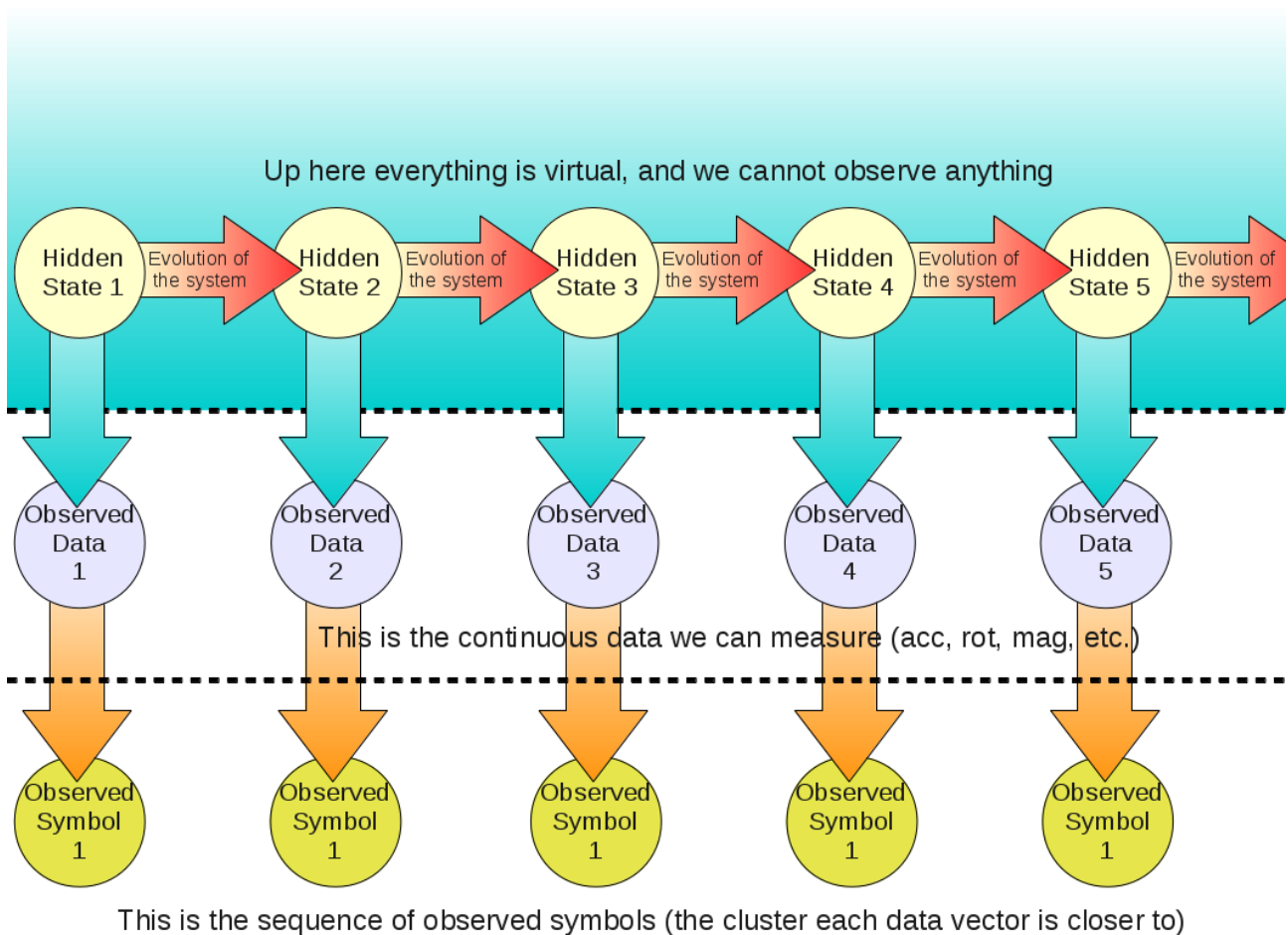
As we can see in the image above, or in the graphs already shown in previous pages, 2 seconds of walking (100 measures, in the data we have taken) include almost 4 full walking cycles. Thus, we would want to include at least enough data to have a full walking cycle. We settled for 60 measures (1.2 s) as a chunk size big enough to include more than a full cycle, but small enough to be able to find out the sequence of actions someone is doing, with a resolution slightly higher than 1 second.

Data dimensions used to generate the observable symbols (D)

For what we have seen in the data, and what our intuition tells us, the most important data (a priori) to find out whether a person is walking or not is that coming from the accelerometers. Thus, for starters, we will settle for sequences of vectors of size $D=3$ to generate the symbols, by clustering those vectors before using or training the HMM model.





Number of observable symbols in the alphabet (N)

We're using a conventional discrete HMM for modeling the walking person, but the data we gather from the sensors is discrete, so we're clustering it. As a result, we get a temporal sequence of points corresponding to the center of the cluster that is closer to the data that has been read. We consider each cluster an observable symbol, and the whole set of clusters is the alphabet. The number of clusters is thus the number of observable symbols, and is something that we're defining beforehand.



Number of internal states of the model (M)

We're assuming that the system is cyclically evolving through a series of finite states. The fact of going orderly through all of them is what we call a step.

| | |
|-------|--|
| N = 3 |  |
| N = 4 |  |
| N = 5 |  |
| N = 6 |  |

Degree of play in the left-to-right HMM transition matrix (LR)

The left-to-right HMM transition matrix keeps the probabilities of moving from a certain internal state to another. Thus, depending on how we configure it we will be able to force the states to move from one to another consecutively, or we will allow some of them to be skipped.

How we initially define this matrix will limit its evolution, but also will help the system to find a proper solution. Thus, we'll experiment how the model works by severely forcing a strict order of internal states (LR=2), we'll also allow one state to be skipped at any point (LR=3), or even two states (LR=4), and see what happens.

The following matrix shows the initial matrix in each of these cases, and also a graphical representation of what it means:

| | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
|---|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|---|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|---|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|
| <p>LR = 2</p> <p>Graphical representation: LR=2 M=6</p> <p>Prior transition matrix:</p> <table border="1"> <tr><td>0.50</td><td>0.50</td><td>0.00</td><td>0.00</td><td>0.00</td><td>0.00</td></tr> <tr><td>0.00</td><td>0.50</td><td>0.50</td><td>0.00</td><td>0.00</td><td>0.00</td></tr> <tr><td>0.00</td><td>0.00</td><td>0.50</td><td>0.50</td><td>0.00</td><td>0.00</td></tr> <tr><td>0.00</td><td>0.00</td><td>0.00</td><td>0.50</td><td>0.50</td><td>0.00</td></tr> <tr><td>0.00</td><td>0.00</td><td>0.00</td><td>0.00</td><td>0.50</td><td>0.50</td></tr> <tr><td>0.50</td><td>0.00</td><td>0.00</td><td>0.00</td><td>0.00</td><td>0.50</td></tr> </table> | 0.50 | 0.50 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.50 | 0.50 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.50 | 0.50 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.50 | 0.50 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.50 | 0.50 | 0.50 | 0.00 | 0.00 | 0.00 | 0.00 | 0.50 | <p>LR = 3</p> <p>Graphical representation: LR=3 M=6</p> <p>Prior transition matrix:</p> <table border="1"> <tr><td>0.33</td><td>0.33</td><td>0.33</td><td>0.00</td><td>0.00</td><td>0.00</td></tr> <tr><td>0.00</td><td>0.33</td><td>0.33</td><td>0.33</td><td>0.00</td><td>0.00</td></tr> <tr><td>0.00</td><td>0.00</td><td>0.33</td><td>0.33</td><td>0.33</td><td>0.00</td></tr> <tr><td>0.00</td><td>0.00</td><td>0.00</td><td>0.33</td><td>0.33</td><td>0.33</td></tr> <tr><td>0.33</td><td>0.00</td><td>0.00</td><td>0.00</td><td>0.33</td><td>0.33</td></tr> <tr><td>0.33</td><td>0.33</td><td>0.00</td><td>0.00</td><td>0.00</td><td>0.33</td></tr> </table> | 0.33 | 0.33 | 0.33 | 0.00 | 0.00 | 0.00 | 0.00 | 0.33 | 0.33 | 0.33 | 0.00 | 0.00 | 0.00 | 0.00 | 0.33 | 0.33 | 0.33 | 0.00 | 0.00 | 0.00 | 0.00 | 0.33 | 0.33 | 0.33 | 0.33 | 0.00 | 0.00 | 0.00 | 0.33 | 0.33 | 0.33 | 0.33 | 0.00 | 0.00 | 0.00 | 0.33 | <p>LR = 4</p> <p>Graphical representation: LR=4 M=6</p> <p>Prior transition matrix:</p> <table border="1"> <tr><td>0.25</td><td>0.25</td><td>0.25</td><td>0.25</td><td>0.00</td><td>0.00</td></tr> <tr><td>0.00</td><td>0.25</td><td>0.25</td><td>0.25</td><td>0.25</td><td>0.00</td></tr> <tr><td>0.00</td><td>0.00</td><td>0.25</td><td>0.25</td><td>0.25</td><td>0.25</td></tr> <tr><td>0.25</td><td>0.00</td><td>0.00</td><td>0.25</td><td>0.25</td><td>0.25</td></tr> <tr><td>0.25</td><td>0.25</td><td>0.00</td><td>0.00</td><td>0.25</td><td>0.25</td></tr> <tr><td>0.25</td><td>0.25</td><td>0.25</td><td>0.00</td><td>0.00</td><td>0.25</td></tr> </table> | 0.25 | 0.25 | 0.25 | 0.25 | 0.00 | 0.00 | 0.00 | 0.25 | 0.25 | 0.25 | 0.25 | 0.00 | 0.00 | 0.00 | 0.25 | 0.25 | 0.25 | 0.25 | 0.25 | 0.00 | 0.00 | 0.25 | 0.25 | 0.25 | 0.25 | 0.25 | 0.00 | 0.00 | 0.25 | 0.25 | 0.25 | 0.25 | 0.25 | 0.00 | 0.00 | 0.25 |
| 0.50 | 0.50 | 0.00 | 0.00 | 0.00 | 0.00 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0.00 | 0.50 | 0.50 | 0.00 | 0.00 | 0.00 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0.00 | 0.00 | 0.50 | 0.50 | 0.00 | 0.00 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0.00 | 0.00 | 0.00 | 0.50 | 0.50 | 0.00 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0.00 | 0.00 | 0.00 | 0.00 | 0.50 | 0.50 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0.50 | 0.00 | 0.00 | 0.00 | 0.00 | 0.50 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0.33 | 0.33 | 0.33 | 0.00 | 0.00 | 0.00 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0.00 | 0.33 | 0.33 | 0.33 | 0.00 | 0.00 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0.00 | 0.00 | 0.33 | 0.33 | 0.33 | 0.00 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0.00 | 0.00 | 0.00 | 0.33 | 0.33 | 0.33 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0.33 | 0.00 | 0.00 | 0.00 | 0.33 | 0.33 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0.33 | 0.33 | 0.00 | 0.00 | 0.00 | 0.33 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0.25 | 0.25 | 0.25 | 0.25 | 0.00 | 0.00 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0.00 | 0.25 | 0.25 | 0.25 | 0.25 | 0.00 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0.00 | 0.00 | 0.25 | 0.25 | 0.25 | 0.25 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0.25 | 0.00 | 0.00 | 0.25 | 0.25 | 0.25 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0.25 | 0.25 | 0.00 | 0.00 | 0.25 | 0.25 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0.25 | 0.25 | 0.25 | 0.00 | 0.00 | 0.25 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

Results

Description of the experiments

For each of the ten people from which data has been gathered, a series of different HMM models have been generated with different parameters, and those have been evaluated against the 6 different movements defined.

A different HMM model has been generated with data corresponding to a walking movement, for each combination of:

- ID (person) in { Gabriel, Marta, Pablo, Jorge, Miguel V, Silvia, Xavi, Arturo, Monica, Angela }
- LR (degree of play in the left-to-right HMM transition matrix) in { 2, 3, 4 }
- M (number of internal states) in { 10, 20, 30, 40 }
- N (number of observable symbols) in { 10, 20, 30, 40 }

That makes a total of 48 different models for every person, 480 models in total. Each of those models have been tested against totally unused data corresponding to 6 different movements:

- Walking

- Running
- Going up a ramp
- Going down a ramp
- Going upstairs
- Going downstairs

Performance of the resulting models

What we wanted to find out was a good approximation of the best parameter set for a walking movement. Each of the resulting models had a certain performance when successfully recognizing walking movements of that person, and also a performance when rejecting similar movements that were not walking. By averaging those performances, we get some indicator of the global performance of each model.

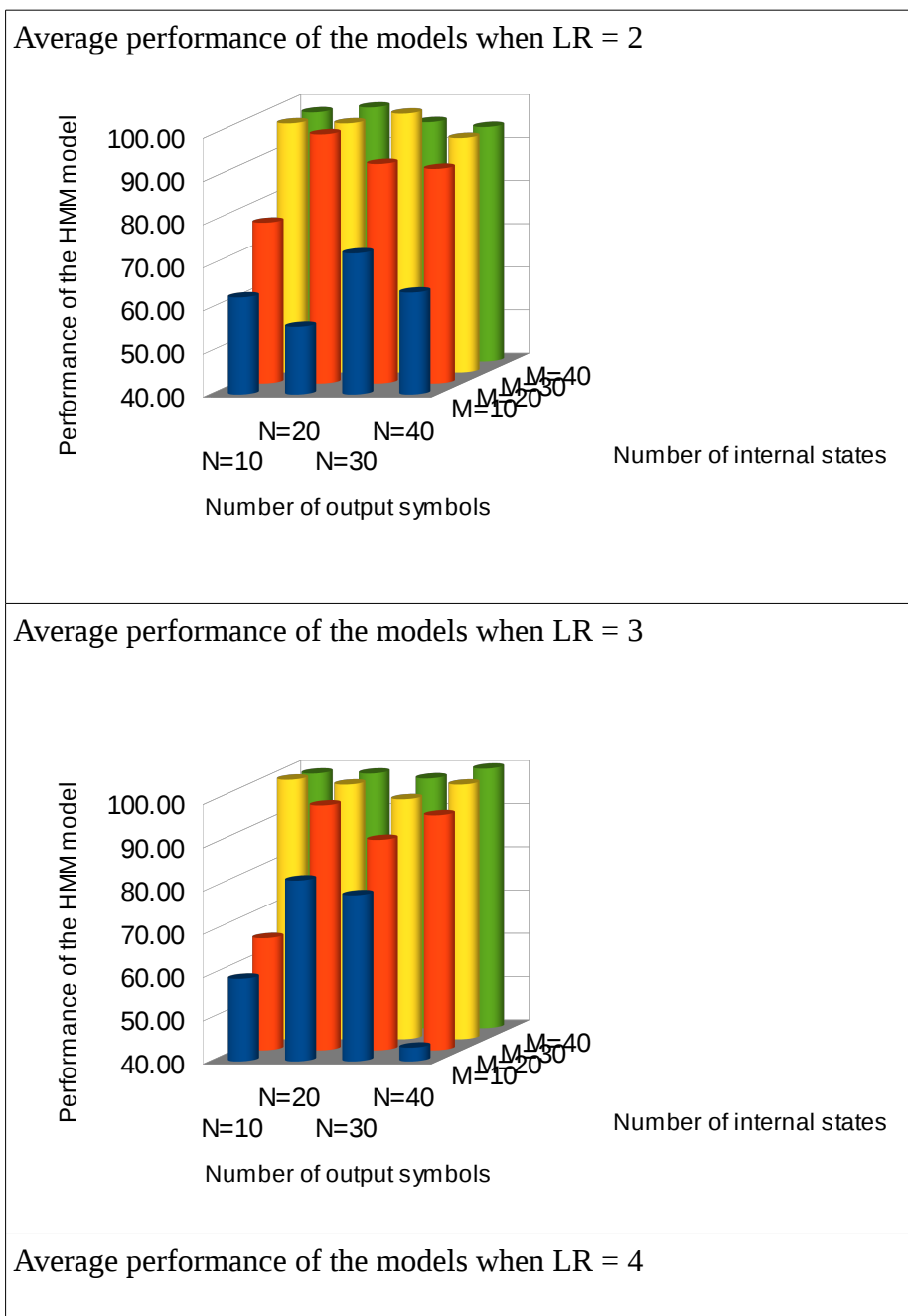
| Average performance of the models when LR = 2 | | | | |
|---|-------|-------|-------|-------|
| LR = 2 | M=10 | M=20 | M=30 | M=40 |
| N=10 | 55.02 | 78.69 | 89.35 | 88.62 |
| N=20 | 55.10 | 89.63 | 95.17 | 93.31 |
| N=30 | 58.68 | 90.31 | 95.69 | 92.84 |
| N=40 | 59.25 | 90.97 | 95.19 | 94.73 |

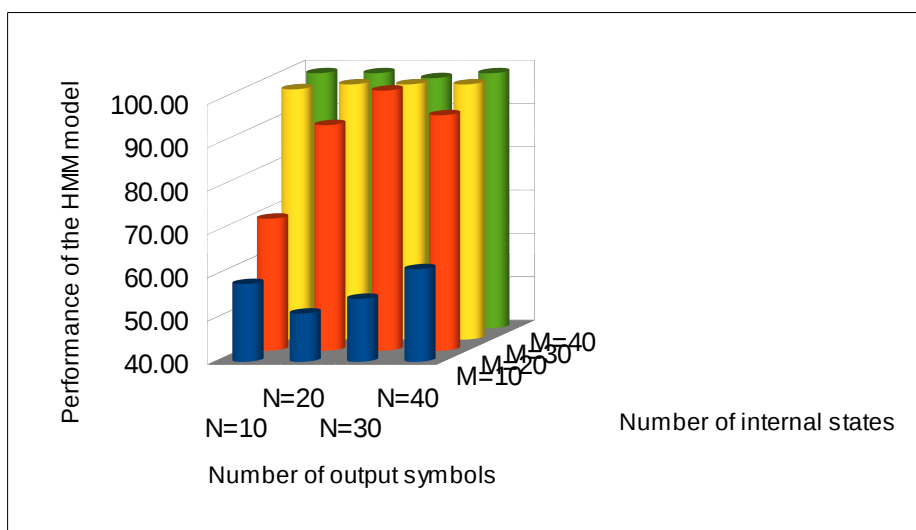
| Average performance of the models when LR = 3 | | | | |
|---|-------|-------|-------|-------|
| LR = 3 | M=10 | M=20 | M=30 | M=40 |
| N=10 | 46.89 | 76.25 | 95.22 | 95.51 |
| N=20 | 51.48 | 87.27 | 96.42 | 96.29 |
| N=30 | 50.83 | 89.66 | 95.04 | 95.88 |
| N=40 | 47.23 | 91.91 | 94.91 | 96.12 |

| Average performance of the models when LR = 4 | | | | |
|---|-------|-------|-------|-------|
| LR = 4 | M=10 | M=20 | M=30 | M=40 |
| N=10 | 44.32 | 84.62 | 93.35 | 94.22 |
| N=20 | 52.86 | 87.03 | 94.85 | 98.01 |
| N=30 | 48.55 | 87.69 | 96.96 | 96.86 |
| N=40 | 55.35 | 88.88 | 95.29 | 95.63 |

As we can see, for certain values of the parameters the performance is quite good. The best case in this experiment is 98% of success, when LR=4, M=40 and N=20. That is, with 20 output symbols (clusters), 40 internal states (M), and being allowed to skip two states.

If we plot those tables in graphical form, we have:





Successful rejection of different movements

Intuitively, it is quite obvious that some movements are more similar to walking than others. We can certify this quantitatively by calculating the average performance of all the models for each person when trying to reject each movement different from walking:

| | Run | Upstairs | Downstairs | Up Ramp | Down Ramp |
|----------|--------------|----------|------------|--------------|--------------|
| Gabriel | 87.67 | 88.69 | 89.93 | 75.00 | 77.19 |
| Marta | 96.96 | 88.10 | 96.88 | 55.39 | 95.57 |
| Pablo | 78.68 | 95.00 | 91.44 | 78.62 | 79.06 |
| Jorge | 81.85 | 79.36 | 85.88 | 83.77 | 72.80 |
| Miguel V | 77.08 | 83.71 | 77.98 | 79.72 | 84.98 |
| Silvia | 94.70 | 95.83 | 99.58 | 90.51 | 53.29 |
| Xavi | 79.38 | 81.25 | 91.67 | 44.72 | 60.69 |
| Arturo | 93.75 | 91.15 | 84.03 | 53.47 | 71.57 |
| Monica | 94.36 | 76.04 | 93.75 | 57.29 | 76.98 |
| Angela | 98.51 | 90.28 | 98.96 | 76.04 | 68.75 |
| Average | 88.29 | 86.94 | 91.01 | 69.45 | 74.09 |

The average rejection rate for the best HMM model (LR=4, M=40, N=20), for each type of movement and each person, is:

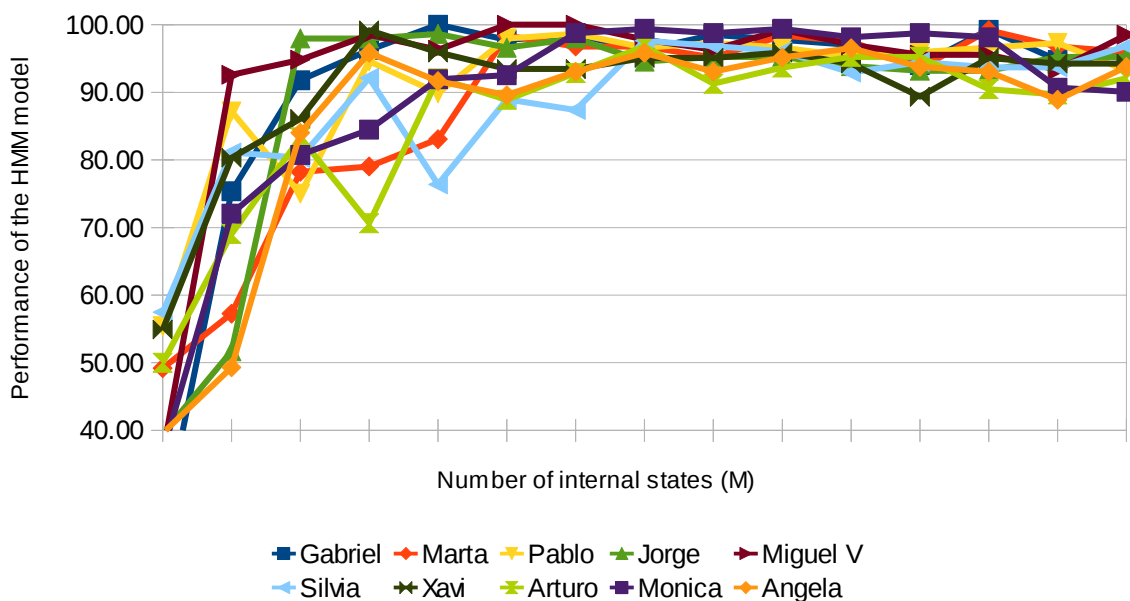
| | Run | Upstairs | Downstairs | Up Ramp | Down Ramp |
|----------|--------|----------|------------|--------------|--------------|
| Gabriel | 100.00 | 100.00 | 100.00 | 100.00 | 100.00 |
| Marta | 100.00 | 100.00 | 100.00 | 100.00 | 100.00 |
| Pablo | 100.00 | 100.00 | 100.00 | 100.00 | 100.00 |
| Jorge | 100.00 | 100.00 | 100.00 | 100.00 | 100.00 |
| Miguel V | 100.00 | 100.00 | 100.00 | 100.00 | 100.00 |
| Silvia | 100.00 | 100.00 | 100.00 | 100.00 | 94.74 |
| Xavi | 100.00 | 100.00 | 100.00 | 86.67 | 93.33 |
| Arturo | 100.00 | 100.00 | 100.00 | 100.00 | 100.00 |
| Monica | 100.00 | 100.00 | 100.00 | 95.45 | 95.00 |
| Angela | 100.00 | 100.00 | 100.00 | 100.00 | 94.74 |
| Average | 100.00 | 100.00 | 100.00 | 98.21 | 97.78 |

Influence of the number of internal states

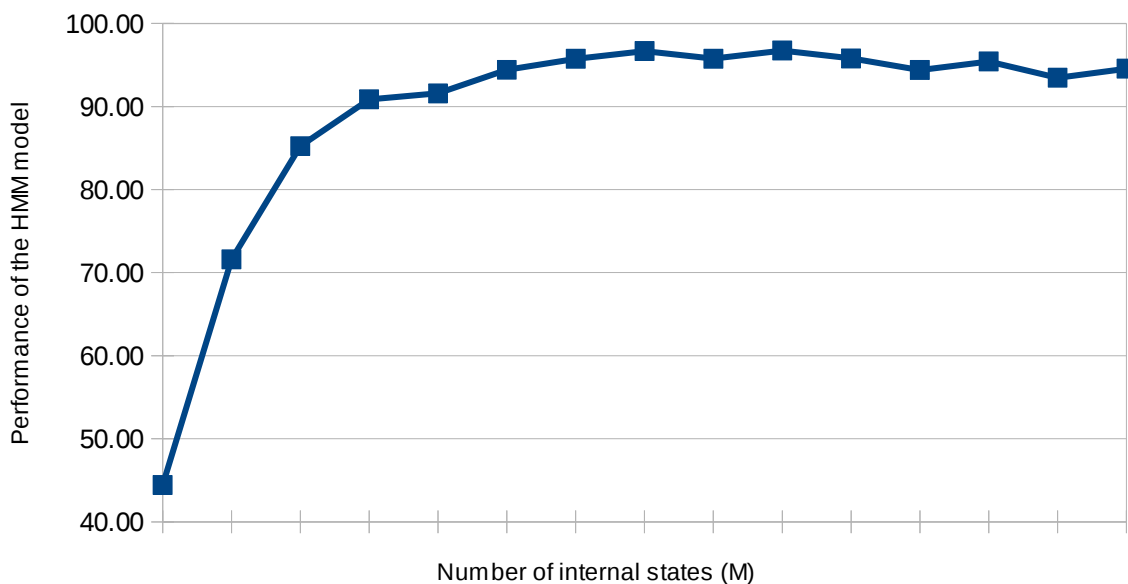
As an additional experiment, some tests were carried out adding more data, and changing only one parameter at the time, starting from the parameters mentioned earlier (LR=4, N=20, M=40).

As in the previous experiment, a different HMM model was trained for each person with the selected parameters, and it was tested against unused data from that person, doing the 6 movement types mentioned beforehand.

The overall performance of each HMM model for each person can be seen in the following graph:



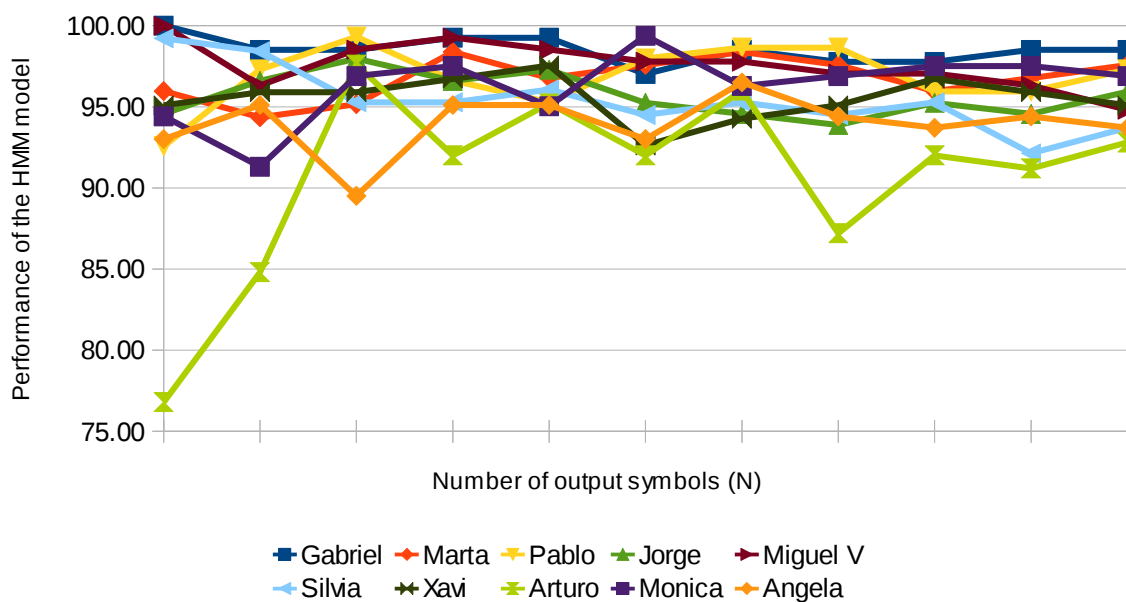
By averaging the results for all the people, we get the following curve:



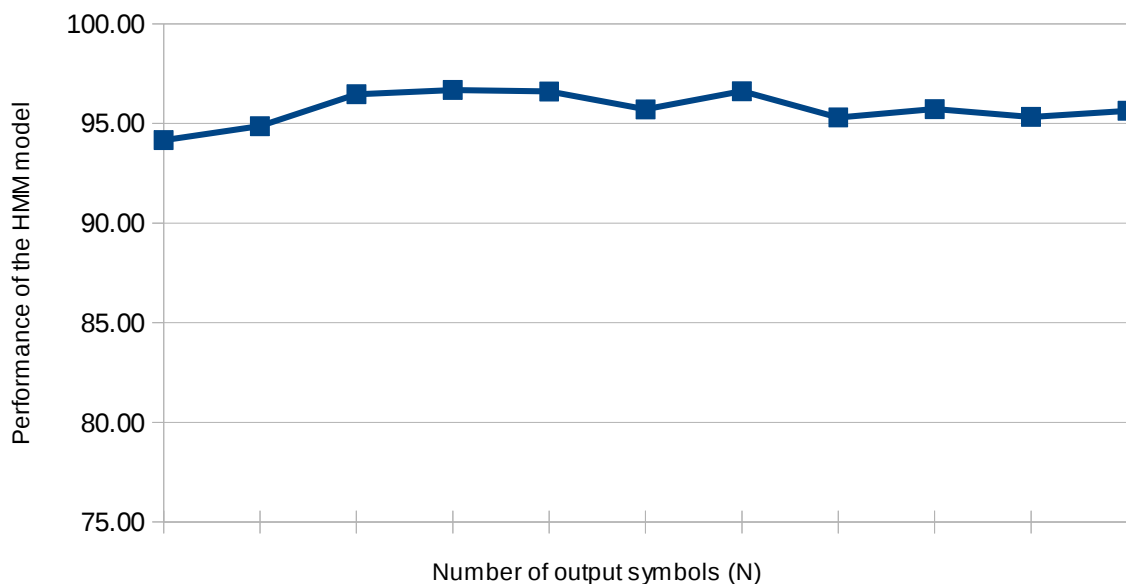
What we can see is that the performance is very sensitive to having less numbers of internal states than needed. When we reach a minimum threshold (in this case 25) the results start to be reasonable, but keep gradually increasing until they reach a maximum (in this case 45). After that, the performance seems to keep stable. Keep in mind that every chunk of data analyzed has, as mentioned earlier, 60 measurements, so any value of M above 60 will imply more internal states than data, which anyway makes sense because we are allowed to skip internal states.

Influence of the number of output symbols (clusters)

The same experiment has been carried out, but changing the number of output symbols instead:



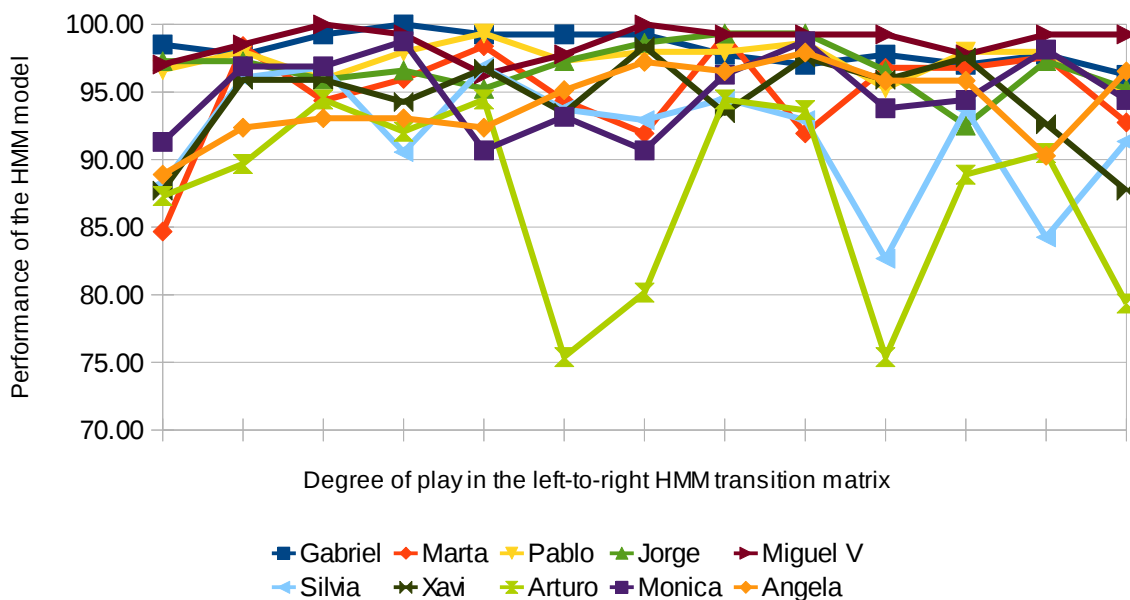
As in the previous case, averaging all the models with respect to the people provides us the following graph:



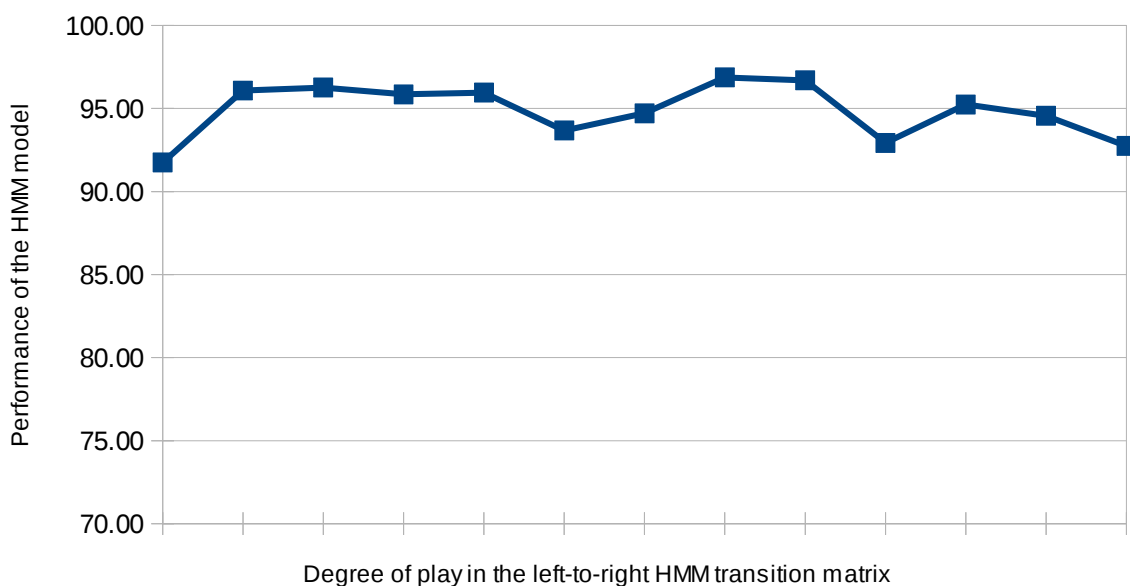
As can be seen, once over a minimum threshold, the number of clusters doesn't seem to be very important. With the exception of one of the people (Arturo), even $N=10$ seems to be almost enough for them. In any case, a maximum seems to be achieved between 20 and 40 (with the exception of 35, which seems to be an outlier).

Influence of the degree of play in the left-to-right transition matrix (LR)

If we keep the parameters mentioned earlier (M=40, N=20), but change LR instead, we get the following results:

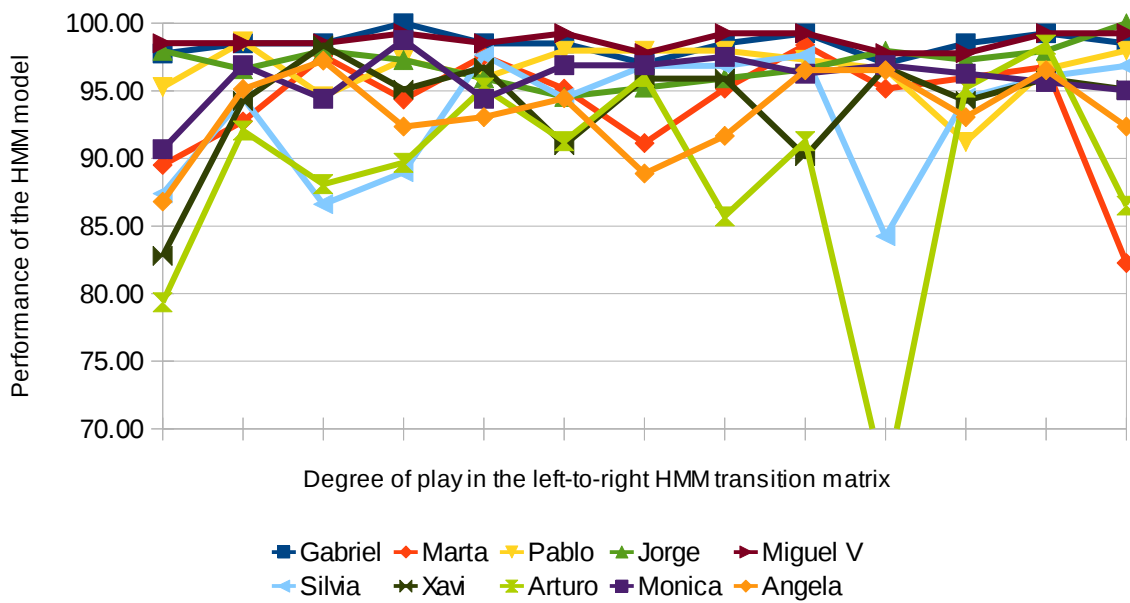


And, as in previous cases, if we average the results, we get:

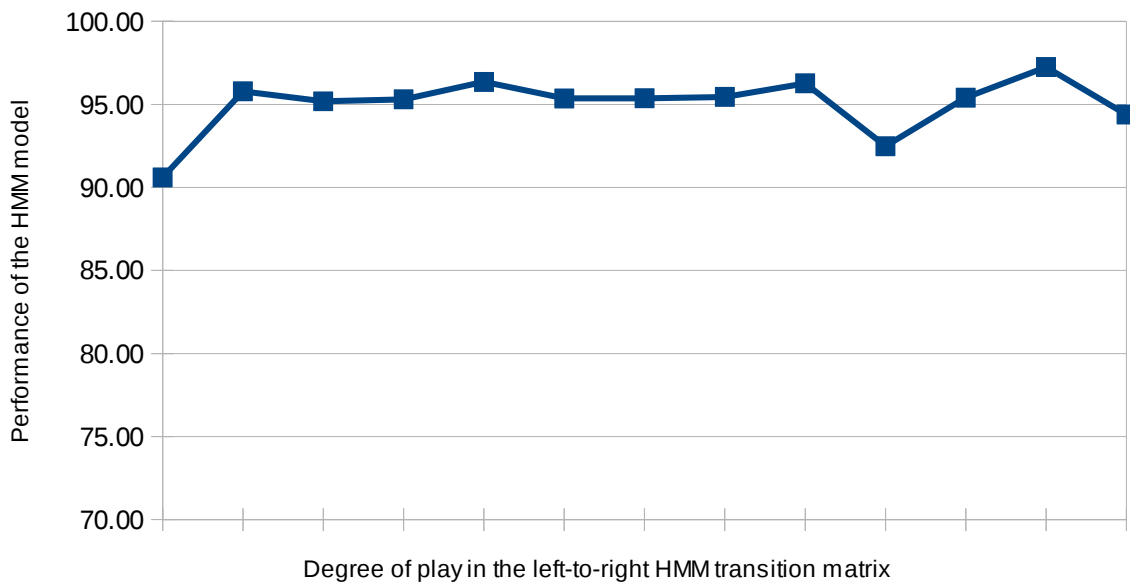


Repeatability of the results

As we can see in the previous graphs, we get very weird results for one of the people, “Arturo”, and to a lesser level also “Silvia”, that can makes us question how much the results will vary if we repeat the same experiment again. Well, here it goes:



And, as usual, the averaged results:



As in the previous results, we can see that the HMM models trained for “Arturo” and “Silvia” have an important degree of randomness, especially when LR is too large (above 6). This is quite likely to be caused by the different solutions provided by the clustering algorithm, and should be something to improve in the future.

Conclusions

Performance of the resulting models

The first thing we can find out about the purposed solution of applying a HMM model to classify the type of movement a person is doing is that it works. The performance of the resulting models are considerably high, even though we have used the simplest possible model.

We can see that the simpler a model is, the more successful it is detecting the movement that it has been trained on, but it commits a lots of mistakes when trying to find out that the movement is a different one. On the other hand, very complex model do have quite low performance in detecting the same kind of movement, although they can easily tell that some data belong to a different one.

In the following table we can see the performance when detecting a similar movement and a different one, when LR=4, respectively:

| Averaged performance of the models when detecting walking on data belonging to walking: | | | | | Averaged performance of the models when detecting walking on other movements: | | | | |
|---|-------|-------|-------|-------|---|-------|-------|-------|-------|
| LR = 4 | M=10 | M=20 | M=30 | M=40 | LR = 4 | M=10 | M=20 | M=30 | M=40 |
| N=10 | 99.29 | 99.29 | 97.75 | 96.10 | N=10 | 30.96 | 80.56 | 92.09 | 93.64 |
| N=20 | 97.90 | 96.39 | 90.08 | 93.28 | N=20 | 41.61 | 84.53 | 95.67 | 98.88 |
| N=30 | 96.02 | 94.65 | 92.64 | 89.73 | N=30 | 36.68 | 85.84 | 97.86 | 98.33 |
| N=40 | 95.46 | 95.17 | 90.61 | 85.03 | N=40 | 45.42 | 87.11 | 96.11 | 97.90 |

Rejection of other movements

We found out that some kinds of movement are more difficult to tell apart from walking than others. The following table shows the averaged results of rejecting different movements:

| Rejection of other mov. | LR = 2 | | | | | LR = 3 | | | | | LR = 4 | | | | |
|-------------------------|--------|-------|-------|-------|-------|--------|-------|-------|-------|-------|--------|-------|-------|-------|-------|
| | LR = 2 | M=10 | M=20 | M=30 | M=40 | LR = 3 | M=10 | M=20 | M=30 | M=40 | LR = 4 | M=10 | M=20 | M=30 | M=40 |
| Run | N=10 | 34.33 | 0.00 | 0.00 | 0.75 | N=10 | 35.82 | 3.73 | 0.00 | 0.00 | N=10 | 60.45 | 3.73 | 0.00 | 0.00 |
| | N=20 | 28.36 | 0.00 | 0.00 | 0.00 | N=20 | 38.81 | 0.00 | 0.00 | 0.00 | N=20 | 41.79 | 12.69 | 0.00 | 0.00 |
| | N=30 | 32.09 | 0.75 | 0.00 | 0.00 | N=30 | 63.43 | 2.24 | 0.00 | 0.00 | N=30 | 49.25 | 8.21 | 0.00 | 0.00 |
| | N=40 | 35.07 | 0.00 | 0.00 | 0.00 | N=40 | 63.43 | 0.00 | 0.00 | 0.00 | N=40 | 38.81 | 6.72 | 0.00 | 0.00 |
| Up Ramp | N=10 | 81.46 | 37.08 | 22.47 | 28.65 | N=10 | 84.27 | 56.74 | 11.24 | 12.36 | N=10 | 81.46 | 38.76 | 11.24 | 14.04 |
| | N=20 | 72.47 | 20.22 | 2.81 | 12.36 | N=20 | 77.53 | 32.02 | 5.62 | 2.81 | N=20 | 70.79 | 19.10 | 9.55 | 1.69 |
| | N=30 | 71.35 | 19.66 | 5.62 | 13.48 | N=30 | 67.98 | 25.28 | 8.99 | 3.37 | N=30 | 76.97 | 33.15 | 4.49 | 3.37 |
| | N=40 | 68.54 | 19.10 | 3.93 | 7.87 | N=40 | 73.60 | 23.60 | 9.55 | 2.81 | N=40 | 71.35 | 24.72 | 7.87 | 5.62 |
| Down Ramp | N=10 | 66.30 | 53.59 | 17.13 | 13.81 | N=10 | 79.01 | 42.54 | 9.39 | 3.87 | N=10 | 80.11 | 25.97 | 16.02 | 7.73 |
| | N=20 | 73.48 | 21.55 | 11.05 | 7.18 | N=20 | 70.17 | 20.99 | 6.63 | 6.63 | N=20 | 68.51 | 23.20 | 5.52 | 2.21 |
| | N=30 | 51.38 | 18.23 | 6.08 | 5.52 | N=30 | 71.27 | 12.71 | 8.84 | 0.55 | N=30 | 71.27 | 12.15 | 3.31 | 2.21 |
| | N=40 | 52.49 | 14.36 | 5.52 | 6.08 | N=40 | 71.82 | 7.73 | 4.97 | 1.66 | N=40 | 64.09 | 12.71 | 4.97 | 1.66 |
| Upstairs | N=10 | 46.07 | 7.87 | 3.37 | 1.12 | N=10 | 62.92 | 7.87 | 0.00 | 0.00 | N=10 | 52.81 | 2.25 | 0.00 | 0.00 |
| | N=20 | 52.81 | 3.37 | 0.00 | 1.12 | N=20 | 49.44 | 0.00 | 0.00 | 0.00 | N=20 | 49.44 | 3.37 | 1.12 | 0.00 |
| | N=30 | 57.30 | 0.00 | 0.00 | 2.25 | N=30 | 33.71 | 1.12 | 0.00 | 0.00 | N=30 | 59.55 | 0.00 | 0.00 | 0.00 |
| | N=40 | 47.19 | 0.00 | 0.00 | 1.12 | N=40 | 55.06 | 0.00 | 0.00 | 0.00 | N=40 | 44.94 | 0.00 | 0.00 | 0.00 |
| Downstairs | N=10 | 14.08 | 2.82 | 0.00 | 0.00 | N=10 | 50.70 | 4.23 | 0.00 | 0.00 | N=10 | 49.30 | 0.00 | 0.00 | 0.00 |
| | N=20 | 19.72 | 0.00 | 0.00 | 0.00 | N=20 | 42.25 | 0.00 | 0.00 | 0.00 | N=20 | 45.07 | 7.04 | 0.00 | 0.00 |
| | N=30 | 14.08 | 0.00 | 0.00 | 0.00 | N=30 | 52.11 | 0.00 | 0.00 | 0.00 | N=30 | 43.66 | 0.00 | 0.00 | 0.00 |
| | N=40 | 19.72 | 0.00 | 0.00 | 0.00 | N=40 | 43.66 | 0.00 | 0.00 | 0.00 | N=40 | 25.35 | 7.04 | 0.00 | 0.00 |

As we can see, going up or down a ramp is generally more difficult to tell apart from walking. Actually we're discriminating the kind of movement according to a log likelihood obtained when training the model. Of course, changing this number will increase or decrease the number of false positives and false negatives depending on our needs, if the movements are really too similar.

If we had a set of movements, some of them too similar, we could also use that log likelihood to select one among all the movements that were detected as probabilistically likely, so that would also improve the overall performance of the system.

Defining a model that would be valid for different users

Among the experiments we carried out, we tried to find whether a model built on walking data from different people would be able to work. The answer is that with our current algorithm we can't, and each model has yet to be trained for each person. The resulting model is not able to tell other movements apart from walking.

For solving that problem, we would have to somehow normalize the input data in a way that the output symbols for each person would be equivalent. That would probably mean a normalization taking into account the average and variance of the input data, as well as probably some other modifications, still to be tested.

Bibliography and links

Measurement of mobility-related activities:

- American College of Sports and Medicine. The Recommended Quantity and Quality of Exercise for Developing and Maintaining Cardiorespiratory and Muscular Fitness, and Flexibility in Adults. *Medicine & Science in Sports & Exercise*. 30(6): p. 975-991.
- Trost, S.G., State of the Art Reviews: Measurement of Physical Activity in Children and Adolescents. *American Journal of Lifestyle Medicine*, 2007. 1(4): p. 299-314.
- RC González, A.L.J.Á., Monitorización de la Actividad Física mediante Sensores Inerciales. Estado del Arte. 2006, Universidad de Oviedo. Laboratorio e Sistemas Multisensor y Robótica.
- Bussmann, J.B.J., et al., Measuring daily behavior using ambulatory accelerometry: The Activity Monitor. *Behavior Research Methods, Instruments, & Computers*, 2001. 33: p. 349-356.
- Kiani, K., C.J. Snijders, and E.S. Gelsema, Computerized analysis of daily life motor activity for ambulatory monitoring. *Technol. Health Care*, 1997. 5(4): p. 307-318.
- Mathie, M., et al., Classification of basic daily movements using a triaxial accelerometer. *Medical and Biological Engineering and Computing*, 2004. 42(5): p. 679-687.
- Veltink, P.H., et al., Detection of static and dynamic activities using uniaxial accelerometers
- Detection of static and dynamic activities using uniaxial accelerometers. *Rehabilitation Engineering, IEEE Transactions on [see also IEEE Trans. on Neural Systems and Rehabilitation]*, 1996. 4(4): p. 375-385.
- Mantyjarvi, J., et al. Recognizing human motion with multiple acceleration sensors
- Recognizing human motion with multiple acceleration sensors. in *Systems, Man, and Cybernetics, 2001 IEEE International Conference on*. 2001.
- Van Laerhoven, K., K. Van Laerhoven, and O. Cakmakci. What shall we teach our pants?
- What shall we teach our pants? in *Wearable Computers, 2000. The Fourth International Symposium on*. 2000.
- Ravi, N., et al., Activity Recognition from Accelerometer Data. *American Association for Artificial Intelligence*, 2005.

Inertia Mobile Units (IMUs)

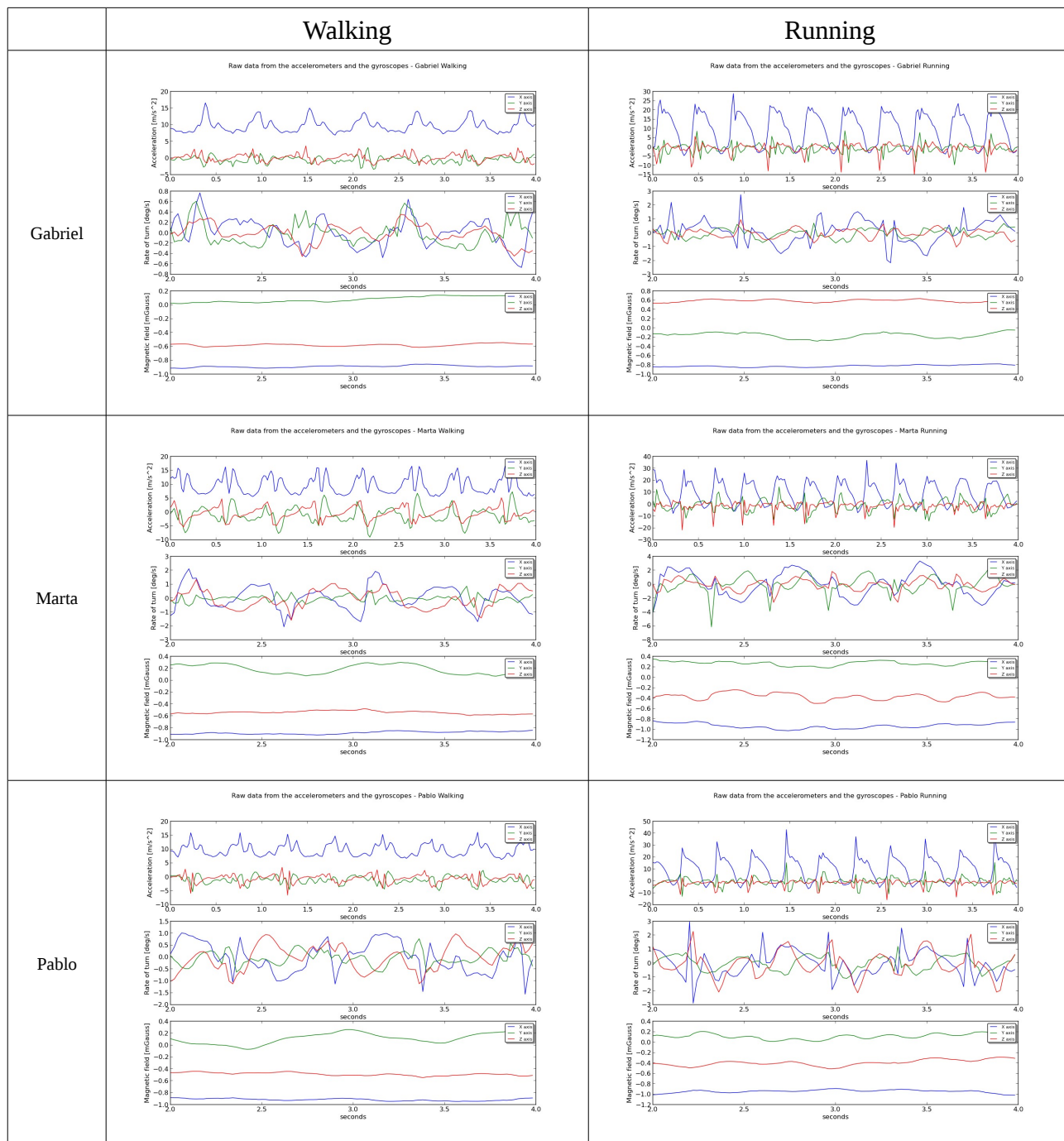
- Oliver J. Woodman, "An introduction to inertial navigation," Technical Report, UCAM-CL-TR-696, University of Cambridge, Aug 2007
- A Guide To using IMU (Accelerometer and Gyroscope Devices) in Embedded Applications (http://www.starlino.com/imu_guide.html)

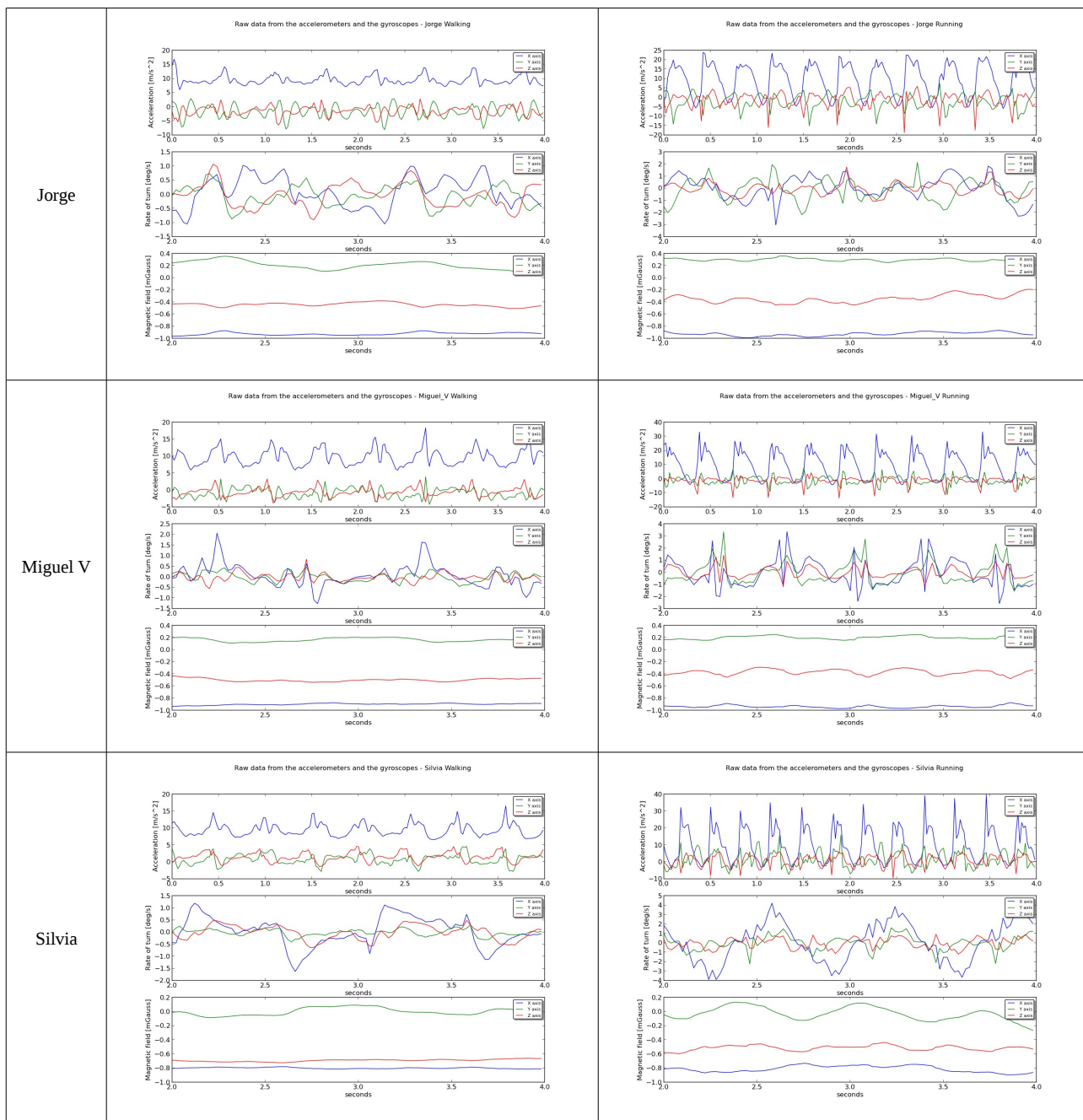
- Introduction to Inertial Navigation and Kalman filtering, Norwegian Space Centre, Kenneth Gade, FFI (<http://www.nornav.org/getfile.php/706509.753.qeqfawvevu/INSFFI.pdf?force=1>)
- Xsens MTx measurement unit specs (<http://www.xsens.com/en/general/mtx>)

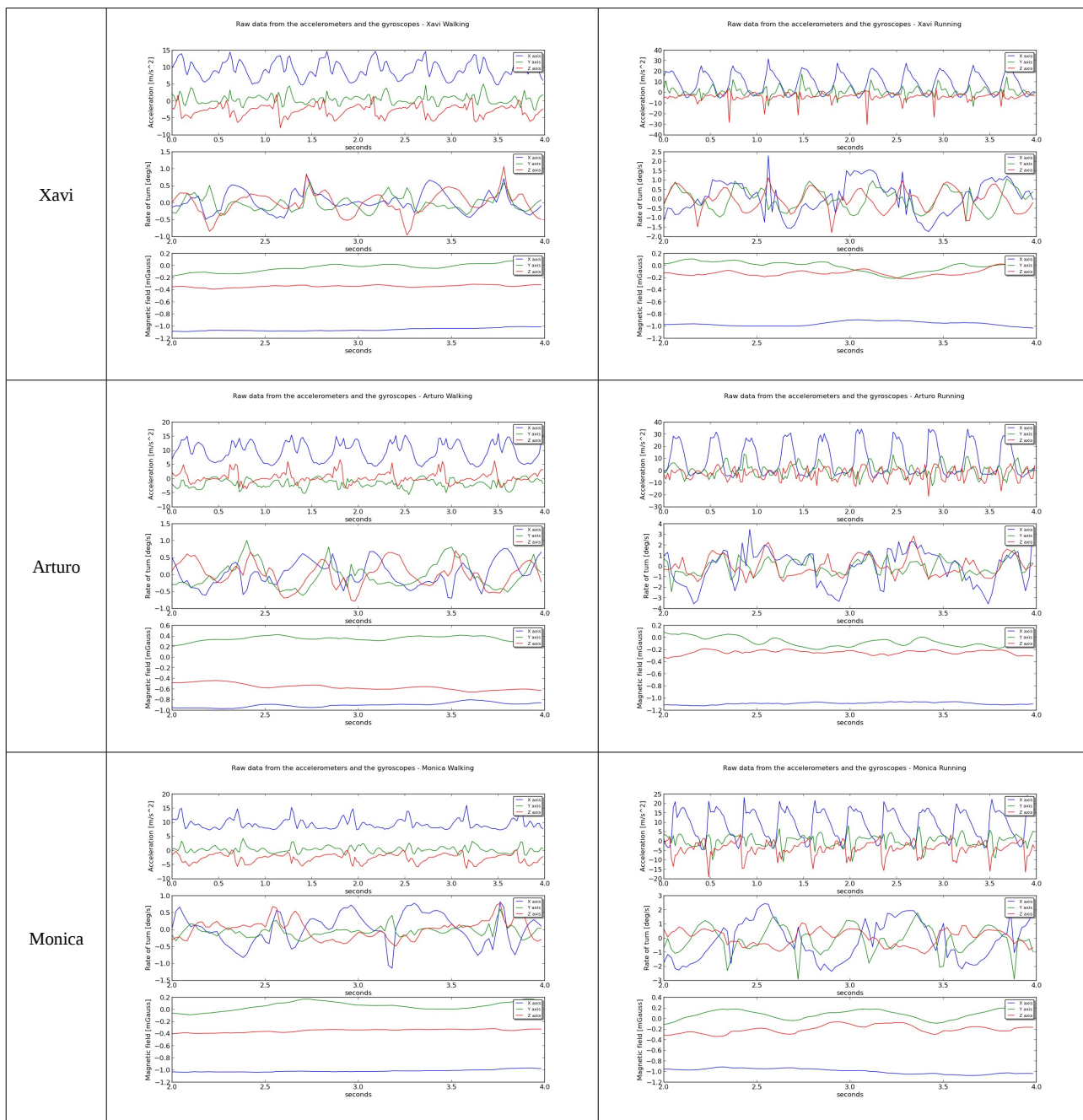
Hidden Markov Models

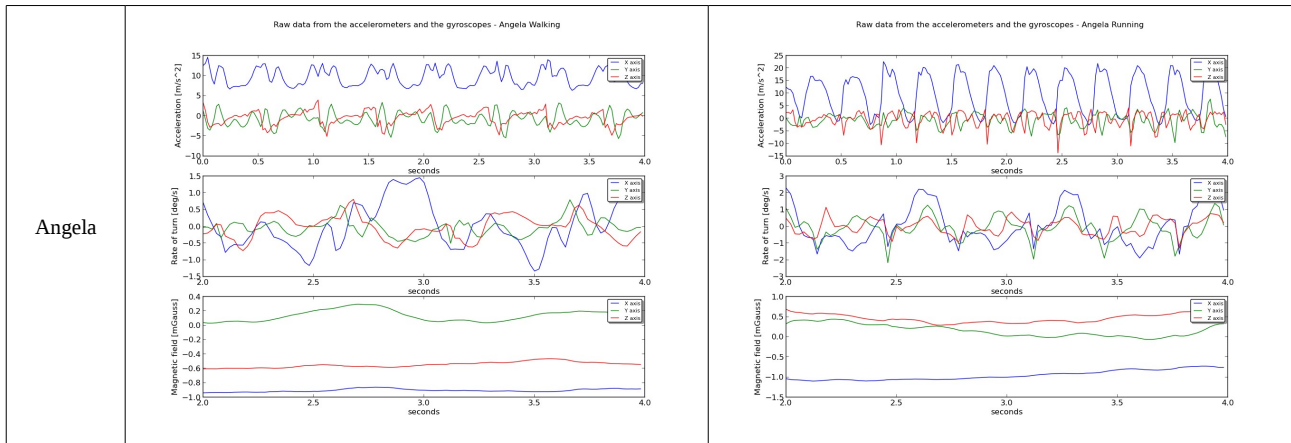
- T. Hastie, R. Tibshirani, J. Friedman, The Elements of Statistical Learning: Data Mining, Inference, and Prediction, Second Edition, 2009
- M. Stamp, A Revealing Introduction to Hidden Markov Models, San Jose State University, 2012
- Mackay, D.J.C., Ensemble Learning for Hidden Markov Models, 2006
- M. J. Beal, Reference code in Matlab for discrete-valued and for real-valued Gaussian observations, 2003 (<http://mlg.eng.cam.ac.uk/zoubin/software.html>)
- G. Shi et al., Recognition using MEMS Inertial Sensors, Proceedings of the 2008 IEEE International Conference on Robotics and Biomimetics, 2008
- A. Y. Benbasat, J. A. Paradiso, An Inertial Measurement Framework for Gesture Recognition and Applications, MIT Media Laboratory
- L. R. Rabiner, A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition, Proceedings of the IEEE, Vol. 77, No. 2, 1989
- V. Petrushin. Hidden Markov Models: Fundamentals and Applications. Part 2: Discrete and Continuous Hidden Markov Models, Online Symposium for Electronics Engineers. 2000
- J. C. Hall, How to Do Gesture Recognition With Kinect Using Hidden Markov Models (HMMs), 2011 (<http://www.creativedistracted.com/demos/gesture-recognition-kinect-with-hidden-markov-models-hmms/>)
- Online course about Machine Learning (<http://www.youtube.com/playlist?list=PLD0F06AA0D2E8FFBA>), by Mathematicalmonk, Chapter 14, Hidden Markov Models, HMMs (<http://www.youtube.com/watch?v=TPRoLreU9IA>)

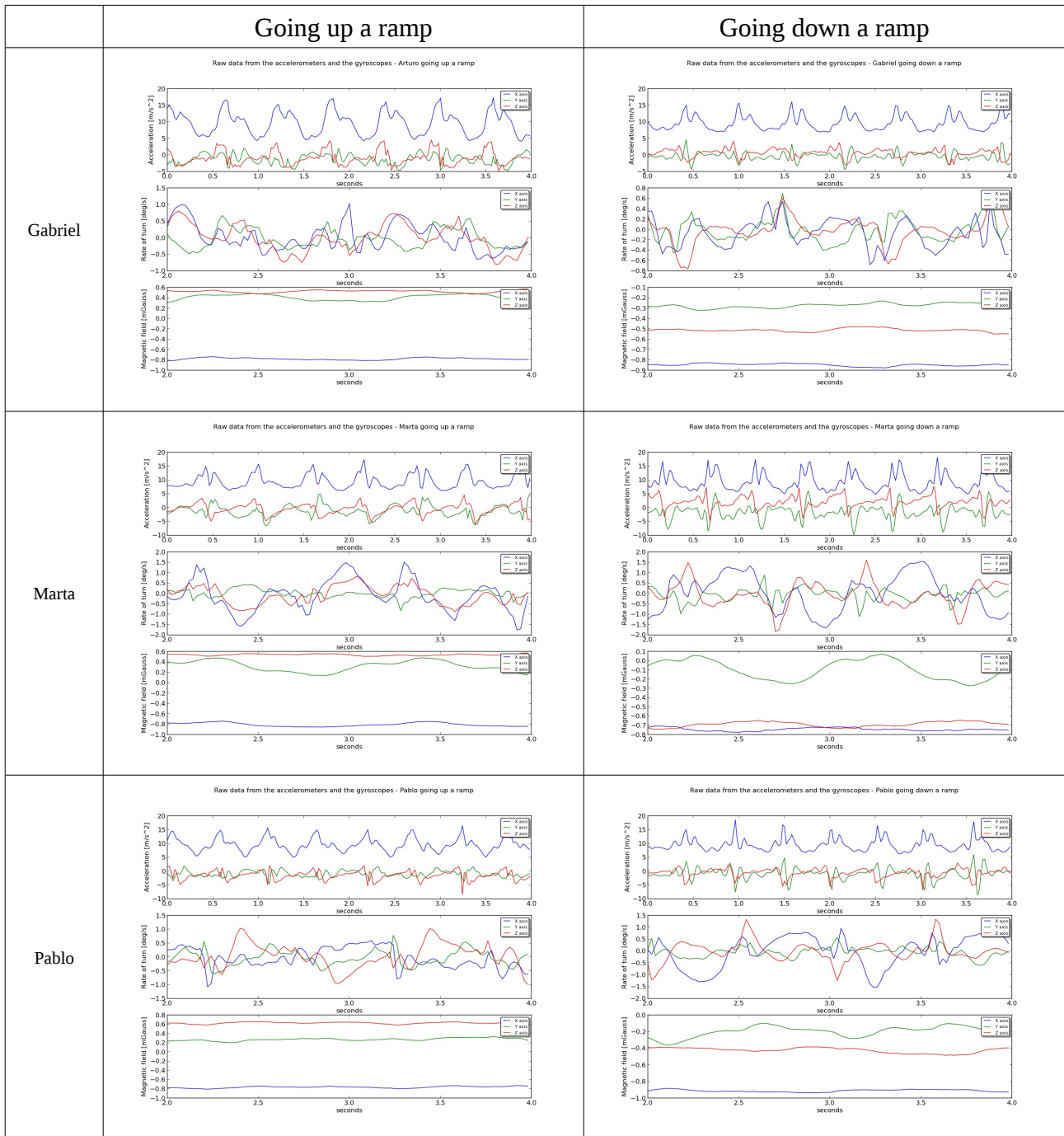
Appendix A: Representation of data for every person and kind of movement

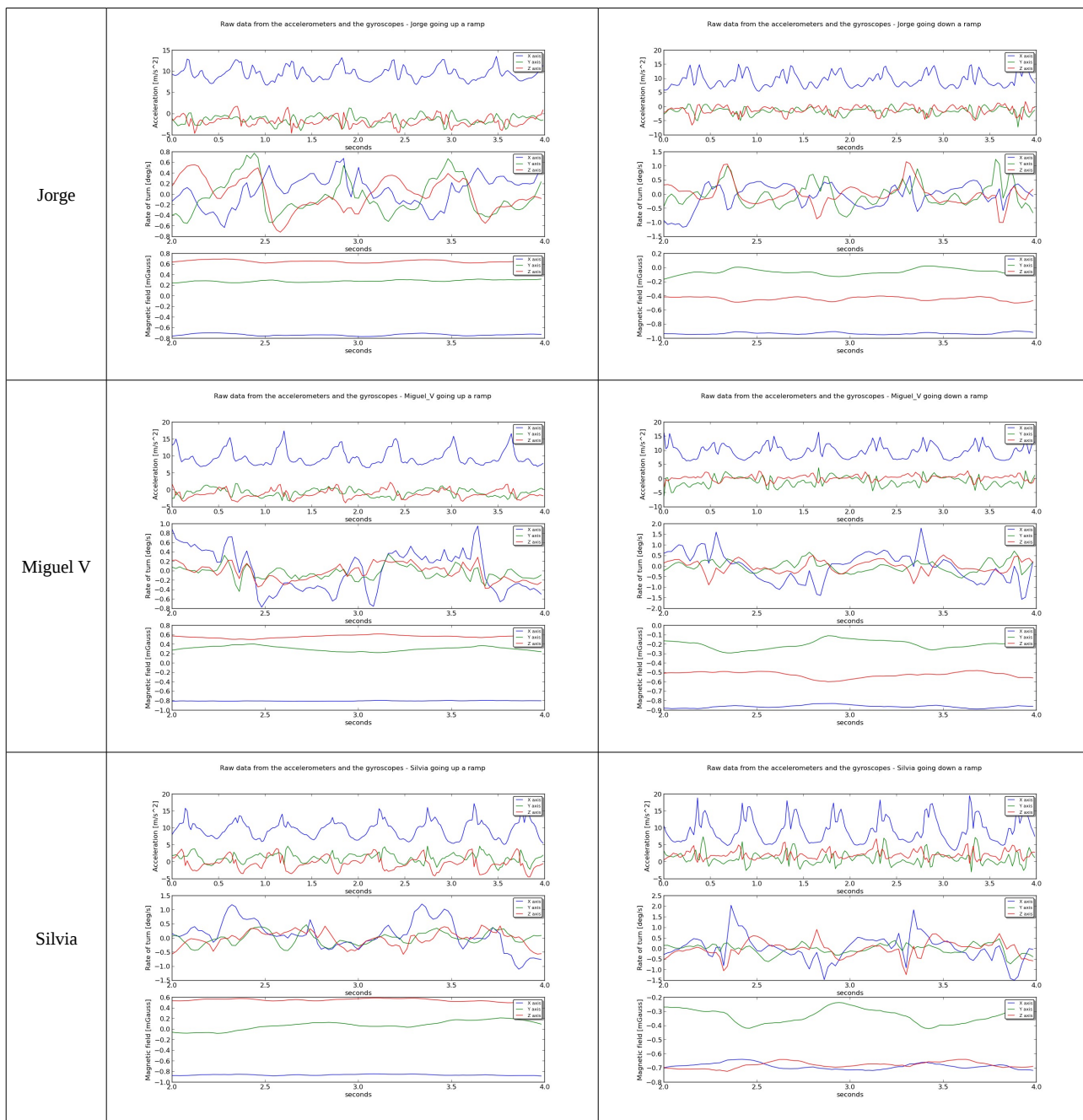


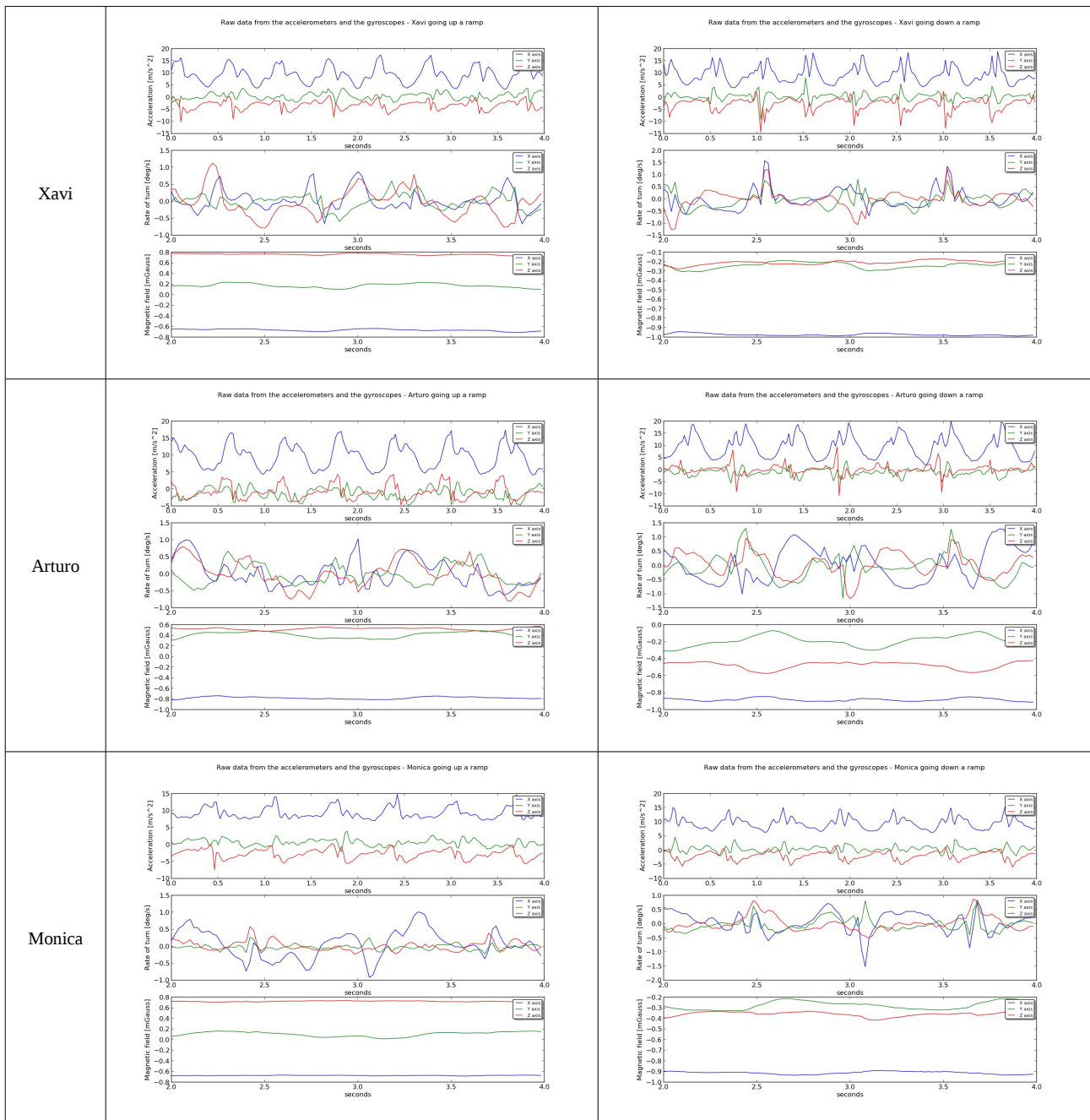


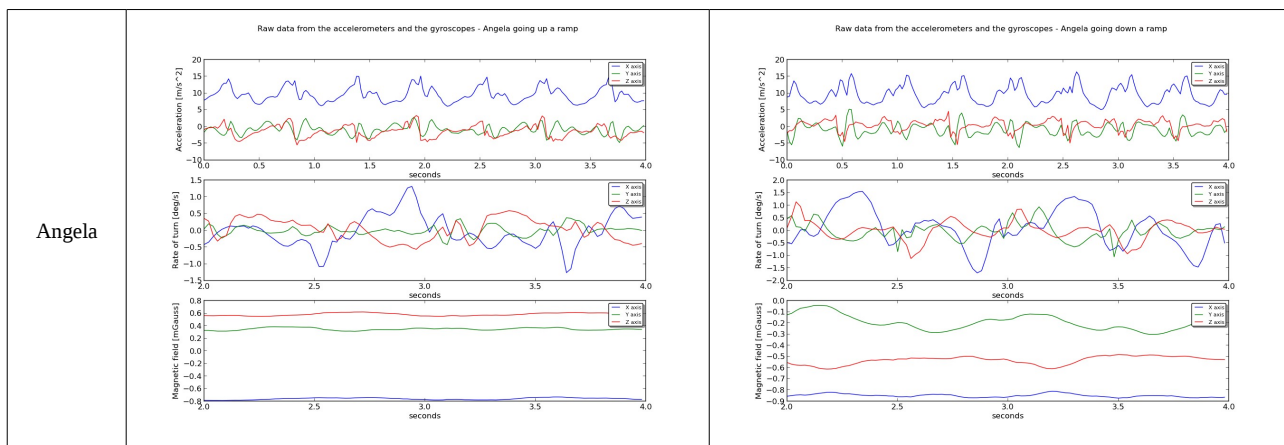


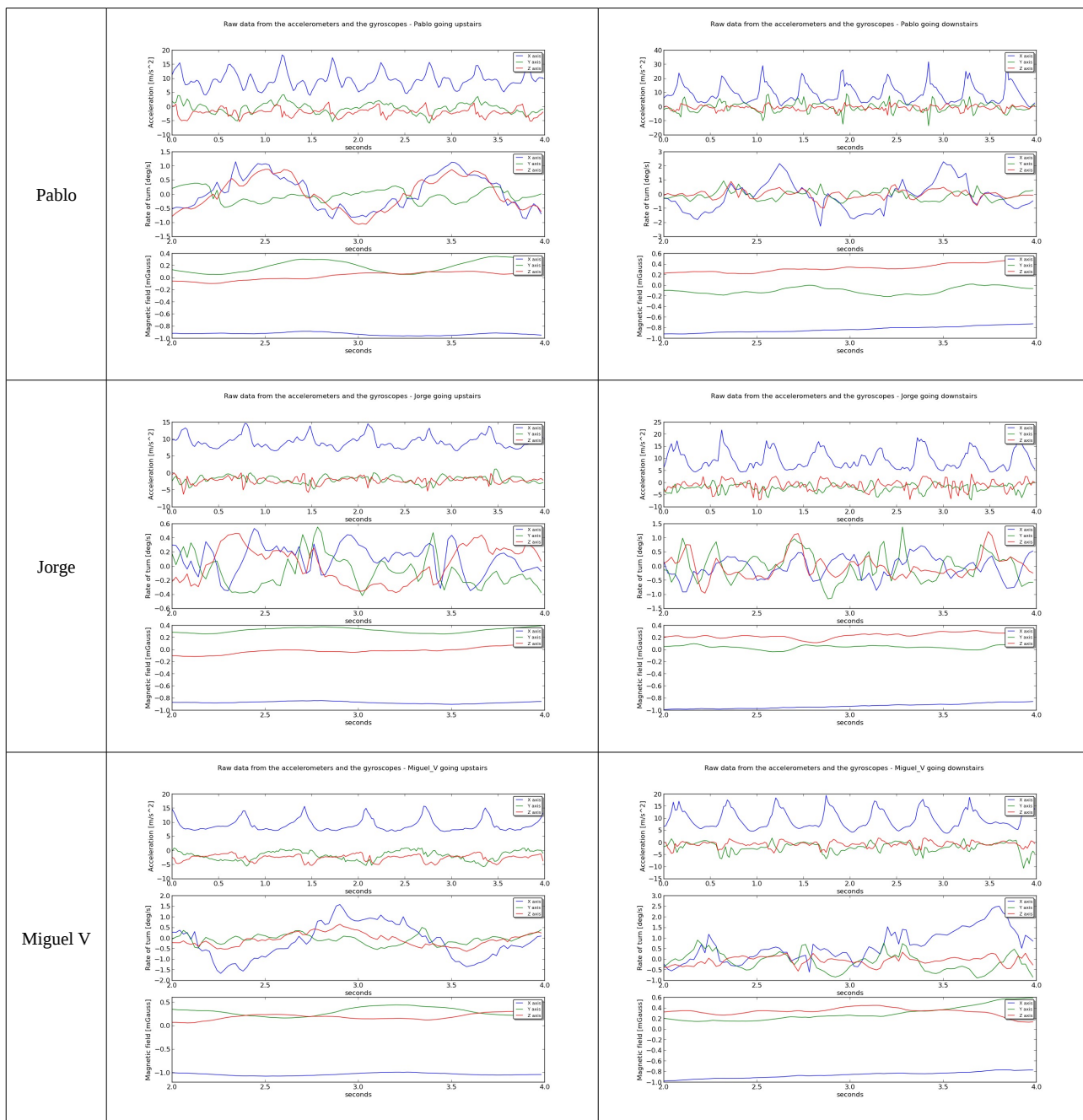


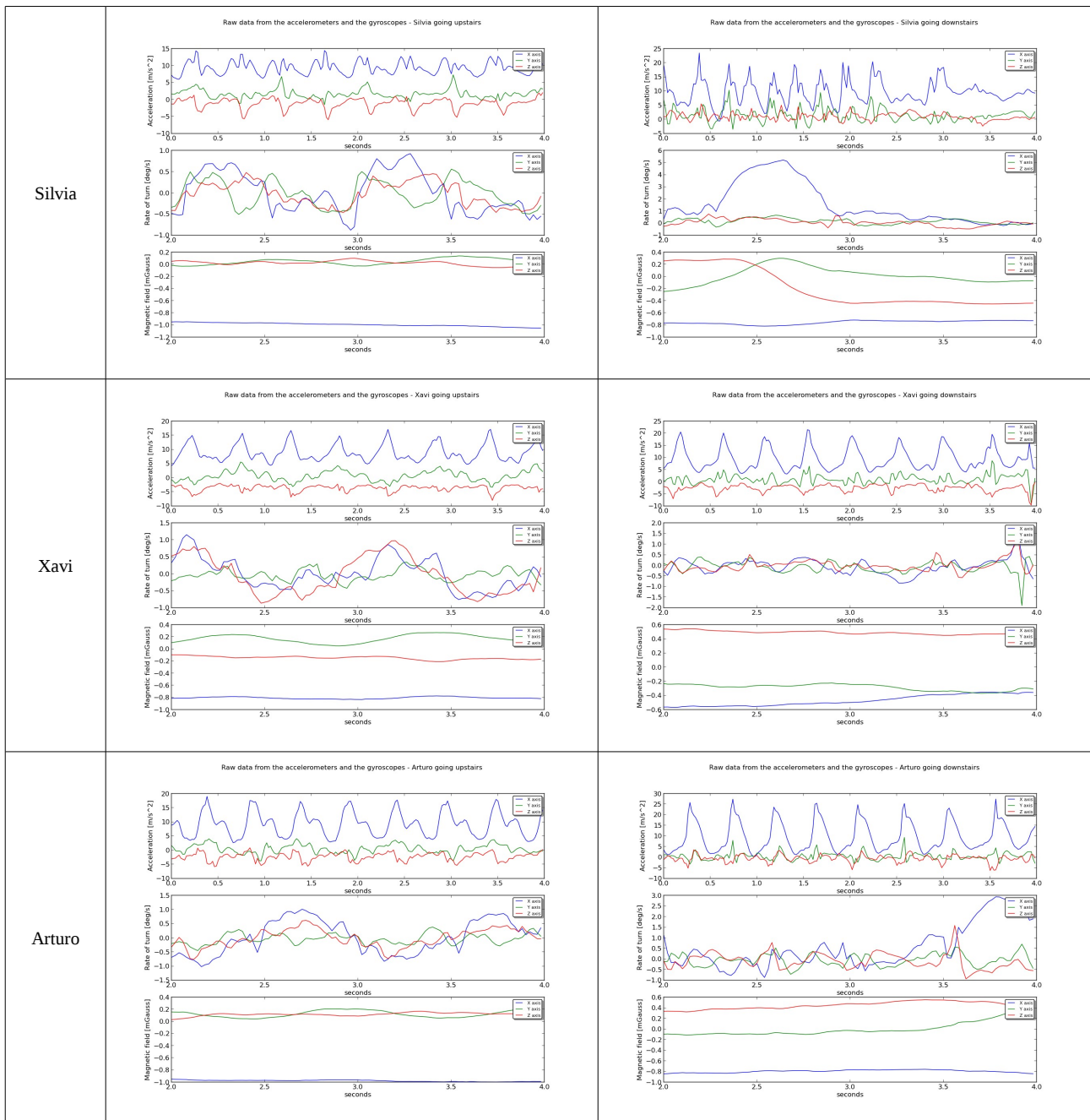


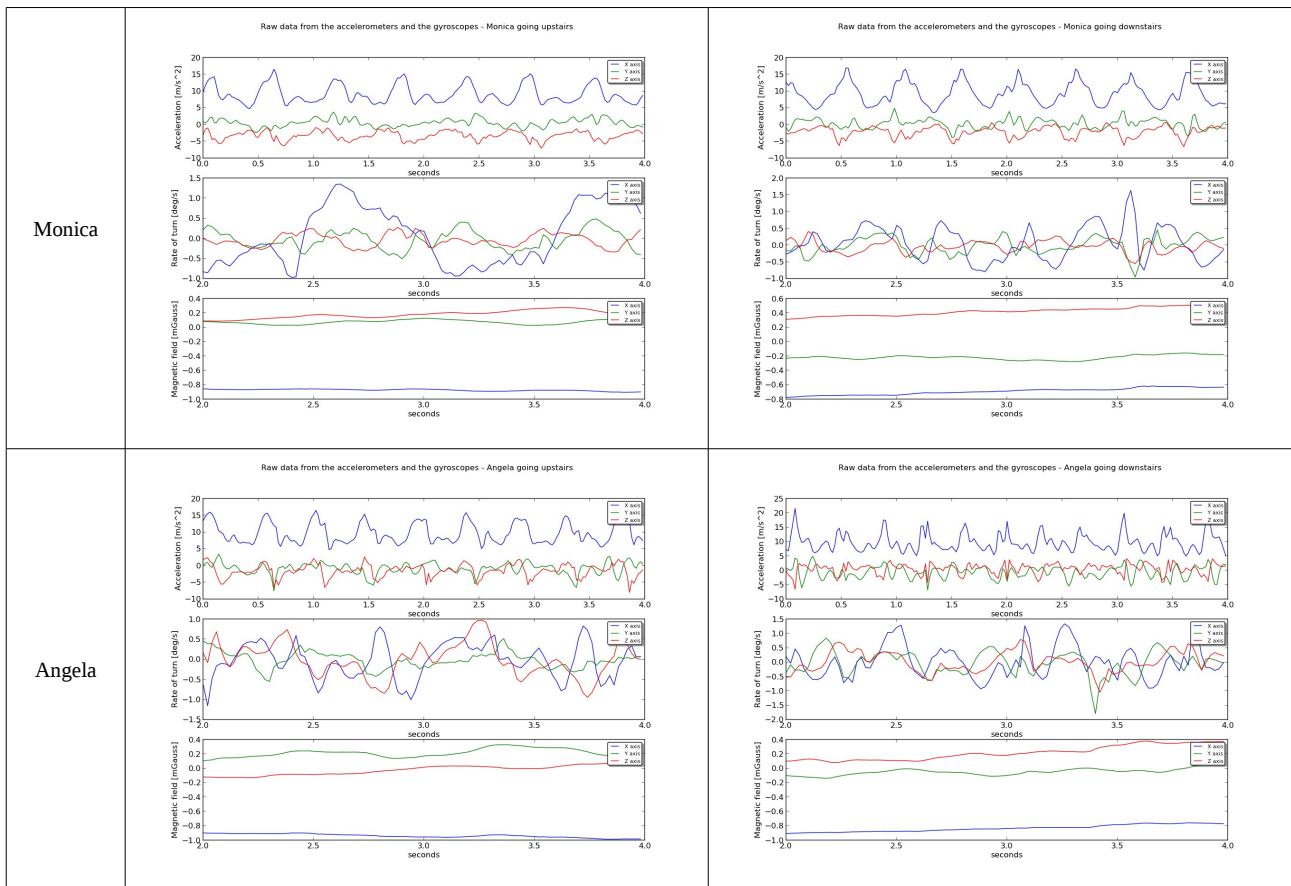




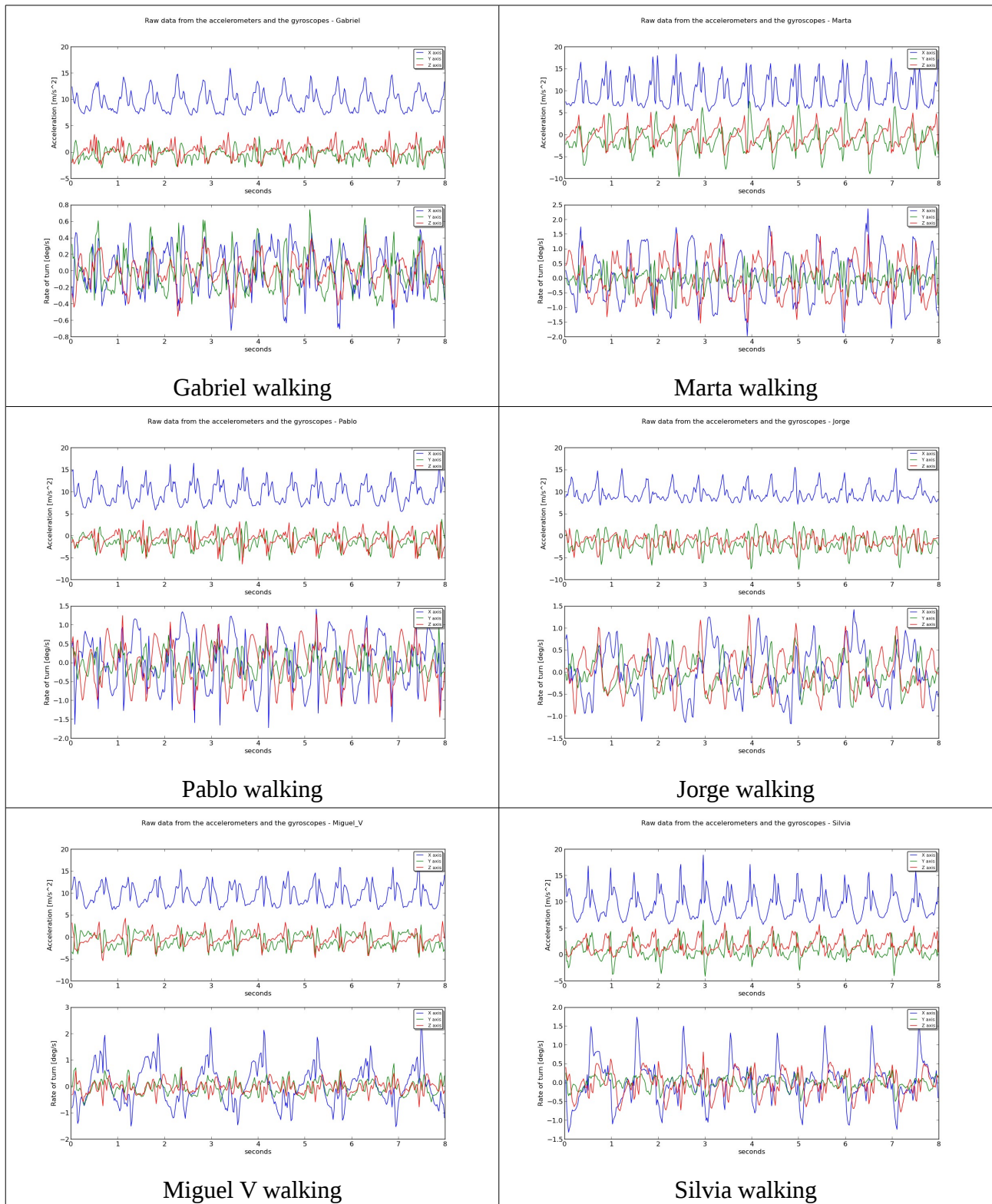


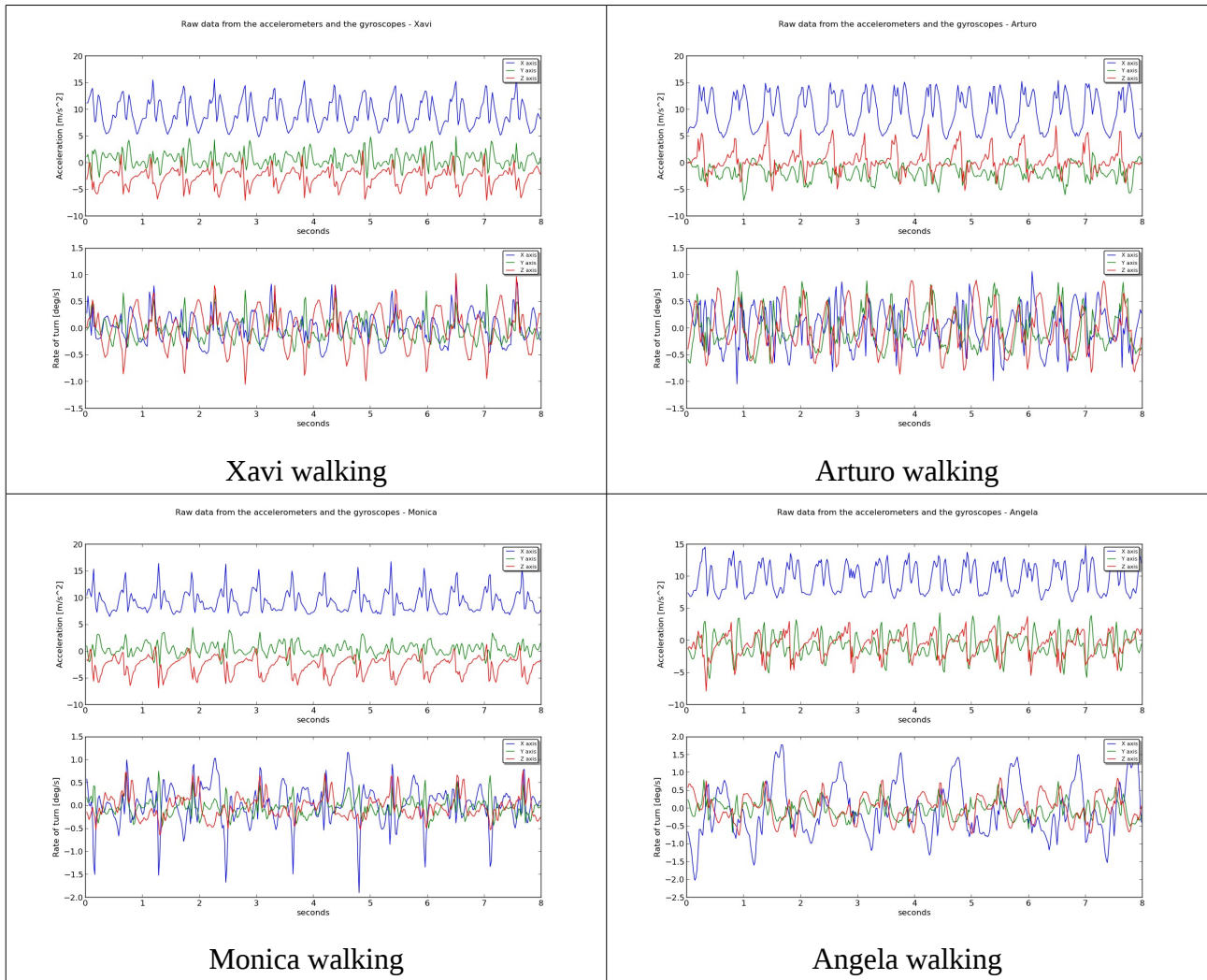






Appendix B: Data from every person while walking





Xavi walking

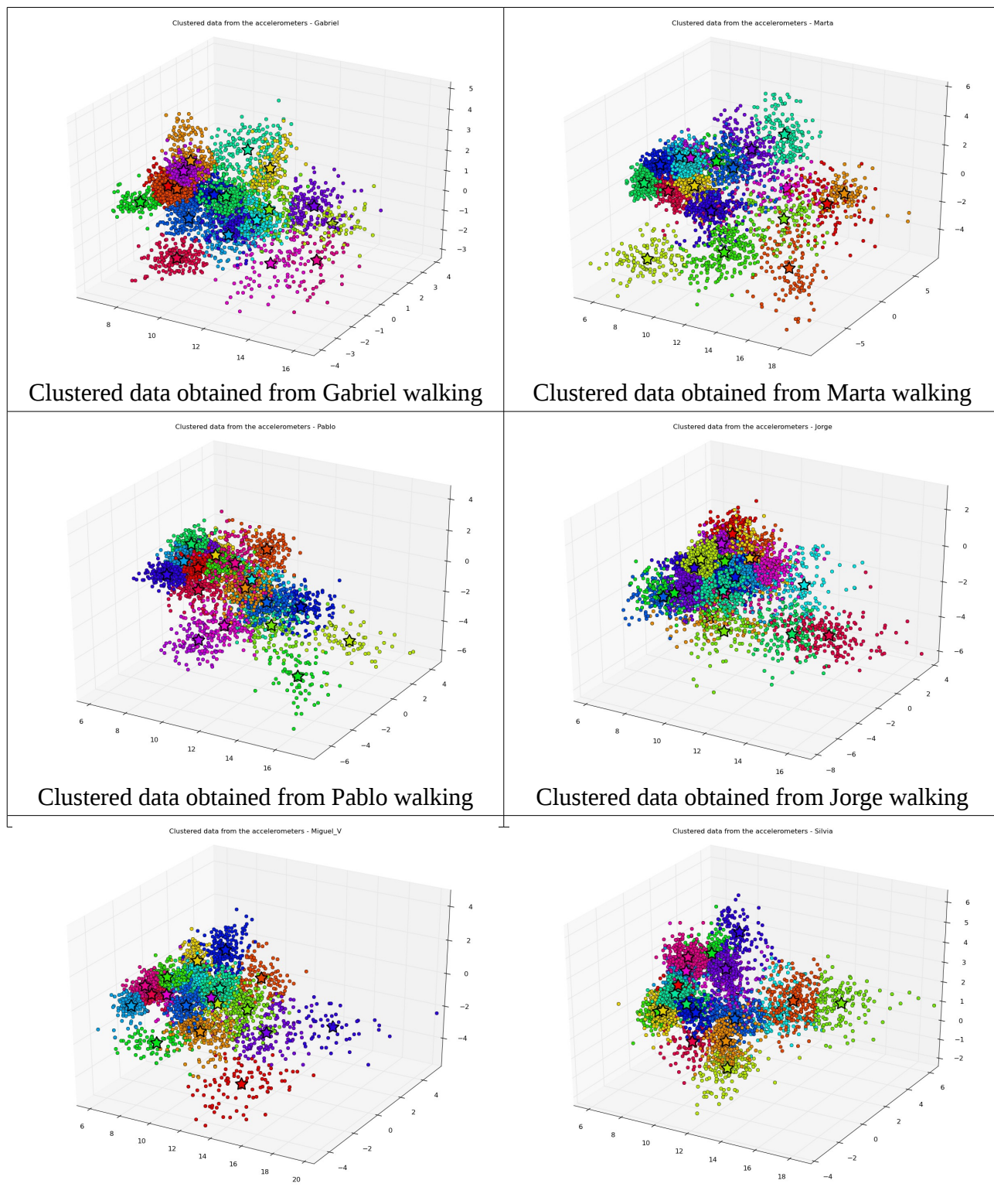
Arturo walking

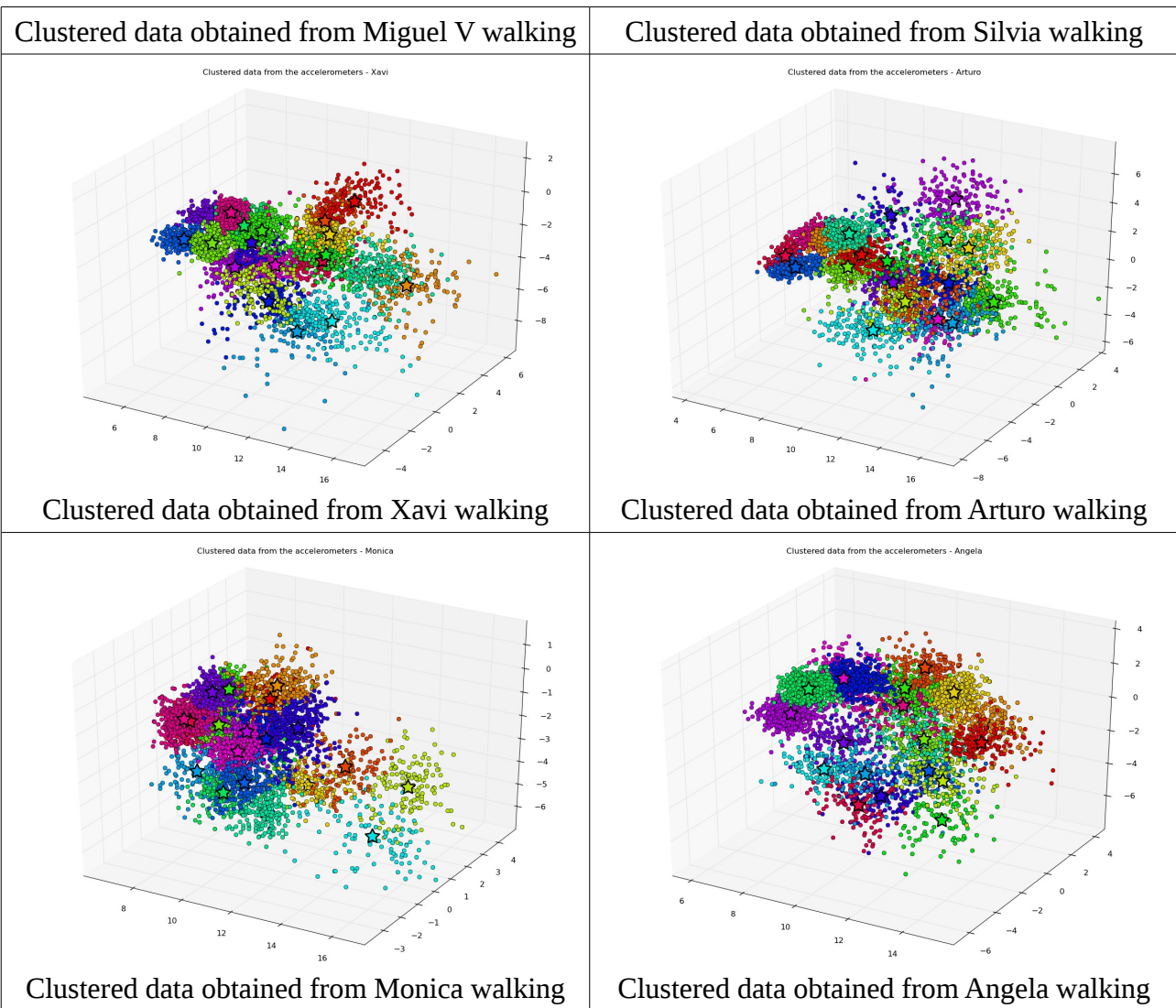
Monica walking

Angela walking

Appendix C: Clustered data from every person while walking

This is the resulting clusters of an execution of the k-means algorithm over the data of each person as they walk, with $k=20$ clusters.





Appendix D: Results of HMM models for identifying walk

| Recognition of walk | LR = 2 | | | | | LR = 3 | | | | | LR = 4 | | | | |
|---------------------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|
| | LR = 2 | M=10 | M=20 | M=30 | M=40 | LR = 3 | M=10 | M=20 | M=30 | M=40 | LR = 4 | M=10 | M=20 | M=30 | M=40 |
| Gabriel | N=10 | 100.00 | 100.00 | 100.00 | 93.33 | N=10 | 100.00 | 100.00 | 100.00 | 100.00 | N=10 | 100.00 | 100.00 | 100.00 | 100.00 |
| | N=20 | 100.00 | 100.00 | 93.33 | 53.33 | N=20 | 100.00 | 100.00 | 100.00 | 100.00 | N=20 | 100.00 | 100.00 | 86.67 | 100.00 |
| | N=30 | 100.00 | 100.00 | 86.67 | 66.67 | N=30 | 100.00 | 100.00 | 100.00 | 80.00 | N=30 | 100.00 | 100.00 | 100.00 | 93.33 |
| | N=40 | 100.00 | 100.00 | 93.33 | 100.00 | N=40 | 100.00 | 100.00 | 100.00 | 73.33 | N=40 | 100.00 | 100.00 | 100.00 | 93.33 |
| Marta | N=10 | 100.00 | 91.67 | 91.67 | 100.00 | N=10 | 100.00 | 100.00 | 100.00 | 100.00 | N=10 | 100.00 | 100.00 | 100.00 | 83.33 |
| | N=20 | 91.67 | 91.67 | 91.67 | 91.67 | N=20 | 91.67 | 91.67 | 100.00 | 91.67 | N=20 | 91.67 | 100.00 | 100.00 | 91.67 |
| | N=30 | 91.67 | 100.00 | 91.67 | 91.67 | N=30 | 100.00 | 91.67 | 100.00 | 83.33 | N=30 | 100.00 | 100.00 | 91.67 | 83.33 |
| | N=40 | 100.00 | 100.00 | 91.67 | 91.67 | N=40 | 100.00 | 100.00 | 100.00 | 91.67 | N=40 | 100.00 | 100.00 | 91.67 | 91.67 |
| Pablo | N=10 | 92.86 | 92.86 | 85.71 | 85.71 | N=10 | 100.00 | 92.86 | 92.86 | 85.71 | N=10 | 92.86 | 92.86 | 92.86 | 100.00 |
| | N=20 | 92.86 | 85.71 | 85.71 | 92.86 | N=20 | 92.86 | 85.71 | 85.71 | 64.29 | N=20 | 92.86 | 78.57 | 64.29 | 85.71 |
| | N=30 | 92.86 | 85.71 | 71.43 | 78.57 | N=30 | 92.86 | 85.71 | 85.71 | 35.71 | N=30 | 92.86 | 92.86 | 78.57 | 71.43 |
| | N=40 | 85.71 | 78.57 | 71.43 | 85.71 | N=40 | 92.86 | 92.86 | 71.43 | 50.00 | N=40 | 92.86 | 92.86 | 78.57 | 71.43 |
| Jorge | N=10 | 100.00 | 100.00 | 94.44 | 94.44 | N=10 | 100.00 | 100.00 | 100.00 | 100.00 | N=10 | 100.00 | 100.00 | 100.00 | 100.00 |
| | N=20 | 88.89 | 100.00 | 94.44 | 94.44 | N=20 | 94.44 | 94.44 | 100.00 | 100.00 | N=20 | 94.44 | 94.44 | 100.00 | 94.44 |
| | N=30 | 100.00 | 94.44 | 94.44 | 94.44 | N=30 | 94.44 | 83.33 | 88.89 | 94.44 | N=30 | 88.89 | 83.33 | 88.89 | 100.00 |
| | N=40 | 83.33 | 88.89 | 83.33 | 83.33 | N=40 | 94.44 | 88.89 | 83.33 | 88.89 | N=40 | 88.89 | 100.00 | 88.89 | 88.89 |
| Miguel V | N=10 | 100.00 | 90.91 | 90.91 | 81.82 | N=10 | 100.00 | 100.00 | 100.00 | 90.91 | N=10 | 100.00 | 100.00 | 90.91 | 100.00 |
| | N=20 | 100.00 | 90.91 | 81.82 | 63.64 | N=20 | 90.91 | 100.00 | 81.82 | 63.64 | N=20 | 100.00 | 90.91 | 72.73 | 72.73 |
| | N=30 | 81.82 | 81.82 | 90.91 | 72.73 | N=30 | 90.91 | 90.91 | 81.82 | 72.73 | N=30 | 90.91 | 81.82 | 90.91 | 81.82 |
| | N=40 | 90.91 | 81.82 | 72.73 | 90.91 | N=40 | 72.73 | 72.73 | 81.82 | 72.73 | N=40 | 90.91 | 81.82 | 81.82 | 72.73 |
| Silvia | N=10 | 100.00 | 100.00 | 87.50 | 81.25 | N=10 | 100.00 | 93.75 | 100.00 | 93.75 | N=10 | 100.00 | 100.00 | 93.75 | 100.00 |
| | N=20 | 87.50 | 100.00 | 93.75 | 100.00 | N=20 | 100.00 | 93.75 | 100.00 | 100.00 | N=20 | 100.00 | 100.00 | 100.00 | 100.00 |
| | N=30 | 87.50 | 93.75 | 100.00 | 93.75 | N=30 | 100.00 | 87.50 | 100.00 | 87.50 | N=30 | 87.50 | 93.75 | 87.50 | 93.75 |
| | N=40 | 87.50 | 93.75 | 81.25 | 93.75 | N=40 | 93.75 | 93.75 | 81.25 | 93.75 | N=40 | 87.50 | 93.75 | 81.25 | 81.25 |
| Xavi | N=10 | 100.00 | 100.00 | 100.00 | 100.00 | N=10 | 100.00 | 100.00 | 100.00 | 100.00 | N=10 | 100.00 | 100.00 | 100.00 | 94.44 |
| | N=20 | 100.00 | 100.00 | 94.44 | 100.00 | N=20 | 100.00 | 100.00 | 100.00 | 100.00 | N=20 | 100.00 | 100.00 | 100.00 | 100.00 |
| | N=30 | 100.00 | 100.00 | 94.44 | 94.44 | N=30 | 100.00 | 88.89 | 100.00 | 88.89 | N=30 | 100.00 | 100.00 | 100.00 | 100.00 |
| | N=40 | 94.44 | 94.44 | 88.89 | 94.44 | N=40 | 94.44 | 100.00 | 94.44 | 88.89 | N=40 | 94.44 | 94.44 | 100.00 | 100.00 |
| Arturo | N=10 | 100.00 | 100.00 | 100.00 | 94.12 | N=10 | 100.00 | 100.00 | 100.00 | 100.00 | N=10 | 100.00 | 100.00 | 100.00 | 88.24 |
| | N=20 | 100.00 | 94.12 | 82.35 | 82.35 | N=20 | 100.00 | 100.00 | 94.12 | 94.12 | N=20 | 100.00 | 100.00 | 82.35 | 88.24 |
| | N=30 | 100.00 | 94.12 | 82.35 | 94.12 | N=30 | 100.00 | 94.12 | 94.12 | 94.12 | N=30 | 100.00 | 100.00 | 94.12 | 94.12 |
| | N=40 | 94.12 | 100.00 | 88.24 | 76.47 | N=40 | 94.12 | 100.00 | 94.12 | 94.12 | N=40 | 100.00 | 94.12 | 94.12 | 82.35 |
| Monica | N=10 | 100.00 | 100.00 | 100.00 | 100.00 | N=10 | 100.00 | 100.00 | 100.00 | 94.74 | N=10 | 100.00 | 100.00 | 100.00 | 100.00 |
| | N=20 | 100.00 | 100.00 | 94.74 | 100.00 | N=20 | 100.00 | 100.00 | 100.00 | 94.74 | N=20 | 100.00 | 100.00 | 94.74 | 100.00 |
| | N=30 | 89.47 | 100.00 | 100.00 | 94.74 | N=30 | 100.00 | 100.00 | 94.74 | 89.47 | N=30 | 100.00 | 94.74 | 94.74 | 89.47 |
| | N=40 | 94.74 | 89.47 | 84.21 | 94.74 | N=40 | 100.00 | 89.47 | 94.74 | 89.47 | N=40 | 100.00 | 94.74 | 94.74 | 73.68 |
| Angela | N=10 | 100.00 | 100.00 | 90.00 | 95.00 | N=10 | 100.00 | 100.00 | 100.00 | 95.00 | N=10 | 100.00 | 100.00 | 100.00 | 95.00 |
| | N=20 | 100.00 | 100.00 | 95.00 | 100.00 | N=20 | 100.00 | 100.00 | 95.00 | 95.00 | N=20 | 100.00 | 100.00 | 100.00 | 100.00 |
| | N=30 | 100.00 | 95.00 | 100.00 | 85.00 | N=30 | 100.00 | 100.00 | 100.00 | 90.00 | N=30 | 100.00 | 100.00 | 100.00 | 90.00 |
| | N=40 | 100.00 | 95.00 | 100.00 | 95.00 | N=40 | 95.00 | 100.00 | 100.00 | 100.00 | N=40 | 100.00 | 100.00 | 95.00 | 95.00 |

The averaged performance of all the models for successfully recognizing walking, for each value of the variable parameters:

| LR = 2 | M=10 | M=20 | M=30 | M=40 | LR = 3 | M=10 | M=20 | M=30 | M=40 | LR = 4 | M=10 | M=20 | M=30 | M=40 |
|--------|-------|-------|-------|-------|--------|--------|-------|-------|-------|--------|-------|-------|-------|-------|
| N=10 | 99.29 | 97.54 | 94.02 | 92.57 | N=10 | 100.00 | 98.66 | 99.29 | 96.01 | N=10 | 99.29 | 99.29 | 97.75 | 96.10 |
| N=20 | 96.09 | 96.24 | 90.73 | 87.83 | N=20 | 96.99 | 96.56 | 95.67 | 90.34 | N=20 | 97.90 | 96.39 | 90.08 | 93.28 |
| N=30 | 94.33 | 94.48 | 91.19 | 86.61 | N=30 | 97.82 | 92.21 | 94.53 | 81.62 | N=30 | 96.02 | 94.65 | 92.64 | 89.73 |
| N=40 | 93.08 | 92.19 | 85.51 | 89.89 | N=40 | 93.73 | 93.77 | 90.11 | 84.28 | N=40 | 95.46 | 95.17 | 90.61 | 85.03 |

Rejection of other movements

| Rejection of other mov. | LR = 2 | | | | | LR = 3 | | | | | LR = 4 | | | | |
|-------------------------|--------|-------|--------|--------|--------|--------|-------|--------|--------|--------|--------|-------|--------|--------|--------|
| | LR = 2 | M=10 | M=20 | M=30 | M=40 | LR = 3 | M=10 | M=20 | M=30 | M=40 | LR = 4 | M=10 | M=20 | M=30 | M=40 |
| Gabriel | N=10 | 27.87 | 83.61 | 93.44 | 100.00 | N=10 | 22.95 | 72.13 | 96.72 | 96.72 | N=10 | 31.15 | 91.80 | 85.25 | 100.00 |
| | N=20 | 50.82 | 95.08 | 98.36 | 100.00 | N=20 | 36.07 | 93.44 | 100.00 | 100.00 | N=20 | 27.87 | 93.44 | 100.00 | 100.00 |
| | N=30 | 57.38 | 98.36 | 98.36 | 100.00 | N=30 | 44.26 | 100.00 | 100.00 | 100.00 | N=30 | 4.92 | 98.36 | 100.00 | 100.00 |
| | N=40 | 40.98 | 98.36 | 100.00 | 100.00 | N=40 | 24.59 | 100.00 | 100.00 | 100.00 | N=40 | 36.07 | 100.00 | 100.00 | 100.00 |
| Marta | N=10 | 61.02 | 89.83 | 83.05 | 72.88 | N=10 | 47.46 | 76.27 | 96.61 | 88.14 | N=10 | 50.85 | 88.14 | 88.14 | 98.31 |
| | N=20 | 64.41 | 93.22 | 100.00 | 94.92 | N=20 | 30.51 | 74.58 | 94.92 | 100.00 | N=20 | 71.19 | 98.31 | 96.61 | 100.00 |
| | N=30 | 61.02 | 88.14 | 98.31 | 91.53 | N=30 | 50.85 | 91.53 | 96.61 | 100.00 | N=30 | 66.10 | 76.27 | 98.31 | 96.61 |
| | N=40 | 61.02 | 91.53 | 100.00 | 94.92 | N=40 | 55.93 | 89.83 | 98.31 | 98.31 | N=40 | 69.49 | 76.27 | 100.00 | 100.00 |
| Pablo | N=10 | 64.86 | 78.38 | 100.00 | 95.95 | N=10 | 4.05 | 81.08 | 97.30 | 100.00 | N=10 | 16.22 | 82.43 | 100.00 | 100.00 |
| | N=20 | 51.35 | 93.24 | 98.65 | 95.95 | N=20 | 32.43 | 87.84 | 94.59 | 100.00 | N=20 | 54.05 | 93.24 | 97.30 | 100.00 |
| | N=30 | 67.57 | 94.59 | 100.00 | 98.65 | N=30 | 35.14 | 95.95 | 98.65 | 98.65 | N=30 | 8.11 | 79.73 | 97.30 | 100.00 |
| | N=40 | 72.97 | 98.65 | 98.65 | 97.30 | N=40 | 52.70 | 94.59 | 97.30 | 98.65 | N=40 | 62.16 | 94.59 | 100.00 | 100.00 |
| Jorge | N=10 | 29.58 | 85.92 | 100.00 | 100.00 | N=10 | 32.39 | 91.55 | 100.00 | 100.00 | N=10 | 4.23 | 100.00 | 94.37 | 100.00 |
| | N=20 | 49.30 | 92.96 | 98.59 | 100.00 | N=20 | 9.86 | 100.00 | 100.00 | 100.00 | N=20 | 16.90 | 90.14 | 100.00 | 100.00 |
| | N=30 | 45.07 | 100.00 | 100.00 | 98.59 | N=30 | 23.94 | 94.37 | 98.59 | 100.00 | N=30 | 18.31 | 94.37 | 100.00 | 100.00 |
| | N=40 | 25.35 | 97.18 | 100.00 | 100.00 | N=40 | 25.35 | 97.18 | 100.00 | 100.00 | N=40 | 49.30 | 85.92 | 100.00 | 100.00 |
| Miguel V | N=10 | 58.46 | 78.46 | 100.00 | 100.00 | N=10 | 18.46 | 96.92 | 96.92 | 100.00 | N=10 | 16.92 | 100.00 | 100.00 | 96.92 |
| | N=20 | 30.77 | 93.85 | 95.38 | 100.00 | N=20 | 47.69 | 96.92 | 100.00 | 100.00 | N=20 | 21.54 | 70.77 | 98.46 | 100.00 |
| | N=30 | 20.00 | 98.46 | 98.46 | 98.46 | N=30 | 30.77 | 95.38 | 100.00 | 100.00 | N=30 | 60.00 | 96.92 | 100.00 | 100.00 |
| | N=40 | 46.15 | 95.38 | 100.00 | 98.46 | N=40 | 41.54 | 98.46 | 100.00 | 100.00 | N=40 | 10.77 | 92.31 | 98.46 | 100.00 |
| Silvia | N=10 | 36.07 | 70.49 | 77.05 | 81.97 | N=10 | 55.74 | 54.10 | 78.69 | 90.16 | N=10 | 44.26 | 80.33 | 85.25 | 83.61 |
| | N=20 | 63.93 | 83.61 | 88.52 | 98.36 | N=20 | 59.02 | 77.05 | 85.25 | 88.52 | N=20 | 63.93 | 78.69 | 86.89 | 98.36 |
| | N=30 | 70.49 | 85.25 | 91.80 | 98.36 | N=30 | 40.98 | 93.44 | 90.16 | 100.00 | N=30 | 62.30 | 83.61 | 95.08 | 93.44 |
| | N=40 | 81.97 | 93.44 | 96.72 | 100.00 | N=40 | 63.93 | 91.80 | 95.08 | 98.36 | N=40 | 70.49 | 91.80 | 96.72 | 98.36 |
| Xavi | N=10 | 18.18 | 47.27 | 65.45 | 65.45 | N=10 | 23.64 | 65.45 | 94.55 | 87.27 | N=10 | 29.09 | 50.91 | 90.91 | 87.27 |
| | N=20 | 43.64 | 58.18 | 94.55 | 81.82 | N=20 | 21.82 | 67.27 | 92.73 | 92.73 | N=20 | 9.09 | 76.36 | 90.91 | 94.55 |
| | N=30 | 16.36 | 60.00 | 89.09 | 74.55 | N=30 | 49.09 | 78.18 | 85.45 | 98.18 | N=30 | 18.18 | 81.82 | 100.00 | 94.55 |
| | N=40 | 23.64 | 70.91 | 89.09 | 83.64 | N=40 | 30.91 | 78.18 | 83.64 | 92.73 | N=40 | 18.18 | 67.27 | 78.18 | 89.09 |
| Arturo | N=10 | 33.87 | 79.03 | 72.58 | 74.19 | N=10 | 53.23 | 59.68 | 87.10 | 95.16 | N=10 | 25.81 | 64.52 | 83.87 | 74.19 |
| | N=20 | 17.74 | 88.71 | 95.16 | 82.26 | N=20 | 58.06 | 67.74 | 96.77 | 96.77 | N=20 | 75.81 | 80.65 | 98.39 | 100.00 |
| | N=30 | 38.71 | 88.71 | 93.55 | 85.48 | N=30 | 43.55 | 69.35 | 90.32 | 96.77 | N=30 | 43.55 | 69.35 | 91.94 | 100.00 |
| | N=40 | 48.39 | 75.81 | 96.77 | 90.32 | N=40 | 30.65 | 77.42 | 90.32 | 98.39 | N=40 | 37.10 | 80.65 | 90.32 | 96.77 |
| Monica | N=10 | 55.84 | 53.25 | 87.01 | 84.42 | N=10 | 36.36 | 50.65 | 93.51 | 94.81 | N=10 | 45.45 | 85.71 | 96.10 | 96.10 |
| | N=20 | 35.06 | 80.52 | 93.51 | 90.91 | N=20 | 31.17 | 87.01 | 100.00 | 94.81 | N=20 | 38.96 | 74.03 | 89.61 | 97.40 |
| | N=30 | 55.84 | 85.71 | 96.10 | 96.10 | N=30 | 3.90 | 84.42 | 94.81 | 96.10 | N=30 | 44.16 | 77.92 | 97.40 | 98.70 |
| | N=40 | 53.25 | 93.51 | 98.70 | 98.70 | N=40 | 1.30 | 90.91 | 94.81 | 100.00 | N=40 | 50.65 | 89.61 | 97.40 | 94.81 |
| Angela | N=10 | 51.47 | 70.59 | 100.00 | 98.53 | N=10 | 47.06 | 55.88 | 100.00 | 100.00 | N=10 | 45.59 | 61.76 | 97.06 | 100.00 |
| | N=20 | 42.65 | 97.06 | 98.53 | 98.53 | N=20 | 76.47 | 95.59 | 100.00 | 100.00 | N=20 | 36.76 | 89.71 | 98.53 | 98.53 |
| | N=30 | 64.71 | 89.71 | 100.00 | 98.53 | N=30 | 72.06 | 85.29 | 94.12 | 100.00 | N=30 | 41.18 | 100.00 | 98.53 | 100.00 |
| | N=40 | 52.94 | 88.24 | 92.65 | 94.12 | N=40 | 27.94 | 92.65 | 98.53 | 100.00 | N=40 | 50.00 | 92.65 | 100.00 | 100.00 |

The averaged performance of all the models for successfully rejecting movements different from walking, for each value of the variable parameters:

| LR = 2 | M=10 | M=20 | M=30 | M=40 | LR = 3 | M=10 | M=20 | M=30 | M=40 | LR = 4 | M=10 | M=20 | M=30 | M=40 |
|--------|-------|-------|-------|-------|--------|-------|-------|-------|-------|--------|-------|-------|-------|-------|
| N=10 | 43.72 | 73.68 | 87.86 | 87.34 | N=10 | 34.13 | 70.37 | 94.14 | 95.23 | N=10 | 30.96 | 80.56 | 92.09 | 93.64 |
| N=20 | 44.97 | 87.64 | 96.13 | 94.27 | N=20 | 40.31 | 84.74 | 96.43 | 97.28 | N=20 | 41.61 | 84.53 | 95.67 | 98.88 |
| N=30 | 49.71 | 88.89 | 96.57 | 94.03 | N=30 | 39.45 | 88.79 | 94.87 | 98.97 | N=30 | 36.68 | 85.84 | 97.86 | 98.33 |
| N=40 | 50.67 | 90.30 | 97.26 | 95.75 | N=40 | 35.48 | 91.10 | 95.80 | 98.64 | N=40 | 45.42 | 87.11 | 96.11 | 97.90 |

Averaged performance

| Average performance | LR = 2 | | | | | LR = 3 | | | | | LR = 4 | | | | |
|---------------------|--------|-------|-------|--------|--------|--------|-------|--------|--------|--------|--------|-------|--------|--------|--------|
| | LR = 2 | M=10 | M=20 | M=30 | M=40 | LR = 3 | M=10 | M=20 | M=30 | M=40 | LR = 4 | M=10 | M=20 | M=30 | M=40 |
| Gabriel | N=10 | 42.11 | 86.84 | 94.74 | 98.68 | N=10 | 38.16 | 77.63 | 97.37 | 97.37 | N=10 | 44.74 | 93.42 | 88.16 | 100.00 |
| | N=20 | 60.53 | 96.05 | 97.37 | 90.79 | N=20 | 48.68 | 94.74 | 100.00 | 100.00 | N=20 | 42.11 | 94.74 | 97.37 | 100.00 |
| | N=30 | 65.79 | 98.68 | 96.05 | 93.42 | N=30 | 55.26 | 100.00 | 100.00 | 96.05 | N=30 | 23.68 | 98.68 | 100.00 | 98.68 |
| | N=40 | 52.63 | 98.68 | 98.68 | 100.00 | N=40 | 39.47 | 100.00 | 100.00 | 94.74 | N=40 | 48.68 | 100.00 | 100.00 | 98.68 |
| Marta | N=10 | 67.61 | 90.14 | 84.51 | 77.46 | N=10 | 56.34 | 80.28 | 97.18 | 90.14 | N=10 | 59.15 | 90.14 | 90.14 | 95.77 |
| | N=20 | 69.01 | 92.96 | 98.59 | 94.37 | N=20 | 40.85 | 77.46 | 95.77 | 98.59 | N=20 | 74.65 | 98.59 | 97.18 | 98.59 |
| | N=30 | 66.20 | 90.14 | 97.18 | 91.55 | N=30 | 59.15 | 91.55 | 97.18 | 97.18 | N=30 | 71.83 | 80.28 | 97.18 | 94.37 |
| | N=40 | 67.61 | 92.96 | 98.59 | 94.37 | N=40 | 63.38 | 91.55 | 98.59 | 97.18 | N=40 | 74.65 | 80.28 | 98.59 | 98.59 |
| Pablo | N=10 | 69.32 | 80.68 | 97.73 | 94.32 | N=10 | 19.32 | 82.95 | 96.59 | 97.73 | N=10 | 28.41 | 84.09 | 98.86 | 100.00 |
| | N=20 | 57.95 | 92.05 | 96.59 | 95.45 | N=20 | 42.05 | 87.50 | 93.18 | 94.32 | N=20 | 60.23 | 90.91 | 92.05 | 97.73 |
| | N=30 | 71.59 | 93.18 | 95.45 | 95.45 | N=30 | 44.32 | 94.32 | 96.59 | 88.64 | N=30 | 21.59 | 81.82 | 94.32 | 95.45 |
| | N=40 | 75.00 | 95.45 | 94.32 | 94.32 | N=40 | 59.09 | 94.32 | 93.18 | 90.91 | N=40 | 67.05 | 94.32 | 96.59 | 95.45 |
| Jorge | N=10 | 43.82 | 88.76 | 98.88 | 98.88 | N=10 | 46.07 | 93.26 | 100.00 | 100.00 | N=10 | 23.60 | 100.00 | 95.51 | 100.00 |
| | N=20 | 57.30 | 94.38 | 97.75 | 98.88 | N=20 | 26.97 | 98.88 | 100.00 | 100.00 | N=20 | 32.58 | 91.01 | 100.00 | 98.88 |
| | N=30 | 56.18 | 98.88 | 98.88 | 97.75 | N=30 | 38.20 | 92.13 | 96.63 | 98.88 | N=30 | 32.58 | 92.13 | 97.75 | 100.00 |
| | N=40 | 37.08 | 95.51 | 96.63 | 96.63 | N=40 | 39.33 | 95.51 | 96.63 | 97.75 | N=40 | 57.30 | 88.76 | 97.75 | 97.75 |
| Miguel V | N=10 | 64.47 | 80.26 | 98.68 | 97.37 | N=10 | 30.26 | 97.37 | 97.37 | 98.68 | N=10 | 28.95 | 100.00 | 98.68 | 97.37 |
| | N=20 | 40.79 | 93.42 | 93.42 | 94.74 | N=20 | 53.95 | 97.37 | 97.37 | 94.74 | N=20 | 32.89 | 73.68 | 94.74 | 96.05 |
| | N=30 | 28.95 | 96.05 | 97.37 | 94.74 | N=30 | 39.47 | 94.74 | 97.37 | 96.05 | N=30 | 64.47 | 94.74 | 98.68 | 97.37 |
| | N=40 | 52.63 | 93.42 | 96.05 | 97.37 | N=40 | 46.05 | 94.74 | 97.37 | 96.05 | N=40 | 22.37 | 90.79 | 96.05 | 96.05 |
| Silvia | N=10 | 49.35 | 76.62 | 79.22 | 81.82 | N=10 | 64.94 | 62.34 | 83.12 | 90.91 | N=10 | 55.84 | 84.42 | 87.01 | 87.01 |
| | N=20 | 68.83 | 87.01 | 89.61 | 98.70 | N=20 | 67.53 | 80.52 | 88.31 | 90.91 | N=20 | 71.43 | 83.12 | 89.61 | 98.70 |
| | N=30 | 74.03 | 87.01 | 93.51 | 97.40 | N=30 | 53.25 | 92.21 | 92.21 | 97.40 | N=30 | 67.53 | 85.71 | 93.51 | 93.51 |
| | N=40 | 83.12 | 93.51 | 93.51 | 98.70 | N=40 | 70.13 | 92.21 | 92.21 | 97.40 | N=40 | 74.03 | 92.21 | 93.51 | 94.81 |
| Xavi | N=10 | 38.36 | 60.27 | 73.97 | 73.97 | N=10 | 42.47 | 73.97 | 95.89 | 90.41 | N=10 | 46.58 | 63.01 | 93.15 | 89.04 |
| | N=20 | 57.53 | 68.49 | 94.52 | 86.30 | N=20 | 41.10 | 75.34 | 94.52 | 94.52 | N=20 | 31.51 | 82.19 | 93.15 | 95.89 |
| | N=30 | 36.99 | 69.86 | 90.41 | 79.45 | N=30 | 61.64 | 80.82 | 89.04 | 95.89 | N=30 | 38.36 | 86.30 | 100.00 | 95.89 |
| | N=40 | 41.10 | 76.71 | 89.04 | 86.30 | N=40 | 46.58 | 83.56 | 86.30 | 91.78 | N=40 | 36.99 | 73.97 | 83.56 | 91.78 |
| Arturo | N=10 | 48.10 | 83.54 | 78.48 | 78.48 | N=10 | 63.29 | 68.35 | 89.87 | 96.20 | N=10 | 41.77 | 72.15 | 87.34 | 77.22 |
| | N=20 | 35.44 | 89.87 | 92.41 | 82.28 | N=20 | 67.09 | 74.68 | 96.20 | 96.20 | N=20 | 81.01 | 84.81 | 94.94 | 97.47 |
| | N=30 | 51.90 | 89.87 | 91.14 | 87.34 | N=30 | 55.70 | 74.68 | 91.14 | 96.20 | N=30 | 55.70 | 75.95 | 92.41 | 98.73 |
| | N=40 | 58.23 | 81.01 | 94.94 | 87.34 | N=40 | 44.30 | 82.28 | 91.14 | 97.47 | N=40 | 50.63 | 83.54 | 91.14 | 93.67 |
| Monica | N=10 | 64.58 | 62.50 | 89.58 | 87.50 | N=10 | 48.96 | 60.42 | 94.79 | 94.79 | N=10 | 56.25 | 88.54 | 96.88 | 96.87 |
| | N=20 | 47.92 | 84.38 | 93.75 | 92.71 | N=20 | 44.79 | 89.58 | 100.00 | 94.79 | N=20 | 51.04 | 79.17 | 90.63 | 97.92 |
| | N=30 | 62.50 | 88.54 | 96.87 | 95.83 | N=30 | 22.92 | 87.50 | 94.79 | 94.79 | N=30 | 55.21 | 81.25 | 96.88 | 96.88 |
| | N=40 | 61.46 | 92.71 | 95.83 | 97.92 | N=40 | 20.83 | 90.63 | 94.79 | 97.92 | N=40 | 60.42 | 90.63 | 96.88 | 90.62 |
| Angela | N=10 | 62.50 | 77.27 | 97.73 | 97.73 | N=10 | 59.09 | 65.91 | 100.00 | 98.86 | N=10 | 57.95 | 70.45 | 97.73 | 98.86 |
| | N=20 | 55.68 | 97.73 | 97.73 | 98.86 | N=20 | 81.82 | 96.59 | 98.86 | 98.86 | N=20 | 51.14 | 92.05 | 98.86 | 98.86 |
| | N=30 | 72.73 | 90.91 | 100.00 | 95.45 | N=30 | 78.41 | 88.64 | 95.45 | 97.73 | N=30 | 54.55 | 100.00 | 98.86 | 97.73 |
| | N=40 | 63.64 | 89.77 | 94.32 | 94.32 | N=40 | 43.18 | 94.32 | 98.86 | 100.00 | N=40 | 61.36 | 94.32 | 98.86 | 98.86 |

The average performance of all the models for each value of the variable parameters:

| LR = 2 | M=10 | M=20 | M=30 | M=40 | LR = 3 | M=10 | M=20 | M=30 | M=40 | LR = 4 | M=10 | M=20 | M=30 | M=40 |
|--------|-------|-------|-------|-------|--------|-------|-------|-------|-------|--------|-------|-------|-------|-------|
| N=10 | 55.02 | 78.69 | 89.35 | 88.62 | N=10 | 46.89 | 76.25 | 95.22 | 95.51 | N=10 | 44.32 | 84.62 | 93.35 | 94.22 |
| N=20 | 55.10 | 89.63 | 95.17 | 93.31 | N=20 | 51.48 | 87.27 | 96.42 | 96.29 | N=20 | 52.86 | 87.03 | 94.85 | 98.01 |
| N=30 | 58.68 | 90.31 | 95.69 | 92.84 | N=30 | 50.83 | 89.66 | 95.04 | 95.88 | N=30 | 48.55 | 87.69 | 96.96 | 96.86 |
| N=40 | 59.25 | 90.97 | 95.19 | 94.73 | N=40 | 47.23 | 91.91 | 94.91 | 96.12 | N=40 | 55.35 | 88.88 | 95.29 | 95.63 |

Appendix E: Source Code

test.m

```

clear
clc

data_width = 60;
movement_name="walk";
data_path="entrenamiento/Angela";
train_files=["andar1"; "andar2"; "andar3"];
test_files_yes=["andar4"];
test_files_no=["bajarescaleras1";"bajarescaleras2";"bajarrampa1";"bajarrampa2";"
    correr1";"correr2";"subirescaleras1";"Angela/subirescaleras2";"subirrampa1"
    ;"subirrampa2"];

D = 3; % number of dimensions
M = 40; % output symbols
N = 20; % internal states
LR = 3; % degree of play in the left-to-right HMM transition matrix

fprintf('\n\n***** NEW EXPERIMENT: D = %i , N = %i , M = %i, LR =
    %i\n\n',D,N,M,LR);

printf('> Loading...\n')

training = get_data(data_path,train_files,data_width);

%*****
%
% Initialization
%
%*****

printf('> Initializing...\n')

RecThreshold = 0; % set below

[centroids N] = get_point_centroids(training,N,D);
ATrainBinned = get_point_clusters(training,centroids,D);

%*****
%
% Training
%
%*****

printf('> Training...\n')

% Set priors:
pP = prior_transition_matrix(M,LR);

% Train the model:
cyc = 100;

```

```

[E,P,Pi,LL] = dhmm_numeric(ATrainBinned,pP,[1:N]',M,cyc,.00001);

%*****
%
% Testing
%
%*****

test_y=[];
test_n=[];

printf('\n> Testing same movement...\n')

for num = 1:size(test_files_yes)(1)
    filename=test_files_yes(num,:);
    printf('\nData file: %s\n\n', filename)

    testing = get_data(data_path,filename,data_width);
    ATestBinned = get_point_clusters(testing,centroids,D);

    sumLik = 0;
    minLik = Inf;
    for j=1:length(ATrainBinned)
        lik = pr_hmm(ATrainBinned{j},P,E',Pi);
        if (lik < minLik)
            minLik = lik;
        endif
        sumLik = sumLik + lik;
    end
    RecThreshold = 2.0*sumLik/length(ATrainBinned);

    fprintf('Testing %i sequences for a log likelihood greater than
        %f\n',length(ATestBinned),RecThreshold);

    recs = 0;
    tLL = zeros(length(ATestBinned),1);
    for j=1:length(ATestBinned)
        tLL(j,1) = pr_hmm(ATestBinned{j},P,E',Pi);
        if (tLL(j,1) > RecThreshold)
            recs = recs + 1;
            fprintf('Log likelihood: %f > %f (threshold) -- FOUND %s
                MOVEMENT!\n',tLL(j,1),RecThreshold,movement_name);
        else
            fprintf('Log likelihood: %f < %f (threshold) -- NO %s
                MOVEMENT.\n',tLL(j,1),RecThreshold,movement_name);
        endif
    endfor
    fprintf('Recognition success rate: %f
        percent\n',100*recs/length(ATestBinned));
    test_y=[test_y 100*recs/length(ATestBinned)];
endfor

printf('\n> Testing other movements...\n')

for num = 1:size(test_files_no)(1)

```

```

filename=test_files_no(num,:);
printf('\nData file: %s\n\n', filename)

testing = get_data(data_path,filename,data_width);
ATestBinned = get_point_clusters(testing,centroids,D);

sumLik = 0;
minLik = Inf;
for j=1:length(ATrainBinned)
    lik = pr_hmm(ATrainBinned{j},P,E',Pi);
    if (lik < minLik)
        minLik = lik;
    endif
    sumLik = sumLik + lik;
endfor
RecThreshold = 2.0*sumLik/length(ATrainBinned);

fprintf('Testing %i sequences for a log likelihood greater than
%f\n',length(ATestBinned),RecThreshold);

recs = 0;
tLL = zeros(length(ATestBinned),1);
for j=1:length(ATestBinned)
    tLL(j,1) = pr_hmm(ATestBinned{j},P,E',Pi);
    if (tLL(j,1) > RecThreshold)
        recs = recs + 1;
        fprintf('Log likelihood: %f > %f (threshold) -- FOUND %s
MOVEMENT!\n',tLL(j,1),RecThreshold,movement_name);
    else
        fprintf('Log likelihood: %f < %f (threshold) -- NO %s
MOVEMENT.\n',tLL(j,1),RecThreshold,movement_name);
    endif
endfor
fprintf('Recognition success rate: %f
percent\n',100*recs/length(ATestBinned));
test_n=[test_n 100*recs/length(ATestBinned)];
endfor

```

get_data.m

```

% Get sequences of data from a log file

function data = get_data(data_path,file_list,data_width)

x=[];
y=[];
z=[];

for num = 1:size(file_list)(1)
    filename=strcat(data_path, filesep, file_list(num,:), '.log');
    filedata=dlmread(filename,"\t");
    while (length(filedata)>=data_width)
        first_chunk=filedata(1:data_width,[2 3 4]);
        filedata=filedata(data_width:end,:);
        x=[x; first_chunk(:,1)'];
        y=[y; first_chunk(:,2)'];
    end
end

```



```

        z=[z; first_chunk(:,3)'];
    endwhile
endfor

```

```
data = cat(3, x', y', z');
```

get_point_centroids.m

```
% Create histogram to bin the data
```

```
function [centroids K] = get_point_centroids(data,K,D)
```

```

mean = zeros(size(data,1),D);
for n = 1:size(data,1)
    for i = 1:size(data,2)
        for j = 1:D
            mean(n,j) = mean(n,j) + data(n,i,j);
        end
    end
end
mean(n,:) = mean(n,:)./size(data,2);
end

```

```

% Using k-means to make data discrete
[centroids,points,idx] = kmeans(mean,K);
K = size(centroids,1);

```

get_point_clusters.m

```
% Create histogram to bin the data
```

```
function XClustered = get_point_clusters(data,centroids,D)
```

```

XClustered = cell(size(data,2),1);
K = size(centroids,1);

```

```

for n = 1:size(data,1)
    for i = 1:size(data,2)
        temp = zeros(K,1);
        for j = 1:K
            if (D==3)
                temp(j) = sqrt( (centroids(j,1) - data(n,i,1))^2+(centroids(j,2)
- data(n,i,2))^2+(centroids(j,3) - data(n,i,3))^2);
            end
        end
        [idx,I] = min(temp);
        XClustered{i}(n,1) = I(1);
    end
end
end

```

kmeans.m

```

%-----
%
% K-means via Christian Herta (http://www.christianherta.de/kmeans.html)
%
%-----

```

```

% function[centroid, pointsInCluster, assignment]= kmeans(data, nbCluster)

% usage
% function[centroid, pointsInCluster, assignment]=
% kmeans(data, nbCluster)
%
% Output:
% centroid: matrix in each row are the Coordinates of a centroid
% pointsInCluster: row vector with the nbDatapoints belonging to
% the centroid
% assignment: row Vector with clusterAssignment of the dataRows
%
% Input:
% data in rows
% nbCluster : nb of centroids to determine
%
% Based on code by Christian Herta:
% http://www.christianherta.de/kmeans.html

function[centroid, pointsInCluster, assignment]= kmeans(data, nbCluster)

data_dim = length(data(1,:));
nbData   = length(data(:,1));

% init the centroids randomly
data_min = min(data);
data_max = max(data);
data_diff = data_max .- data_min ;
% every row is a centroid
centroid = ones(nbCluster, data_dim) .* rand(nbCluster, data_dim);
for i=1 : 1 : length(centroid(:,1))
    centroid( i , : ) = centroid( i , : ) .* data_diff;
    centroid( i , : ) = centroid( i , : ) + data_min;
end
% end init centroids

% no stopping at start
pos_diff = 1.;

% main loop until
while pos_diff > 0.0

    % E-Step
    assignment = [];
    % assign each datapoint to the closest centroid
    for d = 1 : length( data(:, 1) );

        min_diff = ( data( d, :) .- centroid( 1,:) );
        min_diff = min_diff * min_diff';
        curAssignment = 1;

        for c = 2 : nbCluster;
            diff2c = ( data( d, :) .- centroid( c,:) );
            diff2c = diff2c * diff2c';

```

```

        if( min_diff >= diff2c)
            curAssignment = c;
            min_diff = diff2c;
        end
    end

    % assign the d-th dataPoint
    assignment = [ assignment; curAssignment];

end

% for the stoppingCriterion
oldPositions = centroid;

% M-Step
% recalculate the positions of the centroids
centroid = zeros(nbCluster, data_dim);
pointsInCluster = zeros(nbCluster, 1);

for d = 1: length(assignment);
    centroid( assignment(d), :) += data(d, :);
    pointsInCluster( assignment(d), 1 )++;
end

for c = 1: nbCluster;
    if( pointsInCluster(c, 1) != 0)
        centroid( c , : ) = centroid( c , : ) / pointsInCluster(c, 1);
    else
        % set cluster randomly to new position
        centroid( c , : ) = (rand( 1, data_dim) .* data_diff) + data_min;
    end
end

% stoppingCriterion
pos_diff = sum( sum( (centroid .- oldPositions).^2 ) );

end

```

prior_transition_matrix.m

```

% Create a prior for the transition matrix

function P = prior_transition_matrix(K, LR)

% LR is the allowable number of left-to-right transitions

P = ((1/LR))*eye(K);

for i=1:K
    for j=1:LR-1
        k=i+j;
        if (k>K)
            k=k-K;
        endif
        P(i,k) = 1/LR;
    endfor
endfor

```

endfor

dhmm_numeric.m

```
% function [E,P,Pi,LL]=dhmm_numeric(X,alphabet,K,cyc,tol);
%
% simple Hidden Markov Model - variable lengths and discrete symbols
%
% X - N x p array of Xs, p sequences
% bins - possible bins of Xs
% K - number of states (default 2)
% cyc - maximum number of cycles of Baum-Welch (default 100)
% tol - termination tolerance (prop change in likelihood) (default 0.0001)
%
% E - observation emission probabilities
% P - state transition probabilities
% Pi - initial state prior probabilities
% LL - log likelihood curve
%
% Iterates until a proportional change < tol in the log likelihood
% or cyc steps of Baum-Welch
%
% Adapted from Zoubin Ghahramani:
%   http://mlg.eng.cam.ac.uk/zoubin/software.html

function [E,P,Pi,LL]=dhmm_numeric(X,pP,bins,K,cyc,tol)

D = size(X,2);
num_bins = size(bins,1);

epsi = 1e-10;

% number of sequences
N = size(X,1);

% length of each sequence
T = ones(1,N);
for n=1:N,
    T(n)=size(X{n},1);
end;

TMAX = max(T);

if nargin<6    tol=0.0001; end;
if nargin<5    cyc=100; end;
if nargin<4    K=2; end;

fprintf('\n*****\n');
fprintf('Training %i sequences of maximum length %i from an alphabet of size\n',N,TMAX,num_bins);
fprintf('HMM with %i hidden states\n',K);
fprintf('*****\n');

E = (0.1*rand(num_bins,K)+ones(num_bins,K))/num_bins;
```

```

E = cdiv(E,csum(E));
%E = (1/num_bins).*ones(num_bins,K);
B=zeros(TMAX,K);
Pi= rand(1,K);
Pi=Pi/sum(Pi);
% transition matrix
P = pP;
P=rdiv(P,rsum(P));
% This is useful if the transition matrix is initialized with many zeros
% i.e. for a left-to-right HMM
P=sparse(P);
LL=[];
lik=0;
for cycle=1:cyc
    %%%% FORWARD-BACKWARD
    Gammainit=zeros(1,K);
    Gammasum=zeros(1,K);
    Gammaksum = zeros(num_bins,K);
    Scale=zeros(TMAX,1);
    sxi=sparse(K,K);
    for n=1:N
        alpha=zeros(T(n),K);
        beta=zeros(T(n),K);
        gamma=zeros(T(n),K);
        gammaksum = zeros(num_bins,K);
        % Inital values of B = Prob(output|s_i), given data X
        Xcurrent=X{n};
        for i=1:T(n)
            %for j=1:D
            % find the letter in the alphabet
            % m = findstr(alphabet,Xcurrent(i));
            m = find(bins==Xcurrent(i));
            if (m == 0)
                fprintf('Error: Symbol not found\n');
                return;
            end
            B(i,:) = E(m,:);
        %end
        end;
        scale=zeros(T(n),1);
        alpha(1,:)=Pi(:)'.*B(1,:);
        scale(1)=sum(alpha(1,:));
    end
end

```

```

alpha(1, :)=alpha(1, :)/scale(1);
for i=2:T(n)
    alpha(i, :)=(alpha(i-1, :)*P).*B(i, :);
    scale(i)=sum(alpha(i, :));
    alpha(i, :)=alpha(i, :)/scale(i);
end;

beta(T(n), :)=ones(1,K)/scale(T(n));
for i=T(n)-1:-1:1
    beta(i, :)=(beta(i+1, :).*B(i+1, :))*(P')/scale(i);
end;

gamma=(alpha.*beta)+epsi;
gamma=rdiv(gamma, rsum(gamma));

gammасum=sum(gamma);

for i = 1:T(n)
    % find the letter in the alphabet
    % m = findstr(alphabet, Xcurrent(i));
    m = find(bins==Xcurrent(i));
    gammасum(m, :) = gammасum(m, :) + gamma(i, :);
end;

for i=1:T(n)-1
    t=P.*( alpha(i, :)' * (beta(i+1, :).*B(i+1, :)));
    sxi=sxi+t/sum(t(:));
end;

Gammainit=Gammainit+gamma(1, :);
Gammасum=Gammасum+gammасum;
Gammасum = Gammасum + gammасum;

for i=1:T(n)-1
    Scale(i, :) = Scale(i, :) + log(scale(i, :));
end;
Scale(T(n), :) = Scale(T(n), :) + log(scale(T(n), :));

end;

%%%% M STEP

% outputs

E = cdiv(Gammасum, Gammасum);

% transition matrix
P = sparse(rdiv(sxi, rsum(sxi)));
P = P*eye(size(P,1));

% priors
Pi=Gammainit/sum(Gammainit);

oldlik=lik;
lik=sum(Scale);
LL=[LL lik];

```

```

fprintf('\ncycle %i log likelihood = %f ',cycle,lik);
if (cycle<=2)    likbase=lik;
elseif (lik<(oldlik - 1e-6))    fprintf('vionum_binstion');
elseif ((lik-likbase)<(1 + tol)*(oldlik-likbase)||~isfinite(lik))
    fprintf('\nend\n');    return;
end;

```

end

pr_hmm.m

```

% function p=pr_hmm(o,a,b,pi)
%
% INPUTS:
% O=Given observation sequence labeled in numerics
% A(N,N)=transition probability matrix
% B(N,M)=Emission matrix
% pi=initial probability matrix
% Output
% P=probability of given sequence in the given model

function p=pr_hmm(o,a,b,pi)

n=length(a(1,:));
T=length(o);

% it uses forward algorithm to compute the probability

for i=1:n    %it is initialization
    m(1,i)=b(i,o(1))*pi(i);
end
for t=1:(T-1)    %recursion
    for j=1:n
        z=0;
        for i=1:n
            z=z+a(i,j)*m(t,i);
        end
        m(t+1,j)=z*b(j,o(t+1));
    end
end
end
p=0;
for i=1:n    %termination
    p=p+m(T,i);
end
p=log(p);

```

csum.m

```

% column sum
% function Z=csum(X)

function Z=csum(X)

N=length(X(:,1));
if (N>1)

```

```
Z=sum(X);  
else  
    Z=X;  
end;
```

cdiv.m

```
% column division  
% function Z=cdiv(X,Y)  
  
function Z=cdiv(X,Y)  
  
if(length(X(1,:)) ~= length(Y(1,:)) | length(Y(:,1)) ~=1)  
    disp('Error in CDIV');  
    return;  
end  
  
Z=zeros(size(X));  
  
for i=1:length(X(:,1))  
    Z(i,:)=X(i,:)./Y;  
end
```

rsum.m

```
% row sum  
% function Z=rsum(X)  
  
function Z=rsum(X)  
  
[N M]=size(X);  
  
Z=zeros(N,1);  
  
if M==1,  
    Z=X;  
elseif M<2*N,  
    for m=1:M,  
        Z=Z+X(:,m);  
    end;  
else  
    for n=1:N  
        Z(n)=sum(X(n,:));  
    end;  
end
```

rdiv.m

```
% function Z=rdiv(X,Y)  
%  
% row division: Z = X / Y row-wise  
% Y must have one column  
  
function Z=rdiv(X,Y)  
  
[N M]=size(X);
```



```
[K L]=size(Y);
if(N ~= K | L ~=1)
    disp('Error in RDIV');
    return;
end

Z=zeros(N,M);

if M<N,
    for m=1:M
        Z(:,m)=X(:,m)./Y;
    end
else
    for n=1:N
        Z(n,:)=X(n,:)/Y(n);
    end;
end;
```