UNIVERSITA' MILANO-BICOCCA
FACOLTÀ DI SCIENZE MATEMATICHE FISICHE E NATURALI
DIPARTIMENTO DI INFORMATICA SISTEMISTICA E COMUNICAZIONE
SCUOLA DI DOTTORATO IN INFORMATICA

Movements Recognition with Intelligent Multisensor Analysis

From sensors to meaning

Discussion Proposed for the Doctoral Thesis in Computer Science by S. Pinardi

Supervisor: Prof.R.Bisiani
Tutor: Prof. R. Schettini
Supervisor of the Ph.D. program: Prof. S.Bandini

Ph.D. XXIII Cycle - 2010

Index

Preface

"Many atoms disintegrate themselves spontaneously through the emission of an electron or an alpha particle (i.e. *He* nucleus). This is what we usually call the phenomenon of radioactivity discovered by H. Becquerel in 1896 [57]. The disintegration rate is governed by a statistical law that states that given N_i atoms at the time i, the number of surviving atoms at time t is $N(t) = N_i \exp(-\lambda t)$. The λ constant is a characteristic of the intended atom (or nucleus). This law provide us with an useful model to predict radioactivity intensity in the future (or in the past, thinking to the C14 dating method used by anthropologists) and to anticipatory know what will be the quantity of remained atoms of a radiating matter in close or far future (something of interest to every scientist of atoms). For example we know that atoms will be the half exactly at the time $T = (1 / \lambda) \ln(2)$. Once calibrated the parameter λ everything is known."

In movement science we do not have any generic model that could be considered as precise nor accurate for movement classification. Many different explanations of specific aspects of movement have been proposed. They are very interesting, but none of them are generally explicative of what is going on in a semantic sense [6,7,8,10,11,15,20,21, 22,25,27,34,47]. When we deal with the movement recognition/classification area we do not even have a satisfying model of specific aspects that can be considered generally predictive [3,23,26,28,29,33,35,37,38,39]. And in the area of movement recognition with inertial sensors, many technological questions arise: technological diversity, calibration matters, sensors errors, sensor models, orientation and position of sensor in space, and a lot of numerous specificities that, with all the above aspects, and the lack of sufficiently generic and semantically rich public Test Set, contribute to create a strong barrier to any generic approach to a movement classification with wearable sensors at the moment (so far) [3,4,5,23,26,29,30,37].

We have also to note that a movement is a phenomenon explicitly or implicitly (voluntary or involuntary) controlled by brain. The human and the individual free-will introduce a further problem when we want to temporary predict the movements looking at the close past [1,23,26]. Any patterns can change when you can change situation, ambient, psychological context, age, physiological o pathological changes in the body, attitude and will of the subject.

On the other hand machine learning techniques seems quite promising [3, 37], but for some reasons they still lack generality, as far I can see, and are not intended to solve the specifical matters that arise in movement classification area with inertial sensor even in works and research of area specialists [5, 23].

Also a movement, an action, is something that is semantically undefined. Everyone intuitively knows what is a movement, or an action but when we have to observatively say what a person is doing, when we naturally speaking classify a movement, we have to define well what is a movement, when it starts when it ends, and when - this is a difficult problem – we can consider two movements different, in the sense they pertains to different movements classes [16,17,18,19,23,24].

To a normal eye two people walking straight in an aisle for 10 steps are doing the same actions. But slight differences appear (can be evident) to the eye of a physicians or a physiotherapist, that mark a difference between a physiological movement and a pathological one [12,13,31,36,56], and the two actions that we so far considered to pertain to to same class, suddenly happened to fall in two distinctively different categories.

Semantics matters, engineering questions, quality of outcome information, quality of incoming data, lack of general models and the necessity to give a generic explanation, or method, valid in the information area, are the questions that have been considered, and faced, that in my opinion create a factual and conceptual barrier to any innovative approach (scientific improvements) in the area. *The lack of generality is the main barrier*, the lack of a model extends the problem and arise some interesting question like what is the *semantics* of a movement, and the technological question does not help to search for a generic solution useful in different domains or situations.

For all these reasons I considered that a semantical/lexical approach to movement recognition with sensors, using on instance based machine learning techniques could be a promising way to solve some of these challenges and problems. And so it was.

I was initially inspired the by works of G.Guerra-Filho and Y.Aloimonos that - going through other works in the field of medicine and biology and neurology - propose a correlation between the motor grammar and the natural language grammars [17,18,24,28,40]. Their work is specific to a programmatic grammar (something that could be mechanically handled) and does not try to understand how this grammar could be created in the human brain, nor how it could be present and why, but the idea behind was connaturate to my first direct observation and seems interesting.

A language is a complex object, with information about future and past, where relations between subject and objects are described, and where emotions and metaphors are present. And a lot of other questions that are still behind our comprehension.

The main idea is that people share a common representation of the reality through the language. But what do they share? What they are talking about: information, emotions, facts or what else? A conceptual space is possibly involved? And if something like “a grammar of movements” really exists what correlations are present between this grammar and the natural language, and which are the differences?

Talking about semantic of movements you can represents – and consequently share – information about actions. You can classify things like “taking something”, “moving through a space”, “shaking hands”, and maybe even you can classify information like “to act as an happy person”, “to appear sad”, “to be nervous”. Connecting them you can create a sequence of phrases, like in a natural language that represents what is going on in the reality, a concrete representations of reality. You cannot depict anything like “huge as the ocean” or “larger than life”: methaphors are absent in the “movement language”. Metaphors are probably a specifity of the natural language. But some aspects are common to movements and the natural language. For example the lexicon.

Before creating a grammar of movement it is important to understand – i.e. to acquire – a lexicon of movements [1,2,37]. The dictionary creates the common space of information, the basic brick of any representation of the reality: “what am I doing now”. In recent literature, actions are acquired using diverse technologies, all equally good or bad dependently from the situation.

In this thesis wearable inertial sensors has been used to classify movements, the choice has been driven by technological and practical advantages, they are cheap, light, and - differently from video cameras - are not prone to the hidden face problems, or luminance problems, and so on [42,43,46,53,54]. The main idea is to use inertial sensors to understand what a person is doing for ambient-intelligent, healthcare, medical and sport applications.

A lot of literature has been written about the use of inertial sensors to classify movements, with different results; many of the proposed methodologies are very specific to problems or applications or to the given technology, sometimes are difficult to translate into practice, use minimal dictionaries (3-6 actions). Moreover, they do not give a general representation or explanation of the classification space [3,37].

In this thesis, my principal concern is to propose a method that is not centered on technology but on data, that could be a general framework and could also give a general representation of movement space, useful in other areas of research, e.g. in reasoning [1]. Inertial sensors are treated just as an example of sensors, a particular type of sensors not different from others, the method is new, reusable, simple, neat, and the accuracy results very interesting [2,4].

I used a traditional machine learning method where data coming from sensor have been represented like vectors, then I wheighted the features, introducing a transformation of this space, driven by two simple concepts: give the features a weight related to their ability to discriminate and generalize[1,2].

Thanks to this transformation, vectors representing an action are grouped in more dense and separated clusters, classification results are accurate even with “great dictionaries” (up to 21 actions).

The method is generic, freed of an a priori semantic in order not to fall into the problem of semantic questions (What is a movement? What is *this* movement? In what it differs from others?); and, finally, it is technological independent, and highly accurate. The thesis contains, I believe, a new and interesting general representation of the movement space, useful also for other area of research [1,4]. This is the general context of this thesis.

Summary

The goal of the thesis is to recognize activities, interpreting data generated by wearable inertial sensors.
To this aim it is possible to develop specific algorithms able to identify the most specific characteristic of a set of actions, executed by one or more people. The idea is to automatically identify movements and activities, avoiding the use of videocameras, or any direct intervention of a person for all the applications where movements recognition is required.
Movement recognition with inertial sensors proved to be useful for social surveillance applications [4], in neuroscience [12] and for tracking activities [3,37,40]. Inertial sensors can also be used for sport analysis, for gait and posture analysis, for human computer interaction and in motion recognition and capture [3,23,27,37,41,45,58]. To classify movements with inertial sensors could also be an important step to recognize human emotions from body movements and posture [16,19].
On the one hand, inertial sensors require a certain amount of user cooperation and could be considered invasive and cumbersome. On the other hand, since hardware is becoming smaller and smaller the user acceptability of body-worn sensors has improved and will continue to improve [4,58,60]. Moreover, inertial sensors have many advantages since they can be directly placed on specific body segments or in clothes. This has many implications: we know with certainty to which segment of the body the data collected by the sensors refers to, we do not have to solve "hidden parts" problems created by video cameras, nor solve color and luminance issues. Also, we do not have to interpret/understand the surrounding environment, for example separate the body information from the background information, and identify people [46,52,54].
In this work attention is focused on movement recognition with inertial sensors for movement classification, using a generic method, semantically flexible, and technological independent. The method was tested with two very different technologies and vocabularies of actions and in all cases it performed with very high accuracies. [1,2]. I will show that the given methodology is not only general, an technological flexible, but also improves intracluster and intercluster similarity reaching a very high accuracy, with a simple method of classification.

The Methodology

To create a test set, a given set of actions is executed by a set of people, then features are extracted from the generated data, action by action, sample by sample; the given features values are filed in a vector. The set of features-vectors of a action executed by a population constitutes the pattern of the given action. Some features are more frequent within the population, others can be less frequent inside the vocabulary's actions. In order to

take into account this aspects, two weights have been introduced: the FF (“Feature Frequency”) and the IVFF (“Inverse Vocabulary Feature Frequency”). Values are quantized, and FF and IVFF are calculated.
The FF takes into account how frequent a feature is in the given population class by rising the importance of the features that appear in the same class of movements. The IVFF takes into account how frequent a feature is in the dictionary. A feature that is present in more actions is considered less discriminative, and its weight is lowered.
We have to note that FFxIVFF space is different than the original values space: some dimensions can be canceled or enhanced depending on the role of features in the dictionary and population. The FFxIVFF transform the original space in a space where clusters appear to be more dense and more separated. This approach will allow to reach high accuracy (the highest in literature as far as I know), using very simple algorithms of similarity. The algorithmic costs of the similarity algorithms are low, the accuracies are interestingly high, giving to the methodology a promising appeal for real time applications.

The Datasets

In order to test the proposed method, I used two different database: NIDA 1.0 (Nomadis internal Database of Actions) the internal of the Nomadis Laboratory of the University of Milano-Biccoca used for this thesis and other tests, and the WARD 1.0 (Wearable Action Recognition Database) created at UC Berkeley by [A.Y. Yang et al. 2009]. These two databases are quite representative of a typical database of actions, are big in dimensions (270 and 1200 samples) and in number of actions (13 and 21), compared to most databases known in literature. Also, the two databases are quite different for sensors technology, dimension of the vocabulary of actions, number and type of samples, and the granularity of low level data.
A great number of classification algorithm has been tested in different conditions, using the Leave One Out Croos Validation (LOOCV) methodology, and results are very interesting.

Test and Results

The methodology has been applied - as is - without significant changes to both the databases, and accuracies are very high in both cases. Results are the highest in literature, as far as I can know. The accuracy of our methodology on the U.C. Berkely WARD 1.0 database outperforms the results of the U.C. Berkeley researchers, reducing by four times the error rate (from 6.6% to 1.57%) using their dataset.
Also, a lot of tests has been done to test the accuracy reducing the number of wearable sensors, to check the sensitivity to sensor numbers, and results still are very interesting. With the U.C. Berkeley database WARD 1.0 using 3 sensors on the pelvis, the right wrist and ankle I reached an accuracy of 97.63%, higher than the 93.4%

obtained at UC Berkely with all the 5 sensors using the same database. And I reached 93.62% of accuracy using only one sensor, against the 93.4 % of U.C. Berkeley obtained on the same database with all the 5 sensors.

Further Analysis

Both the WARD 1.0 and NIDA 1.0 multidimensional space have been reduced to a three dimensional space representation using Principal Component Analysis methodology. The analysis of three dimensional reduction confirm the peculiar "lexical" characteristic of this space (similar and opposite actions are very close) . Finally, the Intracluster and Intercluster measures confirm that the FFxIVF transformation creates a space where clusters are more dense and well separated, justifying the higher accuracies obtained in the tests sections with the LOOCV methodology.

Conclusions

Conceptual clearness, high accuracy, technology independency, low sensitivity to the number of sensor variation, and the use of fast algorithms, gives this new methodology a great appeal. Also, the FFxIVFF generating a space where clusters are more dense and separated, allow the use of very simple algorithms. The method opens an interesting path to activity recognition with inertial sensors, which is general, flexible, technologically independent, accurate, and creates a bridge between the instance based techniques, the applications and the semantic domains.

State of the art

Many different approaches were used in the movement recognition area with inertial sensors, we can roughly divide them in two different types: the approaches where researchers use a specific set of features that have heuristically been proven to be suitable for characterizing a chosen set of movements [4, 37, 39, 49, 53] andthe approaches where machine learning techniques are used to recognize a movement [5, 23, 26, 37, 45, 52]. An interesting survey can be found in [3,37]. There are also approaches that interpret a movement as a sequence of hidden states utilizing Hidden Markov Models (HMM) to predict movement from observables [26, 39]. HMM works well in some situations, but we have to note that for HMM to work at their best they need a great quantity of information, every single change in the chain of observables needs a new statistically significant quantity of samples. Also you have to train a different HMM for every single action: the introduction of a new action forces a long process of retraining; changing the whole vocabulary of actions requires a great quantity of work. Finally, simple actions that are not composed by a chain of observables, are not well recognized [23,26].

Many different technologies and sensors have been used, both in quality and quantity: some prefer to use different mono dimensional sensors [37], others a single inertial unit mounted at a specific place on the body [4,37]. Others researchers prefer multimodal approaches, conjunctly using audio and inertial sensors [23,26]. Recently, a new technique was proposed by A.Y. Yang et al. of the U.C. of Berkeley called Distributed Sparsity Classifier [5]. For this research a public database of movement has been made available, the WARD 1.0 database. I used the U.C. Berkeley WARD 1.0 database to test the accuracy of my proposed methodology and to compare my results with the interesting results of the U.C.Berkeley researchers.

If a general defect could be found in these approaches is that they often are specific to the chosen technology and problem (the dictionary of actions). The lack of a flexible and general approach, that could be useful in different situations or with different technologies, and different vocabularies, is – in my opinion – the open issue of the area. Given the great differences of technologies and dictionaries, at the moment the methodolgies are quite opportunistics, and this is an impediment to reach “definitive” and general progresses in this field [3,37].

Also, even if great accuracies are sometimes described in literature, the results are related to tests done with limited vocabularies (three to five actions), and above all public database are never used (as far as I know): there is no a direct confrontation and that sometimes makes it difficult to compare the proposed methods.

Finally, some of the proposed methodologies focus on the ability to discriminate, not on the importance of the features, hence they use very complex machine learning technology, with high computational costs [5,23,26,45,52]. And this is an impediment to use them in real time application.

In the next sections, I give a synthetic overview pinpointing the differences of approaches in the activity recognition area using inertial sensors: far to be exhaustive it shows some interesting and representative results in the field from a methodological point of view (cfr. also [5,37]). None of these research use a public database to tests results, and unfortunately this is quite common.

Many different methodologies are used, sometimes they are applied to not a great number of subjects (usually they does not exceed the number of ten subjects), use different technologies with different approaches, and above all often use a limited vocabulary: for example some research tries to recognize just threeactions. It comes easily that accuracies must be "normalized" by the number of actions we want to recognize. An accuracy of 50% in a vocabulary of two actions is not significative at all: a random choice has an accuracy of 50%.

In every listed reference has been indicated the Recognized activities, Accuracy, used algorithms or Methodology, and Numerosity of the population.

Reference: [41]

Activities: Walking, Ascending stairs, Descending stairs

Accuracy: 92,85 % - 95,91%

Method: fuzzy sets of Gaussian distributions based on feature for each activity; samples classified as activity with the highest likelihood given activity distribution

Number of subjects: 8 (6 male , 2 female)

Reference: [45]

Activities: Standing, Walking, Climbing stairs

Accuracy: 83% - 90%

Method: 3 Multiplayer Percepton Neural Network trained for 3 activities using back propagation. 10-fold cross validation used to test classifiers.

Number of subjects: 6

Reference: [52]

Activities: Walking, Ascending stairs, Descending stairs, Sitting, Standing, Lying supine, Sitting-Talking, Bicycling, Typing

Accuracy: 95,8% (subjects perform a scripted sequence of activities), 66,7% (naturalistic data).

Method: weighted sum of absolute differences between feature values and activity reference patterns; sample classified as nearest activity reference using distance metric.

Number of subjects: 24

	Lying	Sitting	Sitting-Talking	Typing	Standing	Walking	Descending stairs	Ascending Stairs	Cycling
Lying	16	0	0	0	0	0	0	0	0
Sitting	2	6	2	0	2	0	0	0	0
Sitting-Talking	0	3	2	0	1	0	0	0	0
Typing	0	16	3	2	1	0	0	0	0
Standing	0	0	0	0	114	3	1	0	0
Walking	0	0	0	0	6	107	4	4	0
Descending Stairs	0	0	0	0	2	80	26	1	0
Ascending Stairs	0	0	0	0	3	20	1	25	0
Cycling	0	0	0	0	0	0	0	0	13

Table 1 Confusion Matrix relative to reference 52

The Confusion Matrix in Table 1 shows that it is difficult to discriminate similar actions like “walking”, “walking downstairs”; similarly, it is difficult to discriminate “sitting”, “sitting-talking” e “sitting-typing”.

Reference: [51]
Activities: Lying, Sitting, Standing, Locomotion
Accuracy: 89,30%
Method: Decision Tree using feature set
Number fo subject: 5

Reference: [39]
Activities: 3 types of karate martial arts movemetns: cuts, elbows and punch blocks
Accuracy: 96,77%
Method: 9 Hidden Markov Models to identify 9 sub-gestures; 3 Markov Model on sequence of sub-gestures used to for each of the 3 major gestures.
Number of subject: 1, a Martial Art Instructor

Reference: [44]
Activities: Walking, Ascending stairs, Descending stairs
Accuracy: 83% – 96%

Method: Nearest neighbor classification using Euclidean Distance between feature vector and personalized reference feature vector for each avtivity; referenced vectors calculated as average of trainig samples feature vectors.
Number of subjects: 6

Reference: [50]
Activities: Walking, Running, Ascending stairs, Descending stairs, Sitting, Standing, Bicycling
Accuracy: 42% - 96%. In particolare: walking (75%), running (78%), ascending stairs (42%), descending stairs (64%).
Method: bidimensional Kohonen Map with 20 x 20 neurons, trained on the feature set.
Number of subjects: 1, researcher.

Reference: [49]
Activities: Walking, Running, Ascending stairs, Descending stairs, Sitting, Standing.
Accuracy: 85% – 90%
Method: Single layer neural network trained using 4 features
Number of subjects: 10

Reference: [37]
Activities: Walking, Walking carrring, Sitting and relaxing, Working on computer, Standing still, Eating or drinking, Watching TV, Reading, Running, Bicycling, Stretching, Strenght training, Folding laundry, Laying down, Brushing teeths, Climbing stairs, Riding elevator, Riding escalator.
Method and Accuracy: “Decision table”, accuracy 49.95%. IBL (“Instance based learning” or “Nearest Neighbor”), accuracy 70,33%. “Decision Tree C4.5”, accuracy 73.69%. Naive Bayes, accuracy 32.22%.
Number of subjects: 20

Reference: [29]
Activities: Standing, Walking, Running, Ascending stairs, Descending stairs, Sit-ups, Vacuuming, Brushing teeth.
Method and Accuracy: (settings 4: leave one out) “Decision table”, accuracy 46,33%. “Decision Tree”, accuracy 57%. “Plurality voting”, accuracy 65%. “Boosted Support Vector Machine”, accuracy 73,3%. “Naive Bayes”, accuracy 64%
Number of subjects: 2

Reference: [26]

Activities: Sitting, Standing, Walking, Jogging, Walking upstairs, Walking downstairs, Driving a car, Riding a bicycle, Riding elevator down, Riding elevator up.

Accuracy: Overall accuracy 95%

Method: Decision stumps and Hidden markov models

Number of samples: about an hour of data per activity and around 100 instances per activity.

	Sitting	**Stand**	**Walking**	**Jogging**	**Walking upstairs**	**Walking downstairs**	**Riding Bicycle**	**Driving car**	**Riding elevator down**	**Riding Elevator up**
Sitting	89.8%	38.5%	0.5%				0.4%	33.4%		
Standing	10.1%	50.8%	1.4%							
Walking	0.1%	7.4%	97.7%		5.2%	2.5%				
Jogging				100%						
Walking upstairs					94.8%					
Walking downstairs			0.5%			97.5%				
Riding bicycle		3.3%					99.6%			
Driving car								66.6%		
Riding elevator down									100%	
Riding elevator up										100%

Table 2 Hmm classifier accuracy

Reference: [5]

Activities: Stand, Sit, Lie down, Walk forward, Walk left-circle , Walk right-circle, Turn left, Turn right, Go upstairs, Go downstairs, Jog, Jump, Push wheelchair.

Accuracy: 93.5%

Method: Distributed Sparsity classifier, an important operation of feature projection is needed to work

Number of subjects: 20 (7 female, 13 male)

	Stand	**Sit**	**Lie Down**	**Walk forward**	**Walk left-circle**	**Walk right-circle**	**Turn left**	**Turn right**	**Up stairs**	**Down stairs**	**Jog**	**Jump**	**Push wheel chair**
Stand	87.2	10.2	0.7	0	0	0	0.1	1.8	0	0	0	0	0
Sit	25.2	66.8	6.8	0	0	0	0.1	0.1	0	0.1	0	0.1	0.7
Lie down	2.6	5.1	91.8	0	0	0	0	0	0	0	0	0.1	0.3
Walk forward	0	0	0	92	2.5	1.6	0.2	0.2	0.4	0.7	0	0.2	0.3
Walk left-circle	0.1	0	0	0.2	97.3	0	0.6	0.3	0.3	0.1	0.1	0.2	1
Walk right-circle	0	0	0	0.1	0.1	95.7	0.2	0.4	0.4	0.4	0.5	0.2	2
Turn left	0	0	0	0	0.6	0	97	2.3	0	0	0	0	0.1
Turn right	0	0	0	0	0	1.6	3.1	95.2	0	0	0	0	0
Upstairs	0	0	0	0	0	0	0	0	98	0.1	1.6	0.1	0.2
Downstairs	0	0	0	0.2	0.1	0	0	0	0.1	98.3	0	0.5	0.8
Jog	0	0	0	0	0	0	0	0	0.5	0	99.3	0.1	0.1
Jump	0.1	0	0	0	0	0	0	0	0.3	0.6	0.5	97.9	0.5
Push wheelchair	0.3	0.1	0	0.1	0.2	0.1	0.1	0.1	0	0.2	0.2	0.1	98.6

Table 3 Confusion Matrix for Distribute Sparsity Classifier. Values are in percent.

Single Action Analysis

The main idea of this thesis is to create a recognition mechanism the could be as generic as possible. The goal is of being able to recognize any type of action with a high degree of accuracy regardless of the type of action executed (an athletic gesture, a daily life acitivity, etc.) and this has to be done in different contests regardeless of the used thechnology. Before starting to analyze a set of different type of actions we have to identify a process to transform data related to a single sample to more useful actions. The idea is at first to filter data to eliminate noise, and to identify other aspects of the movement like the change of velocity, change of accelerations; than, to extract features from the filtered information.
Hence, for every single action data has been filtered and features have been extracted.

Filters

Filters have a role in the feature extraction process, to clean the signal from micro-changes that bring in the signal background noise that affects the information. In addition, filters are useful to enhance some of features that are not evident in the raw signal. The used filters are:

- Smoothing
- Low Pass Filter
- Variance
- Newton’s Difference Quotient (First Derivative)
- Newton’s Difference Quotient (SecondDerivative)

Appendix B contains a detailed description of the filters used in the feature extraction process.

Features

To catch specific aspects of movement very generic features has been used. The idea is that the reweighting process will help to understand which of these generic features is more helpful, avoding the burden to analyze every action for the most promising one. The features used in the feature extraction process are:

- Maximum value
- Minimum value
- Starting value
- End value
- Mean
- Variance
- Zero Crossing Rate
- Mean Crossing Rate
- Skewness
- Kurtosis

In **Appendix C** a detailed description of the filters used in the feature extraction process.

Body Sensor Profiling

In human movement analysis, the position and number of sensors is of great importance. In line of principle, we can think that we should position an inertial unit on every single segment of the body to capture a very significant useful amount of information for actvity recognition. For example, with enough sensors we could capture the movement of the forearm in respect to the arm, or the movement of the head and neck in respect of the torso, or even the fine movements of every single finger.

In sport activities, the sensors' combination that will give more information is the one where sensors are attached to the body segment that carries out these actions. For example, if you want to recognize karate moves like a punch, a front kick or side kick we could ignore the limbs on the left side of the body (if the person is not left-handed of course), and focus on the right side wrist, shoulder, hip, knee and ankle. If you want to recognize generic movents of daily life, it would be more useful to place the sensors on all the arms, using both the left and right side of the body.

Nontheless, there is no a clear evidence in literature – as far as I know - if we really need all these sensors, and of how many sensors are really necessary to accurately classify and identify a movement (e.g., an activity of daily living, or a sport activit)y. As we will see there is no a necessity to place sensors on every segment of body, for an accurate recognition. In general we will see that with the proposed methodology, three sensors work as well as five sensors, even if thery are placed on the left side and the action is right-handed. Still with a single sensor we have very interesting results that improve on the results presented in literature (see Test Results p.59).

The choice to use five sensors, one for every limb and one for the hip according to a X scheme (right wrist, left wrist, pelvis, right ankle, ankle left) as shown in Fig. 1, was driven by the need to create a profile useful in many different situations, and test it. Tests will also verify the accuracy of the recognition varying the number of sensors.

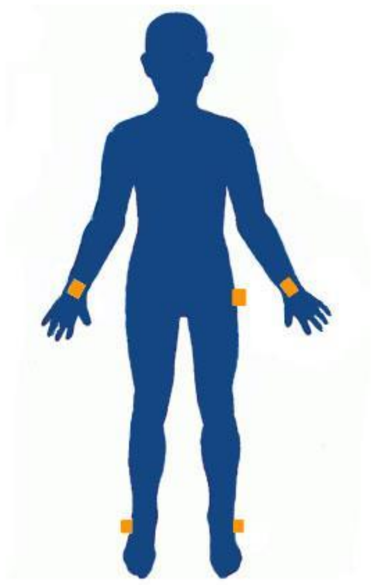


Fig. 1 – Position of sensors on the body.

Body Profile and Human Profile

Another aspect to consider is the “Body Profile”. This term refers to the physical characteristics of the person. The idea is that we have a uniform population in term of body characteristics, so the profiling should be “modeled” on the basis of age, weight and height.

Once clarified the scope of use of sensors, it is important to divide the subjects considering the motor skills (“Human Profile”). A young healthy person with no physical problems, and an elderly person with difficulties to move, execute the same action in a very different way.

Our research tests have been applied to people of age between 19 years and 79 years old with normal motor skills, but users have different heights and weights. In other words, our group of people is formed by people quite etherogeneous for body profile, but contains only healthy people, with normal motor skills.

Vocabulary of Actions – The Methodology

The goal is to propose a method flexible, technology independent, and as general as possible. The main idea is to analyze *features* using a method able to enhance the most important ones, and diminuish or invalidate the less important, using an approach that is not specific of a single domain, and that could be reused in multiple contexts. To this aim I will operate on the features using a method that could be reusable in many different context and as general as possible.

In an domain specific approach you want to identify the type of action to be recognized in its specific context for a specific purpose. In sports activities like karate, there may be actions that differ just in the way you can hit your opponent, for example by kicking and/or using punches. On the other side, the actions we want to recognize in neurology and the rehabilitation fields are completely different and can vary depending on the type of disease of which the person is suffering (neurology, post-traumatic, post-surgery) that we want to recognize or measure. Daily activities are obviously the more generic and probably the more studied type of actions with inertial sensors, and have completely different characterizing features, and a different application context.

Many of the current approaches are based either on a long work of feature analysis [4, 37, 39, 49, 53] in search for the best feature for each specific domain; others use sophisticated classification methodologies, or methodolgies that are domain specific, or are specific for the particular movement that is being acquired and recognized [4, 23, 26, 41, 45, 52].

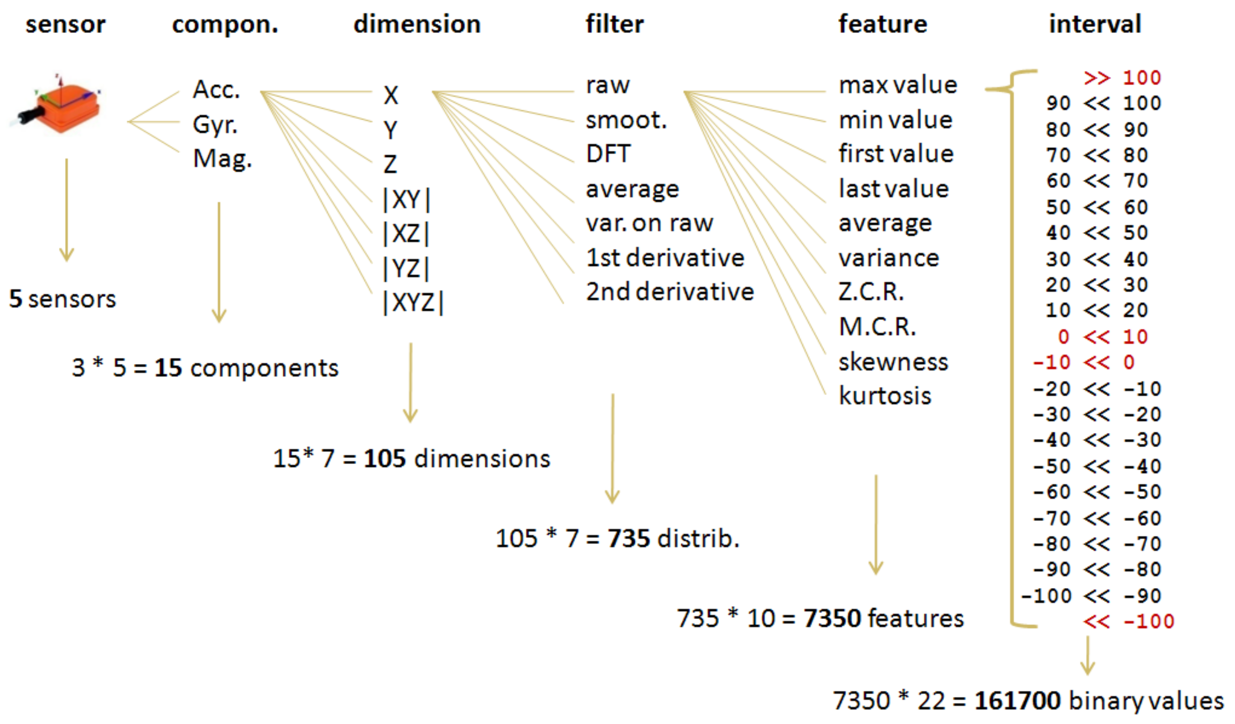
The method here proposed is domain indifferent, does not use biomechanical models of the body, nor require an apriori knowledge of the movement-domain science aspects, focusing on the analysis of the value of features within a set of actions that we call vocabulary or dictionary of actions.

The first step of the proposed method requires the construction of a vocabulary of actions. A series of actions is executed by a group of individuals belonging to a population that has similar *Human Profile* (motor ability) and a different *Body Profile* (different ages, wheights, and heights). Sensors are positioned according to the more appropriate sensory profile for our goals, that is the more generic profile possible. Specifically, sensors have been placed according to a X scheme (right wrist, left wrist, pelvis, right ankle, ankle left) as shown in Fig. 1. I proceeded to acquire the raw data an store them into the file system. These data constitute the vocabulary of data, the NIDA 1.0 database, which subsequently will be used for the recognition phase. In a second phase the public database WARD 1.0 was used for confrontation with results present in literature.

Feature Extraction

Roughly speaking, there are “two ways” to select features: one can select - “a priori” - the most promising feature for a specific set of actions, that means that you have to study the problem and identify the most useful feature for the given domain; otherwise you can use some very generic features and exploit criteria to reweight their role in given application domain (in the given vocabulary of actions).
The proposed approach is to use very generic features, usually considered helpful in literature. The major advantage is to keep the mechanism as general as possible, and to eliminate the problem to consider what features are more suitable each time you change the application domain (the vocabulary of actions). On the other hand, it is necessary to generate a large number of features, in order to be “statistically certain” that among the many values it could be possible to find some very distinctive features useful for the analysis and recognition process. The more are the features, the more difficult it is to efficiently classify activities. In particular if we want to create a methodology useful also at runtime. Hence, there is a tradeoff between generality of the methodology, technology indipendence, and accuracy and efficiency of recognition. As we will see, the proposed methodology can deal well both with the two competing parts of the problem: **generality** and **accuracy**.

The *feature extraction* proposed here is a process that iteratively applies at every level all the filters and features described in the section Single Action Analysis. An action in order to be recognized can by acquired by the database or directy executed. From every inertial sensorraw data are extracted. Then their 2D and 3D magnitude is calculated. The obtained data are transformed using the filters presented before obtaining a great number of distributions. Finally *features* are extracted (Fig. 2).
The result is a series of real values (7350 in the example of Fig. 2) that represent the *feature* values of the executed action. Then values are quantized in order to make easier the re-weighting process. In Fig. 2 the schema of *features extraction* process used with *MTx*sensors .



.

Fig. 2 - The iterative process of feature extraction

Feature Quantization

Feature values of every single action constitute a pattern and we want to compare it with the classes pattern. Hence, these values should be used during the analysis and also during the recognition process. To use these data, values have been quantized, to make the comparison between the values much easier (computational quick) as they can be compared using intervals .

The intervals in which to divide the features can not be chosen at random, for example they cannot be too small, to avoid the risk that any pattern could be self specific, and this will give not a reliable significance the data. On the other side, intervals cannot be too large, because otherwise it will be highly likely that too many feature values will be quantized within the same interval.

The interval dimensions may not be equal for all the observed distributions, because every component's sensor has its own scale of values which is often very different from the others. But roughly speaking it has been decided to divide the domains into 20+2 intervals. For example, for the *Skewness* this is the interval quantization:

```
>> 5
4.5 << 5
4 << 4.5
3.5 << 4
3 << 3.5
2.5 << 3
2 << 2.5
1.5 << 2
1 << 1.5
0.5 << 1
0 << 0.5
-0.5 << 0
-1 << -0.5
-1.5 << -1
-2 << -1.5
-2.5 << -2
-3 << -2.5
-3.5 << -3
-4 << -3.5
-4.5 << -4
-5 << -4.5
<< -5
```

When the value exceeds the maximum (or minimum) it is simply represented with either the values >> 5, << -5.

Features Extraction - Data Dimensions

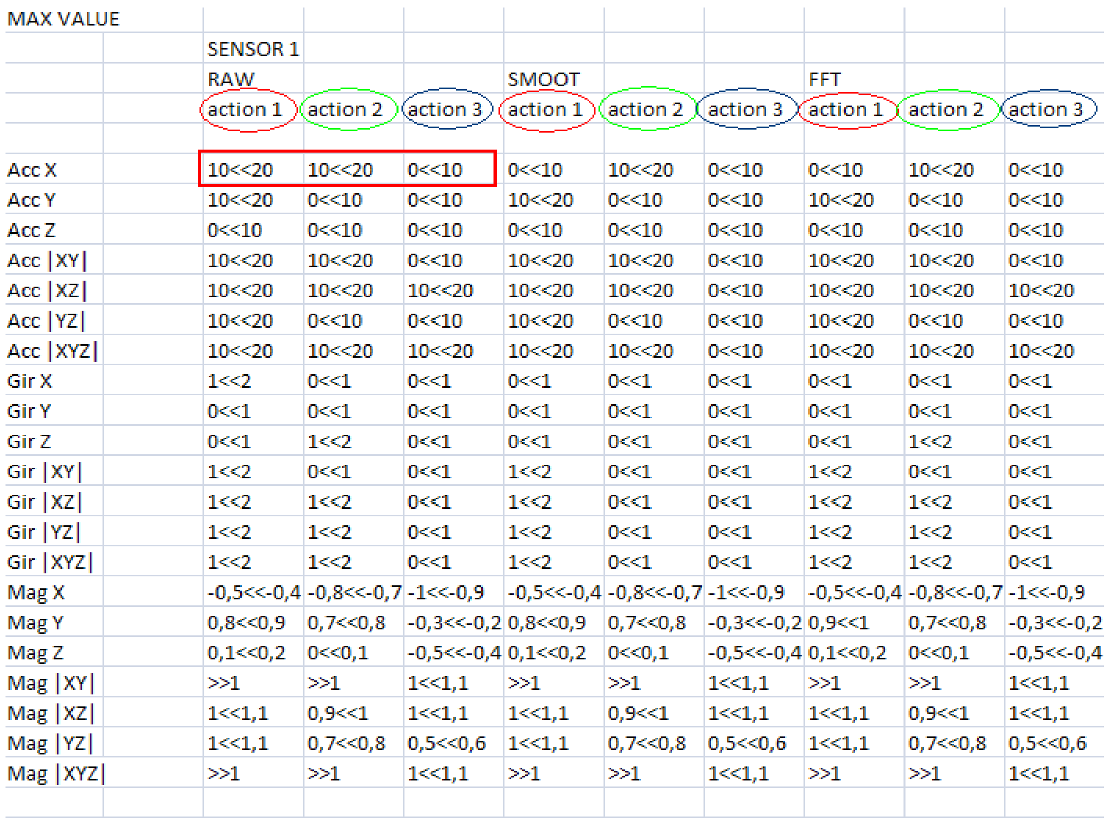
The number of *features* exctracted from every action depend on some factors. The number of sensors S_i on the body, the number of devices D_j in the sensor, the number of C_k components, the number of F_y filters used for the analysis, the number of σ features choose for the analysis. The N total number of generated features is:

$$N = S_i * D_j * C_k * F_y * \sigma$$

For example, the MTx sensors used in the dataset NIDA 1.0 are technologicaly different from the sensors of dataset WARD 1.0, that have only a 3D accelerometer and a 2D gyroscope, the total number of extracted features for a single actions in the two dataset is different.
The number of intervals is not easily calculated, since for each feature there are a number of sub-varying intervals. As said before, in average the intervals are 22.

Weighting Features: Population and Dictionary

The recognition phase is quite simple if the vocabulary of actions consists of a very small set of actions performed by one person. When the number of actions (classes) grows and the number of individuals grows, good results are not easily obtained. Therefore, we would like a feasible way and an efficient mechanism that should be able to tell us which *features* are more suitable to distinguish an action by another (discriminative power of the feature), and which *features* are characteristics of a specific action for the given population, and are not specific of an individual (generalization power of the feature).
The solution proposed to this problem is to weigh the *features*, using an approach very similar to what is done in Information Retrieval and in Text Mining [9].
In the first phase I will extract the *feature* values of every action and every person of the given population.



MAX VALUE									
	SENSOR 1								
	RAW			SMOOT			FFT		
	action 1	action 2	action 3	action 1	action 2	action 3	action 1	action 2	action 3
Acc X	10<<20	10<<20	0<<10	0<<10	10<<20	0<<10	0<<10	10<<20	0<<10
Acc Y	10<<20	0<<10	0<<10	10<<20	0<<10	0<<10	10<<20	0<<10	0<<10
Acc Z	0<<10	0<<10	0<<10	0<<10	0<<10	0<<10	0<<10	0<<10	0<<10
Acc |XY|	10<<20	10<<20	0<<10	10<<20	10<<20	0<<10	10<<20	10<<20	0<<10
Acc |XZ|	10<<20	10<<20	10<<20	10<<20	10<<20	0<<10	10<<20	10<<20	10<<20
Acc |YZ|	10<<20	0<<10	0<<10	10<<20	0<<10	0<<10	10<<20	0<<10	0<<10
Acc |XYZ|	10<<20	10<<20	10<<20	10<<20	10<<20	0<<10	10<<20	10<<20	10<<20
Gir X	1<<2	0<<1	0<<1	0<<1	0<<1	0<<1	0<<1	0<<1	0<<1
Gir Y	0<<1	0<<1	0<<1	0<<1	0<<1	0<<1	0<<1	0<<1	0<<1
Gir Z	0<<1	1<<2	0<<1	0<<1	0<<1	0<<1	0<<1	1<<2	0<<1
Gir |XY|	1<<2	0<<1	0<<1	1<<2	0<<1	0<<1	1<<2	0<<1	0<<1
Gir |XZ|	1<<2	1<<2	0<<1	1<<2	0<<1	0<<1	1<<2	1<<2	0<<1
Gir |YZ|	1<<2	1<<2	0<<1	1<<2	0<<1	0<<1	1<<2	1<<2	0<<1
Gir |XYZ|	1<<2	1<<2	0<<1	1<<2	0<<1	0<<1	1<<2	1<<2	0<<1
Mag X	-0,5<<-0,4	-0,8<<-0,7	-1<<-0,9	-0,5<<-0,4	-0,8<<-0,7	-1<<-0,9	-0,5<<-0,4	-0,8<<-0,7	-1<<-0,9
Mag Y	0,8<<0,9	0,7<<0,8	-0,3<<-0,2	0,8<<0,9	0,7<<0,8	-0,3<<-0,2	0,9<<1	0,7<<0,8	-0,3<<-0,2
Mag Z	0,1<<0,2	0<<0,1	-0,5<<-0,4	0,1<<0,2	0<<0,1	-0,5<<-0,4	0,1<<0,2	0<<0,1	-0,5<<-0,4
Mag |XY|	>>1	>>1	1<<1,1	>>1	>>1	1<<1,1	>>1	>>1	1<<1,1
Mag |XZ|	1<<1,1	0,9<<1	1<<1,1	1<<1,1	0,9<<1	1<<1,1	1<<1,1	0,9<<1	1<<1,1
Mag |YZ|	1<<1,1	0,7<<0,8	0,5<<0,6	1<<1,1	0,7<<0,8	0,5<<0,6	1<<1,1	0,7<<0,8	0,5<<0,6
Mag |XYZ|	>>1	>>1	1<<1,1	>>1	>>1	1<<1,1	>>1	>>1	1<<1,1

Fig. 3 – Feature-Action representation. Actions have different feature values

"All *features* have been created equal", but effectively they are *not* equal. They do not play the same role in different actions, and that dipends on two different competitive contexts: the "population" (intraclass distribution) and the "dictionary" (interclass distribution). Given an Action (class), some *features* are more frequent, others are more frequent in the Dictionary of actions, not all the *features* have the same *discriminative or generalization power*. A value of a *feature* that appears only in one specific action (class) in the vocabulary is certainly very important in terms of its discriminative capability, but if it is infrequent in the population, it has not a *"generalizing ability*". Vice versa, if a feature value appears quite often in the given Action (class), it may have a strong *"power of generalization"*, but if it is present in every action of the vocabulary, it is completely useless to *separate* one action (class) from another. These two factors have been taken into account to reweight features using a simple methodology inspired by Text Mining techniques

	walk	run	jump	fall	open door	punch	front kick	lateral kick	...
feat. 1	σ1	σ1	σ1	σ1	σ1	σ1	σ1	σ1	...
feat. 2	σ1	σ1	σ1	σ1	σ6	σ2	σ2	σ2	...
feat. 3	σ1	σ1	σ1	σ2	σ3	σ3	σ3	σ3	...
feat. 4	σ5	σ4	σ4	σ4	σ1	σ1	σ1	σ1	...
feat. 5	σ3	σ5	σ1	σ2	σ2	σ2	σ1	σ7	...
feat. 6	σ2	σ2	σ2	σ2	σ2	σ2	σ2	σ2	...
feat. 7	σ1	σ2	σ3	σ2	σ2	σ1	σ1	σ1	...
...	...	...	...	...	...	...	...	...	

Fig. 4 – Vocabulary relevance of features: σ6 value is specific of the "open the door" action, its discriminative power is very high. The $\sigma 1$ value of the feature 1 is present in every action of the Dictionary, its discriminative power is null.

In order to capture population relevance and vocabulary relevance of features, two wheights have been introduced. The first is called *Feature Frequency* (*FF*) and the second *Inverse Vocabulary Feature Frequency* (*IVFF*), these weights are inspired by Text Mining methodology [9]. *Feature Frequency* indicates how frequent is the value of a *feature* in the population class by class, and it is calculated using the following formula:

$$Ff_{i,j} = \frac{n_{i,j}}{|P|} \qquad (eq.1.1)$$

where $n_{i,j}$ is the number of occurrences of the σ_i feature in the action a_j , and $|P|$ is the population cardinality. The values of the *FF* go from a value close to zero, to a maximum of 1.

Inverse Vocabulary Feature Frequency indicates the general importance of the feature in the dictionary, if the the feature has the ability to discriminate an action (class) in the vocabulary. The formula is as follows:

$$IVFf_i = \log \frac{|A|}{|\{a : \sigma_i \in a\}|} \qquad (eq.1.2)$$

where $|A|$ represents the cardinality of the vocabulary, and $|\{a : \sigma_i \in a\}|$ the number of actions where feature σ_i assumes values.

The values of the *IVFF* goes from zero (if the *feature* in appears in all the Actions) to a maximum of $\log |A|$ which indicates that the features potentially have a great power to discriminate the actions in which they appear.

The overall weight of a single feature is *given* by the product:

$$W_{i,j} = Ff_{i,j} * IVFf_i \qquad (eq.1.3)$$

Hence, our actions are represented as *N*-dimensional *feature* vectors in an *N*-dimensional vector space.

In Fig. 5 an example of the space of *features*. The F1, F2 axes represent the *features-intervals* (in this case, we have only two features for simplicity), and actions are vectors of this space (three actions A1 A2 A3 are represented). The coordinates of the A1, A2, A3, vectors are given by the *FFxIVFF* weights of the action on the *feature-intervals* F1, F2.

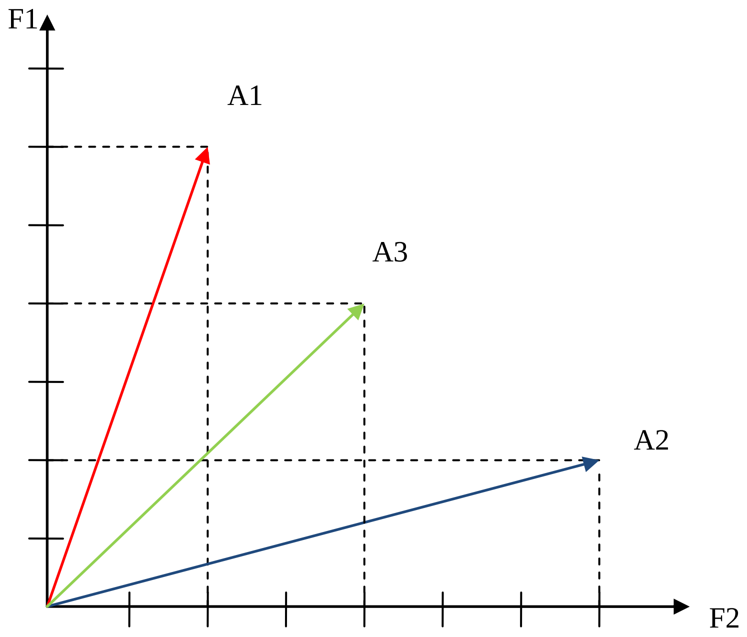


Fig. 5 – Conceptual representation of the actions in the feature space

It is important to say that the *FF xIVFF* formula transform the original values vector space into a different space. The *FF* takes into account how frequently a feature-interval is hit action by rising the importance of the values that appear in the same class of movements (Eq.1.1). The *IVFF* takes into account how frequent is a feature-interval in the dictionary. A feature value that is present in more actions is considered less discriminative, and its weight is lowered according to the formula (Eq. 1.2.). Note that distances in the interval-frequency space (the *FF x IVFF* space) could be very different than in the feature value space: in the *FFxIVFF* space dimensions can be canceled or enhanced depending on the role and frequency of features in the dictionary and in the actions (classes).

The expected behavior that similar actions will be close in the *feature* space. Similar actions will be "parallel", i.e. will share the same subspace. Mutually orthogonal actions will be dissimilar: the orthogonality between two vectors is maximum when two actions have no dimensions in common.

In *FFxIVFF* space what is taken into account are not the feature values, but the frequency a feature-interval is hit. That means that in the *FFxIVFF* space, actions are close when have similar *FFxIVFF* values, i.e. if they have same frequency, on the same intervals.

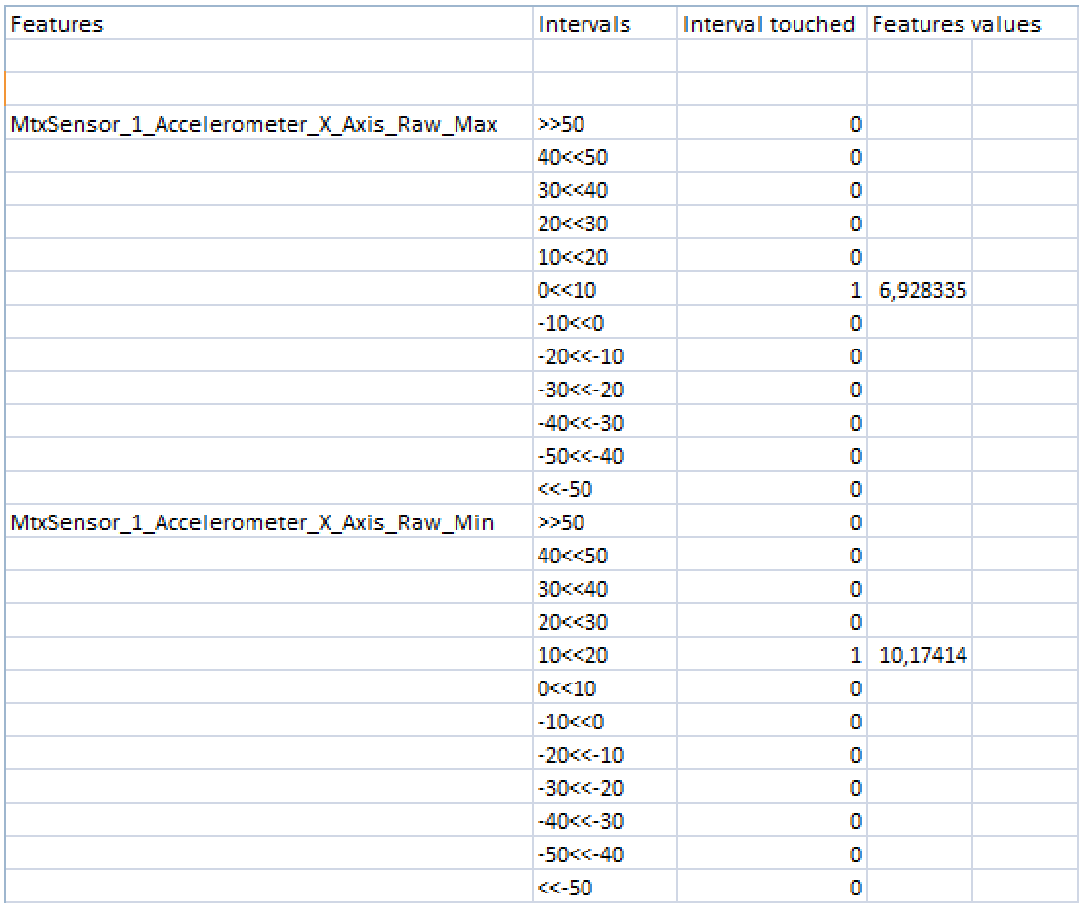
The expected effect of the *FFxIVFF* operation is to modify the dimensions and separability of the cluster of data representing an Action (class) (see Intracluster and Intercluster Similarity, p.136). As we will see, this will grately improve the accuracies (see Test Results, p.59, p.100).

In this space where cluster are more separated and more dense, it is possible to use simple classifiers like distance and similarity classifiers to recognize actions with great accuracy (see Conclusions, p.132). Simpler classifiers have low computational costs (see Computational Costs, p.42) .

Also, *FFxIVFF* values are context dependent, they depend from Dictionary and Popoulation. But given the database their values can be automatically calculated. This assures that the methodology is usable in different domains, and with vocabularies of different dimensions.

Recognition Process

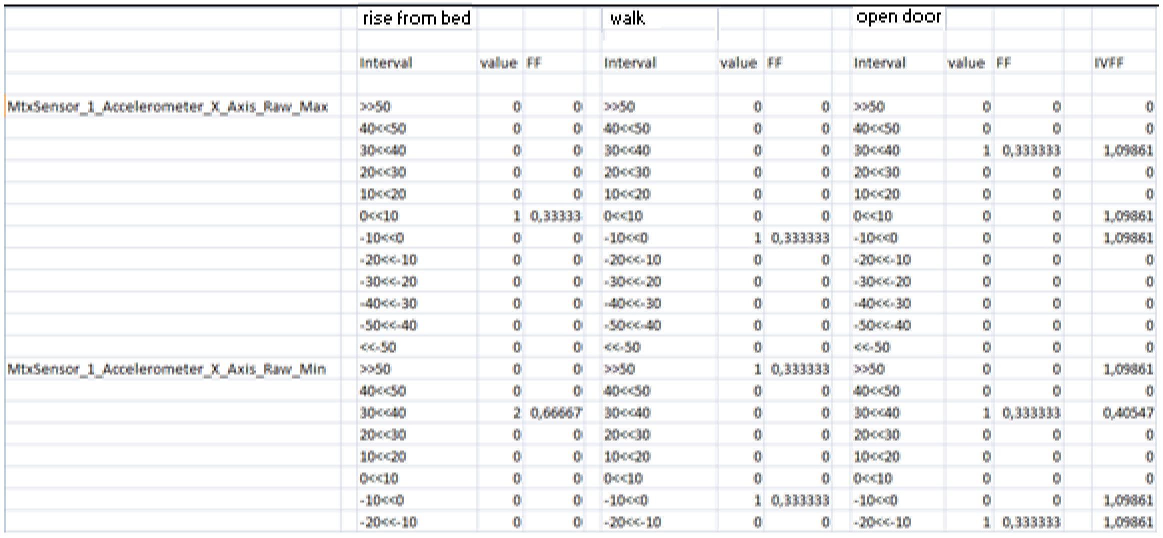
The samples are retrieved from the database class by class, features are extracted with the previously explained method (see Features Extraction, p.28) and stored in vectors.



Features	Intervals	Interval touched	Features values	
MtxSensor_1_Accelerometer_X_Axis_Raw_Max	>>50	0		
	40<<50	0		
	30<<40	0		
	20<<30	0		
	10<<20	0		
	0<<10	1	6,928335	
	-10<<0	0		
	-20<<-10	0		
	-30<<-20	0		
	-40<<-30	0		
	-50<<-40	0		
	<<-50	0		
MtxSensor_1_Accelerometer_X_Axis_Raw_Min	>>50	0		
	40<<50	0		
	30<<40	0		
	20<<30	0		
	10<<20	1	10,17414	
	0<<10	0		
	-10<<0	0		
	-20<<-10	0		
	-30<<-20	0		
	-40<<-30	0		
	-50<<-40	0		
	<<-50	0		

Fig. 6 – Here a part of the array of features representing an action

Given the *feature vectors* of every person and every action, it is possible to build the *Training Set table* that contains the collection of information created during the operation of *feature extraction*.



	rise from bed			walk			open door			
	Interval	value	FF	Interval	value	FF	Interval	value	FF	IVFF
MtxSensor_1_Accelerometer_X_Axis_Raw_Max	>>50	0	0	>>50	0	0	>>50	0	0	0
	40<<50	0	0	40<<50	0	0	40<<50	0	0	0
	30<<40	0	0	30<<40	0	0	30<<40	1	0,333333	1,09861
	20<<30	0	0	20<<30	0	0	20<<30	0	0	0
	10<<20	0	0	10<<20	0	0	10<<20	0	0	0
	0<<10	1	0,33333	0<<10	0	0	0<<10	0	0	1,09861
	-10<<0	0	0	-10<<0	1	0,333333	-10<<0	0	0	1,09861
	-20<<-10	0	0	-20<<-10	0	0	-20<<-10	0	0	0
	-30<<-20	0	0	-30<<-20	0	0	-30<<-20	0	0	0
	-40<<-30	0	0	-40<<-30	0	0	-40<<-30	0	0	0
	-50<<-40	0	0	-50<<-40	0	0	-50<<-40	0	0	0
	<<-50	0	0	<<-50	0	0	<<-50	0	0	0
MtxSensor_1_Accelerometer_X_Axis_Raw_Min	>>50	0	0	>>50	1	0,333333	>>50	0	0	1,09861
	40<<50	0	0	40<<50	0	0	40<<50	0	0	0
	30<<40	2	0,66667	30<<40	0	0	30<<40	1	0,333333	0,40547
	20<<30	0	0	20<<30	0	0	20<<30	0	0	0
	10<<20	0	0	10<<20	0	0	10<<20	0	0	0
	0<<10	0	0	0<<10	0	0	0<<10	0	0	0
	-10<<0	0	0	-10<<0	1	0,333333	-10<<0	0	0	1,09861
	-20<<-10	0	0	-20<<-10	0	0	-20<<-10	1	0,333333	1,09861

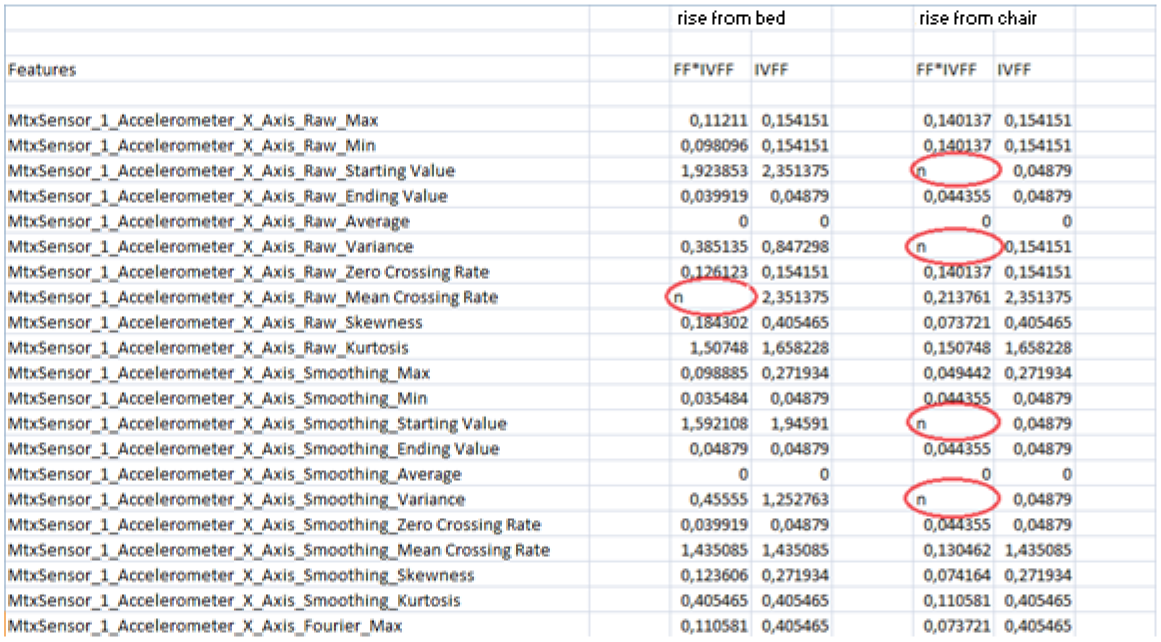
Fig. 7 - Portion of the Training Set table

The results are stored in a table with a number of rows equals to the number of extracted features (in our case 161,700) and a number of columns equal to the number of Actions (classes) (see Fig. 7) . In this table, for each pair of information ("FeatureInterval" , "Action") are represented both the *FF* value and an $n_{i,j}$ value, (see Eq. 1.1). In addition, for each interval the *IVFF value* of all the actions is calculated and stored on the last column.

By analogy with Text Mining terminology, the action to be recognized is called "Query". An action to be recognized is executed at runtime, or retrived from the database. The query action is an n-dimensional vector member of the same space of the *Training Set* (see Fig. 9).

Since we assume that similar actions will be metrically close, to recognize what action is the most similar to the given action (the query), we should measure which action is the nearest in the *FFxIVFF* space To this aim different similarity, distance and correlation functions have been used, the recognition accuracy has been calculated. During this phase we have to "operate" action by action (class by class), on the *FF* values hit by the query in the *Traing Set table*.

Not all the interval are represented in every action of the *Traing Set table*. If the interval hit by the query is in the given action (class), an *FF* values is present, in the opposite case, no *FF* is associated with the interval: this is marked using an *n* value (null) (See Fig. 8).



	rise from bed		rise from chair	
Features	FF*IVFF	IVFF	FF*IVFF	IVFF
MtxSensor_1_Accelerometer_X_Axis_Raw_Max	0,11211	0,154151	0,140137	0,154151
MtxSensor_1_Accelerometer_X_Axis_Raw_Min	0,098096	0,154151	0,140137	0,154151
MtxSensor_1_Accelerometer_X_Axis_Raw_Starting Value	1,923853	2,351375	n	0,04879
MtxSensor_1_Accelerometer_X_Axis_Raw_Ending Value	0,039919	0,04879	0,044355	0,04879
MtxSensor_1_Accelerometer_X_Axis_Raw_Average	0	0	0	0
MtxSensor_1_Accelerometer_X_Axis_Raw_Variance	0,385135	0,847298	n	0,154151
MtxSensor_1_Accelerometer_X_Axis_Raw_Zero Crossing Rate	0,126123	0,154151	0,140137	0,154151
MtxSensor_1_Accelerometer_X_Axis_Raw_Mean Crossing Rate	n	2,351375	0,213761	2,351375
MtxSensor_1_Accelerometer_X_Axis_Raw_Skewness	0,184302	0,405465	0,073721	0,405465
MtxSensor_1_Accelerometer_X_Axis_Raw_Kurtosis	1,50748	1,658228	0,150748	1,658228
MtxSensor_1_Accelerometer_X_Axis_Smoothing_Max	0,098885	0,271934	0,049442	0,271934
MtxSensor_1_Accelerometer_X_Axis_Smoothing_Min	0,035484	0,04879	0,044355	0,04879
MtxSensor_1_Accelerometer_X_Axis_Smoothing_Starting Value	1,592108	1,94591	n	0,04879
MtxSensor_1_Accelerometer_X_Axis_Smoothing_Ending Value	0,04879	0,04879	0,044355	0,04879
MtxSensor_1_Accelerometer_X_Axis_Smoothing_Average	0	0	0	0
MtxSensor_1_Accelerometer_X_Axis_Smoothing_Variance	0,45555	1,252763	n	0,04879
MtxSensor_1_Accelerometer_X_Axis_Smoothing_Zero Crossing Rate	0,039919	0,04879	0,044355	0,04879
MtxSensor_1_Accelerometer_X_Axis_Smoothing_Mean Crossing Rate	1,435085	1,435085	0,130462	1,435085
MtxSensor_1_Accelerometer_X_Axis_Smoothing_Skewness	0,123606	0,271934	0,074164	0,271934
MtxSensor_1_Accelerometer_X_Axis_Smoothing_Kurtosis	0,405465	0,405465	0,110581	0,405465
MtxSensor_1_Accelerometer_X_Axis_Fourier_Max	0,110581	0,405465	0,073721	0,405465

Fig. 8 – Portion of the Training Set Table. The execution of a query.

We can see represented in Fig. 24 the results of a possible execution of a *query* on the *Training Set*. The term *"n"* (null) indicates that an interval is present in the *query* but not in the given Action (class) of the *Training Set*.. If an *"n"* value is present it is not possible calculate the *FFxIVFF* and the similarty of a given Action (class) in respect to the given *query*.

To deal with the *"n"* interval problem, different policies can be adopted that lead to very significant variations in accuracy. The used policy is as follows:

$$x_{i,j} = \begin{cases} Ff_{i,j} * IVFf_i & if\ Ff_{i,j} \neq n \\ 0 & if\ Ff_{i,j} = n \end{cases}$$

$$y_i = \begin{cases} IVFf_i & if\ Ff_{i,j} \neq n \\ Max(IVFf_i) & if\ Ff_{i,j} = n \end{cases}$$

$$Max(IVFf_i) = \log|D|$$

Where $1 \leq i \leq N$ e $1 \leq j \leq D$ vary on the number of N features and D actions (classes) present in the vocabulary. $|D|$ is the cardinality of the vocabulary of actions (the number of classes); $x_{i,j}$ represents the weight of the σ_i feature of the A_j action of the *Training-Set*; y_i is the *IVFF* value of the correlated *interval-feature*.

In short, if the value of *FF* is missing, the value $x_{i,j}$ approaches zero, while the y_i value is set equal to the maximum possible value of *IVFF*. The role of $x_{i,j}$ and y_i in the calculus of similarity has been explained in detail in the next section (see also **Appendix D - Classifiers**).

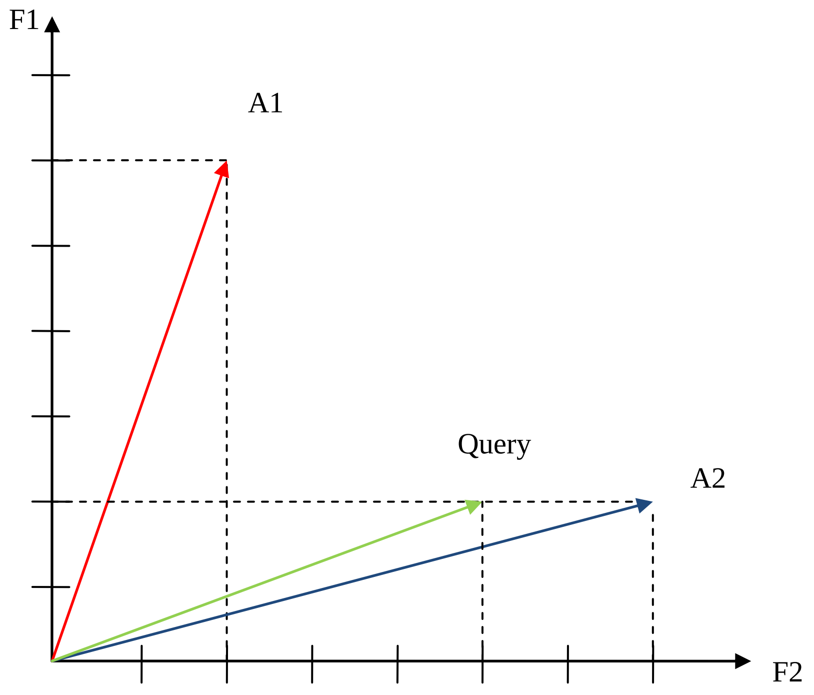


Fig. 9 - A to-be-recognized sample is called query, this is a vector in the feature-interval space.

The *Training Set* table (shown in Fig. 8) can be viewed as a set of column vectors. The algorithm compares the query vector with each of these vectors, applying a classification function and returning a sorted result.

Classifiers

A large number of classifiers that belong to the following categories has been tested:

• Similarities
• Distances
• Correlation

That was done in order to understand which differences are present and what could be the role of these classifiers. Different measures of distance or similarity are convenient for different types of analysis. As we will see (Test Results, p. 59) some important differences in term of results are present in some cases, but usually the average results are quite similar. In fact, the operation of *FFxIVFF* feature weighting has the effect to change the density and distance of the clusters (see Intracluster and Intercluster Similarity, p. 127). Given that in the *FFxIVFF* space clusters are more dense and separated, it is possible to use computationally quick classifiers, with very good results. The complete list of classifiers is:

- Similarity:

  Ranking
  Cosine Similarity
  Tanimoto
  Simpson
  Braun-Blanquet
  Kulczynski 1
  Kulczynski 2
  Dice
  Otsuka
  MountFord
  Nclassifier
  Nweightclassifier
  Entropy

- Distance:

  Euclidean
  Manhattan
  Camberra
  Chi-Quadro
  Hellinger-Quadro
  Hamming
  Bray-Curtis
  Min variance

- Correlation:

  Pearson

For a detailed description of the given classifiers see **Appendix D - Classifiers**

Computational Costs

The given classifiers operate on a data structure organized as follows:

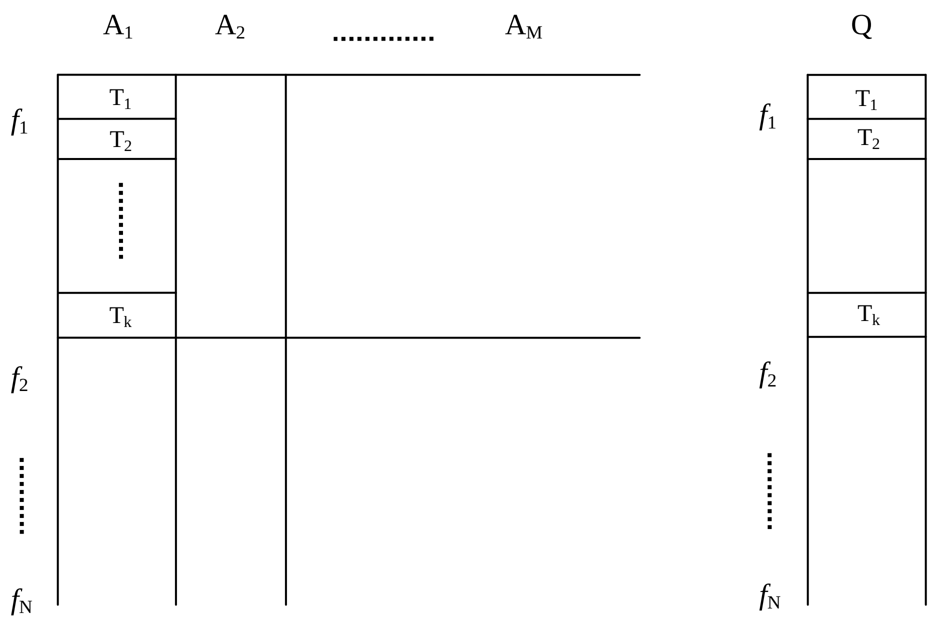


Fig. 10 - Training Set and Query data structures

Where $f_{1,}$ f_2 ... f_N are the *features*, $T_{1,}$ T_2 … T_k are the *interval-features* e A_1 , A_2 .. A_M are the Actions of the *Training Set; N is the number of features, M the number of actions.*

Given a *query* Q, and an action A_j, the classifiers check if the interval T_S of the *feature* f_i is in the *Training Set.* Then, it executes a simple mathematical operation (sum, product, subtraction, division) using the *FF* and *IVFF* weights. One compared the query with all the Actions of *Training Set,* the algorithm order the obtained values.

In first approximation, assuming that mathematical operations have costant complexity of *O(1),* the computational cost of the classifier is:

$$O(N * M) + O(\mathrm{M} * \log M)$$

where $O(N * M)$ is the cost to run through the bidimensional *Training Set* table and $O(\mathrm{M} * \log M)$ is the cost of sorting the results.

There is no reason to think that the number of *features* should indefinitely grow, actually the number of features is usually reduced with the operation of features reduction. Under this condition we can consider N as a costant, and $\mathrm{O}(N * M)$ become $\mathrm{O}(M)$, hence the prevailing cost is the sorting cost $O(\mathrm{M} * \log M)$.

On the other side, for small values of M, when $N \gg M$, the prevailing cost is $O(N * M)$.

Given that some classification algorithms like the *Pearson* correlation, *Min variance* run through the *Training Set* table more than one time, the computational cost of these algorithms is higher for a constant factor.

In general we can say that the described similarity and distance classifier have low computational costs, and correlation classifier have a higher computational cost.

From a computational point of view, we can hardly imagine situations in which M (the number of actions'class) has an order of magnitude really high. In some particular contexts, such in professional karate, we can imagine to have a number of actions of the order of 10^3.

In principle, we could imagine a situation in which every action is a combination of two (or more) basic gestures, performed by different subjects. In this way we could create a set of basic gestures to expand the actual size of the dictionary of actions that could be recognized. In this case, since we are providing the Cartesian product of the given basic gestures of the training set, the actual number of recognizable actions may rise to really important dimensions, even using relatively small *Training Set* tables.

The next chapter gives a description of the two databases used to test the performance of the described classification algorithms.

NIDA Database

The NIDA 1.0 (Nomadis Internal Database of Actions) database contains movements acquired by the NOMADIS Laboratory of the University of Milano-Bicocca. These acquisitions have been obtained using 5 MTx sensors positioned on the pelvis, on the right and left wrist, and on the right and left ankle. NIDA includes 21 types of actions performed by 7 people (5 males and 2 females) ranging from 19 to 44-year-old, repeated twice time, for a total of 273 actions. The database has a rich vocabulary. The complete list is the following:

1. Get up from bed.
2. Get up from a chair.
3. Open a wardrobe.
4. Open a door.
5. Fall.
6. Walk forward
7. Run.
8. Turn left 180 degrees.
9. Turn right 180 degrees.
10. Turn left 90 degrees.
11. Turn right 180 degrees.
12. Karate frontal kick.
13. Karate side kick.
14. Karate punch.
15. Go upstairs.
16. Go downstairs.
17. Jump.
18. Write.
19. Lie down on a bed.
20. Sitting on a chair
21. Heavily sitting on a chair

WARD Database

WARD 1.0 (Wearable Action Recognition Database) was created at UC Berkeley. Acquisitions have been obtained positioning 5 sensors on the pelvis, on the right and left wrist, and on the right and left ankle. Each sensors has a 3-axial accelerometer and a 2-axial gyroscope, magnetometers are not present.
Data have been calibrated and normalized (mapped) to their appropriate unit of measure before using them for the training phase, as suggested by the creators of the database [5].
WARD contains 13 types of actions performed by 20 people (13 men and 7 women) ranging from 20 to 79-years-old with 5 repetitions per action, for a total of 1300 samples. The complete list of actions is given in the next section.

Warable sensor network

WARD acquisitions has been done through a *wearable sensor network*. This network of sensors is based on a platform called DexterNet (see [5]). DexterNet is an open-source platform for sensors that adopts a three-tier architecture to control a group of heterogeneous sensors mounted on the body. The three levels are: *body sensor layer*, *global personal network layer* and *network layer*.
In terms of *body sensor layer*, the platform is able to integrate and control various types of sensors including MICAz, SHIMMER and GPS. In terms of *personal network layers*, a mobile station (eg a PDA or SmartPhone) is used to communicate and to process the sensor data of a single subject. In terms of *global network layer*, subjects are interconnected via the Internet, to provide wireless coverage for the monitoring of both indoors and outdoors activities.
The Five sensors have been postioned with the same Sensor Profile used in our database (see Fig. 1). Sensors communicate with the mobile station connected to a computer server through a USB port. Each sensor has a triaxial accelerometer and a dual-axial gyroscope. For each axis the values vary in the range of $\pm 2g$ and $\pm 500°/s$ respectively for accelerometers and gyroscopes.

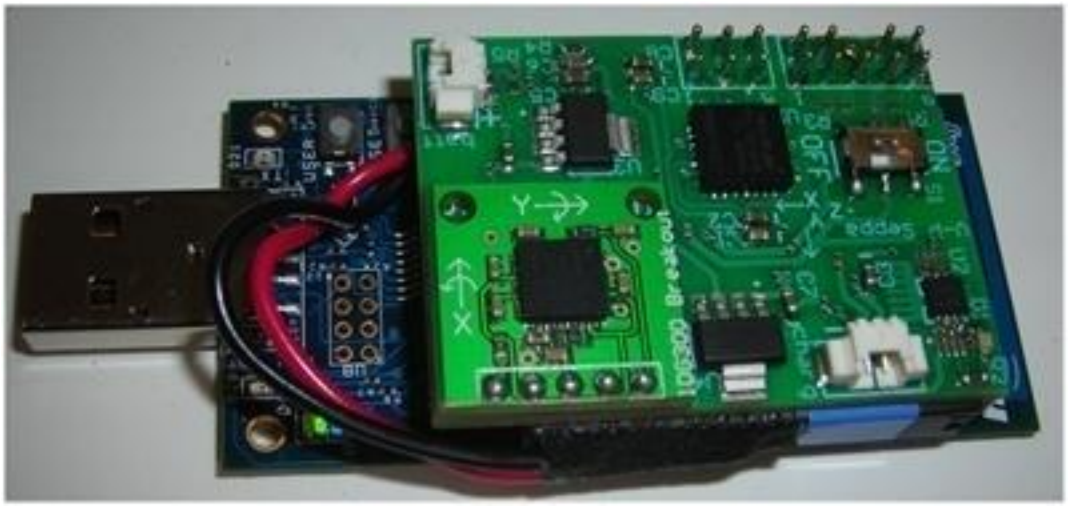


Fig. 11 – The sensor board of U.C. Berkeley

WARD database

The WARD database respects the following criteria:

1. The database contains a significant number of subjects with people of different ages.
2. The given activities cover typical actions of daily life.
3. The *Sensor Profile* used for WARD is identical to the *Body Sensor profile* of the NIDA database

The WARD database can be downloaded at: http://www.eecs.berkeley.edu/~yang/software/WAR/
Data are related to 20 subjects (7 men, 13 women) ranging from 19 to 75 years and includes the following 13 categories of actions:

1. *Rest at Standing* - (*at least 10 sec)*
2. *Rest at Sitting - (at least 10 sec)*
3. *Rest at Lying* - *(at least 10 sec)*
4. *Walk forward* - *(at least 10 sec)*
5. *Walk forward left-circle* - *(at least 10 sec)*
6. *Walk forward right-circle* - *(at least 10 sec)*.
7. *Turn left* - *(the subject remains on place truning right for more than 10 seconds).*
8. *Turn right* - *(the subject remains in place and turning left for more than 10 seconds)*

9. *Go upstairs - (the subject goes up a flight of stairs)*
10. *Go downstairs - (the subject goes down a flight of stairs)*
11. *Jogging - (at least 10 sec).*
12. *Jumping – (the subject remains on place and jumps more than 5 times)*
13. *Push wheelchair – (the subject is seated in a wheelchair and moves for more than 10 seconds).*

For the acquisitions 5 sensors were used in an X configuration (see Figure 1). The sensors were respectively placed:

• sensor 1 and 2 on the left wrist and right;
• 3 sensor on the side;
• sensor 4 and 5 on the right and left ankles.

Given 13 types of actions, 20 subjects, and that each subject executes the same action 5 times (trial) the total number of samples is 20 * 5 * 13 = 1300. The acquistion has been stored in MATLAB format. Each file contains a MATLAB structure that stores the data of the five sensors. Each sensor contains *5 * t* , where *t* represents the length of the sequence.
The database contains some actions with a loss of information caused mainly by insufficient battery power or packet loss in the transmission network. These files are indicated in MATLAB with *"Inf"*. The total number of these corrupted samples is 27 and they were not used for the analysis. MATLAB data were converted to CSV files and acquired into our analysis server for training and recognition

WARD Data Analysis

The "wearable sensors used by U.C. Berkely are subject to the drifting problem. Even in the same type of action in subsequent trials of the same subject, using the same sensor a shift can be observed in the mean values of accelerations and gyroscopes.

It was necessary to appropriately map the data on the range of values used by the *MTx* sensors, in order to start the recognition process, without change any part of the architecture used for *MTx*. The process is described in details in **Appendix E – Calibration**

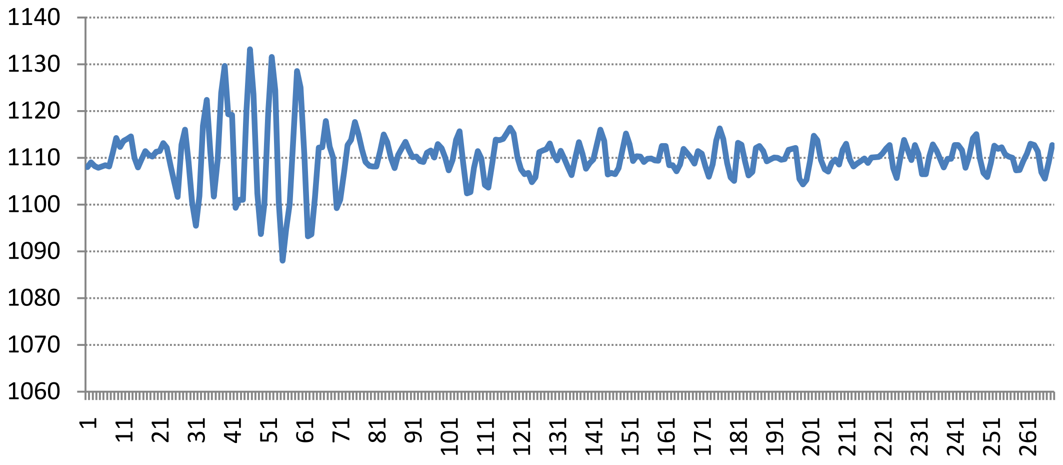


Fig. 12 – Magnitude of accelerometer, trial 4 of Subject 9, left wrist, "Rest at standing",

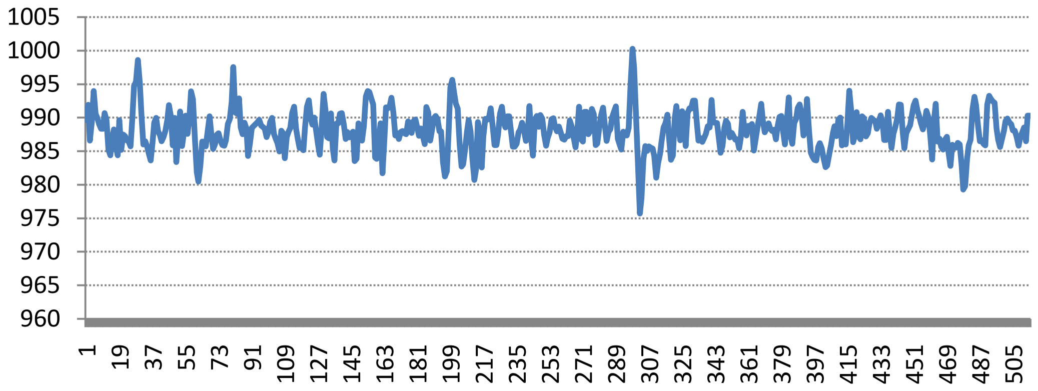


Fig. 13 – Magnitude of accelerometer, trial 5 of Subject 9, left wrist, "Rest at standing",

In fig. 12 and Fig. 13 it is possible to see the magnitude of the accelerometers of two different trials, they differs in average by about 100 values. This should not happen in actions like *Rest at Standing*, *Rest at Sitting* and *Rest at Lying*, where magnitude of the accelerometers should be similar.

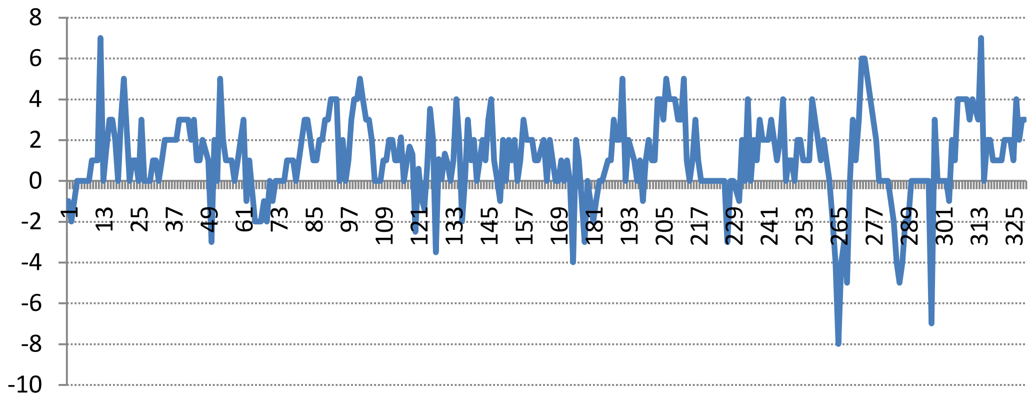


Fig. 14 – Trial 1, X axis of the giroscope, subject 4, left wrist, "Rest at standing".

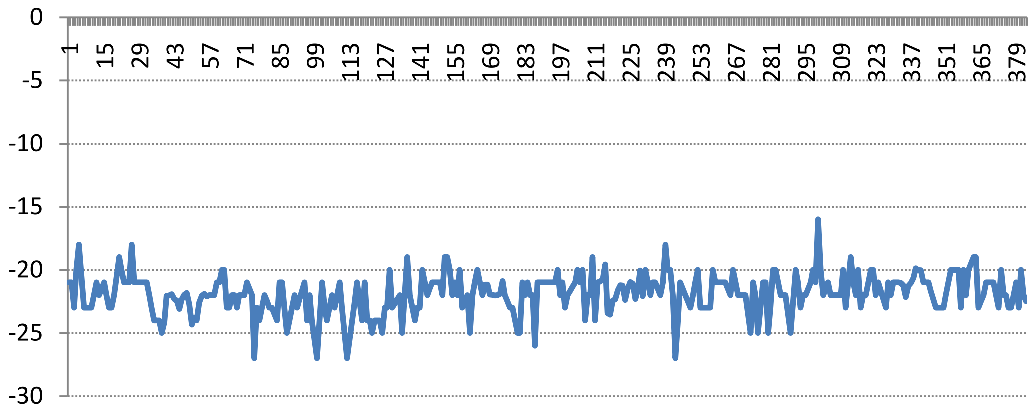


Fig. 15– Trial 1, X axis of the giroscope, subject 4, left wrist, "Rest at standing".

Since the gyroscope measures the rate of turn, we expect that in the first three measures the average value of the gyroscope in the X and Y should be close to zero. As we can see in Fig. 15, this is not always true. To resolve this problem, it was used a procedure of calibration and normalization (mapping). Data must be calibrated and appropriately mapped on the range of values used by the *MTx* sensors, in order to start the training and recognition process, without changing any part of the architecture used for *MTx*. The process is described in details in **Appendix E - Calibration** .

Testing Methods

To check the effects of the *FFxIVFF* operation and to test the performarnce of the given classification functions in the *FFxIVFF* space for the movement recognition, several tests have been performed. Test were done with an *"off-line"* analysis approach using the two formerly described databases (NIDA 1.0 and WARD 1.0, p.44).
Two type of approaches has been used for tests: *Leave One Out Cross-Validation (LOOCV)* and *Majority Voting Combinations (MVC)*..

Leave One Out Cross-Validation (LOOCV)

The *Leave One Out Cross-Validation (LOOCV)* leave one sample in the *Test Set* and all the others sample in the *Training Set*. In our tests we leave all the sample of a person in the *TestSet*, and all the other people's sample in the *Training Set*. *LOOCV* was preferred to other approaches because of the relatively low number of samples present in the two dabasases.
For every action of the *Test Set* (query), the classification function returns at first what is considered the most similar action in the given *Training Set*, then all the other actions in order of importance (from the most to the least similar).
The results given by different classification function during the recognition phase are plausibly comparable when classification function performs well.
For example, if the *Training Set* contains the following actions:

Action	Identifier
Walk	0
Run	1
Going Downstair	2
Jump	3
Going Upstair	4

Table 4 – example of Training Set

A classification will produce a similar output:

Test Set 1

Executed Action	Recognized				
0	0	4	1	3	2
1	1	3	4	0	2
2	2	4	0	3	1
3	3	1	0	4	2
4	4	0	3	1	2

Table 5 – example of results of a classification on Test Set 1

Test Set 2

Executed	Recognized				
0	0	1	4	3	2
1	1	3	0	4	2
2	2	4	0	1	3
3	3	1	0	4	2
4	4	0	1	3	2

Table 6 – example of results of a classification on Test Set 2

In Table 6 and 7 we can see in the first column the label relative to executed actions (the *Ground Truth*), in the second column (blue) the labels of the recognized actions. In this example the classifier has correctly recognized all the actions of the two *Test Sets*. Finally, all the results are collected in a so called *Confusion Matrix* that displays how many times a particular action has been recognized for *all* the run.
The Confusion Matrix has the advantage to represent the tests result in a synoptic way.

	0	**1**	**2**	**3**	**4**
0	2	0	0	0	0
1	0	2	0	0	0
2	0	0	2	0	0
3	0	0	0	2	0
4	0	0	0	0	2

Table 7 – example of Confusion Matrix of Tes Set 1 Tes tSet2

Given the *Confusion Matrix* of the classifier, then the accuracy is calculated as follows:

$$Accuracy = \frac{\sum_{i=1}^{N} ConfusionMatrix(i,i)}{\sum_{i=1}^{N}\sum_{j=1}^{N} ConfusionMatrix(i,j)} * 100$$

where $ConfusionMatrix(i,j)$ indicates the i ,j, element of the matrix, N is the number of rows (and columns) of the matrix.

Majority Voting

In line of principle, if we have different classifier with a good accuracy that gives different results it is possible to decide which is the most similar action through a *Majority Voting*. For example given the three classifiers:

Classifier 1

Execued	**Recognized**				
0	0	0	2	1	3
1	1	3	4	0	2
2	2	4	0	3	1
3	3	1	0	4	2
4	4	2	0	3	1

Classifier 2

Executed	**Recognized**				
0	2	4	0	1	3
1	1	3	4	0	2
2	4	2	0	3	1
3	3	1	0	4	2
4	4	2	0	3	1

Classifier 3

Executed	**Recognized**				
0	4	2	2	1	3
1	1	3	4	0	2
2	2	4	0	3	1
3	0	1	3	4	2
4	4	2	0	3	1

The *Majority Voting* returns:

Executed	**Recognized**
0	4
1	1
2	2
3	3
4	4

Table 9 - Majority Voting of the given results

In case of parity, for example when all the classifiers recognize different actions, the choice is done giving a *preference* to a specific classifier. In Tab. 9 *Majority Voting* has been applied with a *preference* for *Classifier1*.

This method produces good results only if the most part of the classifiers have a good accuracy, and if the classifiers make errors in different regions of the space, i.e. on different actions.

Majority Voting Combinations

The *Majority Voting Combinations* is an extension of the LOOCV test. It performs all the possible combinations of *Majority Voting* among all the classifiers, varying at every run the classifier of preference.

The purpose of this test is to find the best combination of algorithms that maximize accuracy. Since it is possible that different combinations of classifiers give the same accuracy, in case of parity, has been preferred the combination with the lower computational costs. The method find the combination that minimizes the time of execution and maximizes the accuracy.

For example, given 3 generic classifiers A_1, A_2 e A_3 the number of C_i combination with preference is 6.

$Preference\ A_1$ $\quad$ $Preference\ A_2$ $\quad$ $Preference\ A_3$

$C_1 = A_1$ $\quad$ $C_2 = A_2$ $\quad$ $C_3 = A_3$

$Preference\ A_1$ $\quad$ $Preferenza\ A_2$ $\quad$ $Preferenza\ A_3$

$$C_4 = \begin{cases} A_1 \\ A_2 \\ A_3 \end{cases} \qquad C_5 = \begin{cases} A_1 \\ A_2 \\ A_3 \end{cases} \qquad C_6 = \begin{cases} A_1 \\ A_2 \\ A_3 \end{cases}$$

Combinations with preference of cardinality 2, gives the same results of the combinations with cardinality 1, so they are not taken into account.

The method evaluates the accuracy of each combination, create the *Majority Confusion Matrix* and if there are combinations with same accuracy, results are ordered by computational cost.

The total number of combinations is given by the formula:

$$Combinations = \sum_{i=1, i\neq 2}^{NumAlg} \binom{NumAlg}{i} * i$$

Where $NumAlg$ is the total number of classifiers, $\binom{NumAlg}{i}$ are the combinations without repetition, and because at each execution we have a classifier of preference is set, the binomial factor is multiplied by i . As already said, combinations with cardinality 2 are not used.

Classifiers Number	**Total combination**
3	6
4	20
5	60
6	162
7	406
8	968
9	2.232
10	5.030
11	11.154
12	24.444
13	53.092
14	114.506
15	245.550
16	524.048
17	1.113.840
18	2.358.990
19	4.980.394
20	10.485.380
21	22.019.676
22	46.136.882

Table 10 - The exploding number of tests using combinations of classifiers in Majority Voting Combination

Performing this test is computational expensive, in fact the number of combinations increases very quickly, increasing the number of used classifiers (see Table 10).

To decrease the number of combinations you can select only the classifiers that performs best as a single classifiers (this can be easily established) or exclude the classifiers that give the same false positives on the same actions.

Test Results NIDA

Tests have been done using both WARD 1.0 and NIDA 1.0 databases in order to understand the effects of the *FFxIVFF* operation and to test the accuracy of the given classification functions, and their combined use in the *FFxIVFF* space.

As we saw in previous chapters, the two databases differ in the number of acquisitions (the size of WARD 1.0 is almost 5 times the dimension of NIDA 1.0) and the type and number of actions (WARD 1.0 contains 13 actions against the 21 of NIDA 1.0). Tests has also been performed varying the number of sensors in order to see how vary the accuracy of recognition.

These tests are important because they show us how a body part is involved in recognition of a particular movement, and can tell us how many sensors are actually necessary to recognize an action and where they can be positioned. Tests results gives the *Confusion Matrix* and *Accuracy* of every single test.

For tests with a reduced number of sensors, has been used the classifier that has the highest accuracy for the given dataset..

Test using NIDA database

The tests performed on the NIDA database are as follows. At first all the formerly described classifiers have been taken and tested one at the time (Test 1), then they have been tested combining them (Test 2):

- Test 1 – LOOCV: all classifiers with all the sensors
- Test 2 - Majority Voting Combinations of all classifiers with all the sensors

Then the single classifier with the best accuracy has been taken, the Bray-Curtis, and the accuracy has been tested varying the number and position of sensors on the body:

- Test 3 - LOOCV with hip, right wrist, right ankle sensors
- Test 4 - LOOCV hip, left wrist, left ankle sensors
- Test 5 - LOOCV right wrist, right ankle sensors
- Test 6 - LOOCV left wrist left ankle sensors
- Test 7 - LOOCV hip, right wrist, left ankle sensors
- Test 8 - LOOCV hip, left wrist, right ankle sensors

- Test 9 - LOOCV right wrist, left ankle sensors
- Test 10 - LOOCV left wrist, right ankle sensors
- Test 11 - LOOCV hip sensor
- Test 12 - LOOCV right wrist sensor
- Test 13 - LOOCV right ankle sensor

The *legenda* to identify the actions is as follows:

Action	Identifier
Get up from bed	0
Get up from a chair	1
Open a wardrobe	2
Open a door	3
Fall	4
Walk forward	5
Run	6
Turn left 180 degrees	7
Turn right 180 degrees	8
Turn left 90 degrees	9
Turn right 180 degrees	10
Karate frontal kick	11
Karate side kick	12
Karate punch	13
Go upstairs	14
Go downstairs	15
Jump	16
Write	17
Lying down on a bed	18
Sitting on a chair	19
Heavily sitting on a chair	20

Test 1 – LOOCV: All Classifiers with All the Sensors

Ranking classifier *Confusion Matrix*

	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**19**	**20**
0	13	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	8	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	5	0
2	0	0	13	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3	0	0	0	10	0	0	0	0	0	0	0	0	0	0	0	0	0	3	0	0	0
4	0	0	0	0	11	0	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5	0	0	0	0	0	13	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	13	0	0	0	0	0	0	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	12	0	1	0	0	0	0	0	0	0	0	0	0	0
8	0	0	0	0	0	0	0	4	6	0	3	0	0	0	0	0	0	0	0	0	0
9	0	0	0	0	0	0	0	0	0	12	1	0	0	0	0	0	0	0	0	0	0
10	0	0	0	0	0	0	0	0	0	0	13	0	0	0	0	0	0	0	0	0	0
11	0	0	0	0	0	0	0	0	0	0	0	12	1	0	0	0	0	0	0	0	0
12	0	0	0	0	1	0	2	0	0	0	0	0	10	0	0	0	0	0	0	0	0
13	0	0	0	0	0	0	0	0	0	0	0	0	0	13	0	0	0	0	0	0	0
14	0	0	0	0	0	0	0	0	0	0	0	0	0	0	13	0	0	0	0	0	0
15	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	12	0	0	0	0	0
16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	13	0	0	0	0
17	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	13	0	0	0
18	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	13	0	0
19	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	12	0
20	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	2	10

$$Accuracy = \ 89{,}74\%$$

Cosine Similarity classifier *Confusion Matrix*

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
0	13	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	12	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
2	0	0	12	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
3	0	0	0	13	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	13	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5	0	0	0	0	0	13	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	13	0	0	0	0	0	0	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	13	0	0	0	0	0	0	0	0	0	0	0	0	0
8	0	0	0	0	0	0	0	3	10	0	0	0	0	0	0	0	0	0	0	0	0
9	0	0	0	0	0	0	0	0	0	12	1	0	0	0	0	0	0	0	0	0	0
10	0	0	0	0	0	0	0	0	0	0	13	0	0	0	0	0	0	0	0	0	0
11	0	0	0	0	0	0	0	0	0	0	0	12	1	0	0	0	0	0	0	0	0
12	0	0	0	0	2	0	0	0	0	0	0	0	11	0	0	0	0	0	0	0	0
13	0	0	0	0	0	0	0	0	0	0	0	0	0	13	0	0	0	0	0	0	0
14	0	0	0	0	0	0	0	0	0	0	0	0	0	0	13	0	0	0	0	0	0
15	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	12	0	0	0	0	0
16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	13	0	0	0	0
17	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	13	0	0	0
18	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	13	0	0
19	0	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	11	0
20	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	12

$$Accuracy = \ 95{,}23\%$$

Tanimoto classifier *Confusion Matrix*

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
0	13	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	11	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1
2	0	0	13	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3	0	0	0	13	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	13	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5	0	0	0	0	0	13	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	13	0	0	0	0	0	0	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	13	0	0	0	0	0	0	0	0	0	0	0	0	0
8	0	0	0	0	0	0	0	3	9	0	1	0	0	0	0	0	0	0	0	0	0
9	0	0	0	0	0	0	0	0	0	12	1	0	0	0	0	0	0	0	0	0	0
10	0	0	0	0	0	0	0	0	0	0	13	0	0	0	0	0	0	0	0	0	0
11	0	0	0	0	0	0	0	0	0	0	0	12	1	0	0	0	0	0	0	0	0
12	0	0	0	0	0	0	1	0	0	0	0	0	12	0	0	0	0	0	0	0	0
13	0	0	0	0	0	0	0	0	0	0	0	0	0	13	0	0	0	0	0	0	0
14	0	0	0	0	0	0	0	0	0	0	0	0	0	0	13	0	0	0	0	0	0
15	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	12	0	0	0	0	0
16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	13	0	0	0	0
17	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	13	0	0	0
18	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	13	0	0
19	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	13	0
20	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	1	11

$$Accuracy = \ 95{,}60\%$$

Simpson classifier *Confusion Matrix*

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
0	0	0	0	0	3	0	0	0	0	0	0	1	8	0	0	1	0	0	0	0	0
1	0	0	0	0	4	0	0	0	0	0	0	0	6	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0	0	0	0	2	0	0	11	0	0	0	0	0
3	0	0	0	0	4	0	0	0	0	0	0	0	8	0	0	1	0	0	0	0	0
4	0	9	0	0	0	0	0	0	2	0	0	1	0	0	0	0	0	0	0	0	1
5	0	0	0	0	0	0	0	0	0	0	0	0	0	6	0	0	0	0	0	0	7
6	0	6	0	0	0	0	0	0	4	0	0	0	0	0	0	0	0	0	0	3	0
7	0	0	0	0	11	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0
8	0	0	0	0	10	0	0	0	0	0	0	2	0	0	0	1	0	0	0	0	0
9	0	0	0	0	12	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
10	0	0	0	0	9	0	0	0	0	0	0	1	2	0	0	1	0	0	0	0	0
11	0	0	0	0	0	1	0	0	1	0	0	0	0	6	0	0	0	0	0	0	5
12	0	6	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	1	5
13	0	0	0	0	0	8	0	0	0	0	0	3	2	0	0	0	0	0	0	0	0
14	0	1	0	0	0	0	0	0	0	0	0	0	0	11	0	0	0	0	0	0	1
15	0	10	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	2	0
16	0	2	0	0	0	0	0	0	0	0	0	0	0	10	0	0	0	0	0	0	1
17	0	0	0	0	1	0	0	0	0	0	0	0	3	0	0	9	0	0	0	0	0
18	0	0	0	0	0	0	0	0	0	0	0	0	12	0	0	1	0	0	0	0	0
19	0	0	0	0	4	0	0	0	0	0	0	0	5	0	0	4	0	0	0	0	0
20	0	0	0	0	0	1	0	0	0	0	0	0	11	0	0	1	0	0	0	0	0

$$Accuracy = \ 0\%$$

Braun-Blanquet classifier *Confusion Matrix*

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
0	13	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	10	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	1
2	0	0	13	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3	0	0	0	13	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	13	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5	0	0	0	0	0	13	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	13	0	0	0	0	0	0	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	13	0	0	0	0	0	0	0	0	0	0	0	0	0
8	0	0	0	0	0	0	0	3	9	0	1	0	0	0	0	0	0	0	0	0	0
9	0	0	0	0	0	0	0	0	0	12	1	0	0	0	0	0	0	0	0	0	0
10	0	0	0	0	0	0	0	0	0	0	13	0	0	0	0	0	0	0	0	0	0
11	0	0	0	0	0	0	0	0	0	0	0	12	1	0	0	0	0	0	0	0	0
12	0	0	0	0	0	0	1	0	0	0	0	0	12	0	0	0	0	0	0	0	0
13	0	0	0	0	0	0	0	0	0	0	0	0	0	13	0	0	0	0	0	0	0
14	0	0	0	0	0	0	0	0	0	0	0	0	0	0	13	0	0	0	0	0	0
15	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	12	0	0	0	0	0
16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	13	0	0	0	0
17	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	13	0	0	0
18	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	13	0	0
19	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	13	0
20	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	1	11

$$Accuracy = \ 95{,}23$$

Kulcynski 1 classifier *Confusion Matrix*

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
0	13	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	11	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1
2	0	0	13	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3	0	0	0	13	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	13	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5	0	0	0	0	0	13	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	13	0	0	0	0	0	0	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	13	0	0	0	0	0	0	0	0	0	0	0	0	0
8	0	0	0	0	0	0	0	3	9	0	1	0	0	0	0	0	0	0	0	0	0
9	0	0	0	0	0	0	0	0	0	12	1	0	0	0	0	0	0	0	0	0	0
10	0	0	0	0	0	0	0	0	0	0	13	0	0	0	0	0	0	0	0	0	0
11	0	0	0	0	0	0	0	0	0	0	0	12	1	0	0	0	0	0	0	0	0
12	0	0	0	0	0	0	1	0	0	0	0	0	12	0	0	0	0	0	0	0	0
13	0	0	0	0	0	0	0	0	0	0	0	0	0	13	0	0	0	0	0	0	0
14	0	0	0	0	0	0	0	0	0	0	0	0	0	0	13	0	0	0	0	0	0
15	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	12	0	0	0	0	0
16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	13	0	0	0	0
17	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	13	0	0	0
18	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	13	0	0
19	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	13	0
20	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	1	11

$$Accuracy = \ 95{,}60\%$$

Kulcynski 2 classifier *Confusion Matrix*

	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**19**	**20**
0	0	0	0	0	3	0	0	0	0	0	0	1	8	0	0	1	0	0	0	0	0
1	0	0	0	0	7	0	0	0	0	0	0	0	6	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0	0	0	0	2	0	0	11	0	0	0	0	0
3	0	0	0	0	4	0	0	0	0	0	0	0	8	0	0	1	0	0	0	0	0
4	0	9	0	0	0	0	0	0	2	0	0	1	0	0	0	0	0	0	0	0	1
5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	7
6	0	6	0	0	0	0	0	0	4	0	0	0	0	0	0	0	0	0	0	3	0
7	0	0	0	0	11	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0
8	0	0	0	0	10	0	0	0	0	0	0	2	0	0	0	1	0	0	0	0	0
9	0	0	0	0	12	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
10	0	0	0	0	9	0	0	0	0	0	0	1	2	0	0	1	0	0	0	0	0
11	0	0	0	0	0	1	0	0	1	0	0	0	0	6	0	0	0	0	0	0	5
12	0	6	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	1	5
13	0	0	0	0	0	8	0	0	0	0	0	3	2	0	0	0	0	0	0	0	0
14	0	1	0	0	0	0	0	0	0	0	0	0	0	11	0	0	0	0	0	0	1
15	0	10	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	2	0
16	0	2	0	0	0	0	0	0	0	0	0	0	0	10	0	0	0	0	0	0	1
17	0	0	0	0	1	0	0	0	0	0	0	0	3	0	0	9	0	0	0	0	0
18	0	0	0	0	0	0	0	0	0	0	0	0	12	0	0	1	0	0	0	0	0
19	0	0	0	0	4	0	0	0	0	0	0	0	5	0	0	4	0	0	0	0	0
20	0	0	0	0	0	1	0	0	0	0	0	0	11	0	0	1	0	0	0	0	0

$$Accuracy = \ 0\%$$

Dice classifier *Confusion Matrix*

	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**19**	**20**
0	13	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	11	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1
2	0	0	13	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3	0	0	0	13	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	13	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5	0	0	0	0	0	13	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	13	0	0	0	0	0	0	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	13	0	0	0	0	0	0	0	0	0	0	0	0	0
8	0	0	0	0	0	0	0	3	9	0	1	0	0	0	0	0	0	0	0	0	0
9	0	0	0	0	0	0	0	0	0	12	1	0	0	0	0	0	0	0	0	0	0
10	0	0	0	0	0	0	0	0	0	0	13	0	0	0	0	0	0	0	0	0	0
11	0	0	0	0	0	0	0	0	0	0	0	12	1	0	0	0	0	0	0	0	0
12	0	0	0	0	0	0	1	0	0	0	0	0	12	0	0	0	0	0	0	0	0
13	0	0	0	0	0	0	0	0	0	0	0	0	0	13	0	0	0	0	0	0	0
14	0	0	0	0	0	0	0	0	0	0	0	0	0	0	13	0	0	0	0	0	0
15	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	12	0	0	0	0	0
16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	13	0	0	0	0
17	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	13	0	0	0
18	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	13	0	0
19	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	13	0
20	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	1	11

$$Accuracy = \ 95{,}60\%$$

Otsuka classifier *Confusion Matrix*

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
0	13	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	12	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
2	0	0	12	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
3	0	0	0	13	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	13	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5	0	0	0	0	0	13	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	13	0	0	0	0	0	0	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	13	0	0	0	0	0	0	0	0	0	0	0	0	0
8	0	0	0	0	0	0	0	3	10	0	0	0	0	0	0	0	0	0	0	0	0
9	0	0	0	0	0	0	0	0	0	12	1	0	0	0	0	0	0	0	0	0	0
10	0	0	0	0	0	0	0	0	0	0	13	0	0	0	0	0	0	0	0	0	0
11	0	0	0	0	0	0	0	0	0	0	0	12	1	0	0	0	0	0	0	0	0
12	0	0	0	0	2	0	0	0	0	0	0	0	11	0	0	0	0	0	0	0	0
13	0	0	0	0	0	0	0	0	0	0	0	0	0	13	0	0	0	0	0	0	0
14	0	0	0	0	0	0	0	0	0	0	0	0	0	0	13	0	0	0	0	0	0
15	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	12	0	0	0	0	0
16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	13	0	0	0	0
17	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	13	0	0	0
18	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	13	0	0
19	0	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	11	0
20	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	12

$$Accuracy = \ 95{,}23\%$$

Mountford classifier *Confusion Matrix*

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
0	0	0	0	1	0	0	0	0	0	0	0	0	12	0	0	0	0	0	0	0	0
1	0	4	0	0	0	0	0	0	0	0	0	1	8	0	0	0	0	0	0	0	0
2	0	0	1	0	0	0	0	0	0	0	0	1	11	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0	0	0	0	13	0	0	0	0	0	0	0	0
4	0	7	0	0	0	0	0	0	0	0	0	0	5	0	0	0	0	0	0	0	1
5	0	0	0	0	5	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	6
6	0	8	0	0	0	0	0	0	2	0	0	0	0	0	0	0	0	0	0	1	2
7	0	0	0	0	0	0	0	5	7	0	0	1	0	0	0	0	0	0	0	0	0
8	0	0	0	0	0	0	0	1	9	0	0	2	1	0	0	0	0	0	0	0	0
9	0	0	0	0	0	0	0	0	2	10	0	1	0	0	0	0	0	0	0	0	0
10	0	0	0	0	0	0	0	0	3	0	3	2	5	0	0	0	0	0	0	0	0
11	0	0	0	0	1	0	0	0	0	0	0	1	0	2	0	0	0	0	0	0	9
12	0	3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	10
13	0	0	0	0	0	0	0	0	0	0	0	10	3	0	0	0	0	0	0	0	0
14	0	0	0	0	0	0	0	0	0	0	0	0	0	1	4	0	0	0	0	0	8
15	0	6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	7
16	0	0	0	0	0	0	0	0	0	0	0	0	0	2	0	0	4	0	0	0	7
17	0	0	0	1	0	0	0	0	0	0	0	3	8	0	0	1	0	0	0	0	0
18	0	0	0	0	0	0	0	0	0	0	0	0	11	0	0	0	0	0	2	0	0
19	0	0	0	0	0	0	0	0	0	0	0	2	8	0	0	0	0	0	0	3	0
20	0	0	0	0	0	0	0	0	0	0	0	3	6	0	0	0	0	0	0	0	4

$$Accuracy = \ 20{,}14\%$$

NClassifier classifier *Confusion Matrix*

	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**19**	**20**
0	13	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	12	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
2	0	0	12	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
3	0	0	0	13	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	13	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5	0	0	0	0	0	13	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	13	0	0	0	0	0	0	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	13	0	0	0	0	0	0	0	0	0	0	0	0	0
8	0	0	0	0	0	0	0	3	10	0	0	0	0	0	0	0	0	0	0	0	0
9	0	0	0	0	0	0	0	1	0	11	1	0	0	0	0	0	0	0	0	0	0
10	0	0	0	0	0	0	0	0	1	0	12	0	0	0	0	0	0	0	0	0	0
11	0	0	0	0	0	0	0	0	0	0	0	12	1	0	0	0	0	0	0	0	0
12	0	0	0	0	0	0	0	0	0	0	0	0	13	0	0	0	0	0	0	0	0
13	0	0	0	0	0	0	0	0	0	0	0	0	0	13	0	0	0	0	0	0	0
14	0	0	0	0	0	0	0	0	0	0	0	0	0	0	13	0	0	0	0	0	0
15	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	13	0	0	0	0	0
16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	13	0	0	0	0
17	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	13	0	0	0
18	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	13	0	0
19	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	12	0
20	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	1	11

$$Accuracy = \ 95{,}60\%$$

NWeightClassifier classifier *Confusion Matrix*

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
0	13	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	13	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	0	0	12	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
3	0	0	0	13	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	12	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
5	0	0	0	0	0	13	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	13	0	0	0	0	0	0	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	12	0	1	0	0	0	0	0	0	0	0	0	0	0
8	0	0	0	0	0	0	0	2	9	1	1	0	0	0	0	0	0	0	0	0	0
9	0	0	0	0	0	0	0	1	0	12	0	0	0	0	0	0	0	0	0	0	0
10	0	0	0	0	0	0	0	0	3	1	9	0	0	0	0	0	0	0	0	0	0
11	0	0	0	0	0	0	0	0	0	0	0	12	1	0	0	0	0	0	0	0	0
12	0	0	0	0	0	0	0	0	0	0	0	3	10	0	0	0	0	0	0	0	0
13	0	0	0	0	0	0	0	0	0	0	0	0	0	13	0	0	0	0	0	0	0
14	0	0	0	0	0	0	0	0	0	0	0	0	0	0	13	0	0	0	0	0	0
15	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	10	0	0	0	0	0
16	0	0	0	0	0	0	0	0	0	0	0	0	2	0	0	0	13	0	0	0	0
17	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	13	0	0	0
18	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	13	0	0
19	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	12	0
20	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	12

$$Accuracy = \ 92{,}30\%$$

Entropy classifier *Confusion Matrix*

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
0	13	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	12	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
2	0	0	12	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
3	0	0	0	13	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	13	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5	0	0	0	0	0	13	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	13	0	0	0	0	0	0	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	12	0	1	0	0	0	0	0	0	0	0	0	0	0
8	0	0	0	0	0	0	0	3	10	0	0	0	0	0	0	0	0	0	0	0	0
9	0	0	0	0	0	0	0	0	0	12	1	0	0	0	0	0	0	0	0	0	0
10	0	0	0	0	0	0	0	0	1	0	12	0	0	0	0	0	0	0	0	0	0
11	0	0	0	0	0	0	0	0	0	0	0	12	1	0	0	0	0	0	0	0	0
12	0	0	0	0	2	0	0	0	0	0	0	1	10	0	0	0	0	0	0	0	0
13	0	0	0	0	0	0	0	0	0	0	0	0	0	13	0	0	0	0	0	0	0
14	0	0	0	0	0	0	0	0	0	0	0	0	0	0	13	0	0	0	0	0	0
15	0	0	0	0	3	0	0	0	0	0	0	0	2	0	0	8	0	0	0	0	0
16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	13	0	0	0	0
17	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	12	0
18	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	13	0	0
19	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	12	0
20	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	12

$$Accuracy = 88{,}27\%$$

Euclidean Distance classifier *Confusion Matrix:*

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
0	13	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	12	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
2	0	0	12	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
3	0	0	0	13	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	13	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5	0	0	0	0	0	13	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	13	0	0	0	0	0	0	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	13	0	0	0	0	0	0	0	0	0	0	0	0	0
8	0	0	0	0	0	0	0	3	9	0	1	0	0	0	0	0	0	0	0	0	0
9	0	0	0	0	0	0	0	1	0	11	1	0	0	0	0	0	0	0	0	0	0
10	0	0	0	0	0	0	0	0	2	0	11	0	0	0	0	0	0	0	0	0	0
11	0	0	0	0	0	0	0	0	0	0	0	12	1	0	0	0	0	0	0	0	0
12	0	0	0	0	0	0	0	0	0	0	0	0	13	0	0	0	0	0	0	0	0
13	0	0	0	0	0	0	0	0	0	0	0	0	0	13	0	0	0	0	0	0	0
14	0	0	0	0	0	0	0	0	0	0	0	0	0	0	13	0	0	0	0	0	0
15	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	13	0	0	0	0	0
16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	13	0	0	0	0
17	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	13	0	0	0
18	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	13	0	0
19	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	12	0
20	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	12

$$Accuracy = \ 95{,}23\%$$

Manhattan classifier *Confusion Matrix*

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
0	13	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	13	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	0	0	12	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
3	0	0	0	13	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	13	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5	0	0	0	0	0	13	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	13	0	0	0	0	0	0	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	13	0	0	0	0	0	0	0	0	0	0	0	0	0
8	0	0	0	0	0	0	0	3	9	0	1	0	0	0	0	0	0	0	0	0	0
9	0	0	0	0	0	0	0	1	0	11	1	0	0	0	0	0	0	0	0	0	0
10	0	0	0	0	0	0	0	0	2	0	11	0	0	0	0	0	0	0	0	0	0
11	0	0	0	0	0	0	0	0	0	0	0	12	1	0	0	0	0	0	0	0	0
12	0	0	0	0	0	0	0	0	0	0	0	1	12	0	0	0	0	0	0	0	0
13	0	0	0	0	0	0	0	0	0	0	0	0	0	13	0	0	0	0	0	0	0
14	0	0	0	0	0	0	0	0	0	0	0	0	0	0	13	0	0	0	0	0	0
15	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	13	0	0	0	0	0
16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	13	0	0	0	0
17	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	13	0	0	0
18	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	13	0	0
19	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	12	0
20	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	12

$$Accuracy = 95{,}23\%$$

Camberra classifier *Confusion Matrix*

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
0	13	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	12	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
2	0	0	13	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3	0	0	0	13	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	13	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5	0	0	0	0	0	13	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	13	0	0	0	0	0	0	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	12	0	1	0	0	0	0	0	0	0	0	0	0	0
8	0	0	0	0	0	0	0	2	8	1	2	0	0	0	0	0	0	0	0	0	0
9	0	0	0	0	0	0	0	0	0	12	1	0	0	0	0	0	0	0	0	0	0
10	0	0	0	0	0	0	0	0	1	0	12	0	0	0	0	0	0	0	0	0	0
11	0	0	0	0	0	0	0	0	0	0	0	12	1	0	0	0	0	0	0	0	0
12	0	0	0	0	0	0	0	0	0	0	0	3	10	0	0	0	0	0	0	0	0
13	0	0	0	0	0	0	0	0	0	0	0	0	0	13	0	0	0	0	0	0	0
14	0	0	0	0	0	0	0	0	0	0	0	0	0	0	13	0	0	0	0	0	0
15	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	12	0	0	0	0	0
16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	13	0	0	0	0
17	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	13	0	0	0
18	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	13	0	0
19	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	13	0
20	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	11

$$Accuracy = \ 94{,}13\%$$

Chi-Square classifier *Confusion Matrix*

	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**19**	**20**
0	13	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	13	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	0	0	12	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
3	0	0	0	13	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	13	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5	0	0	0	0	0	13	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	13	0	0	0	0	0	0	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	13	0	0	0	0	0	0	0	0	0	0	0	0	0
8	0	0	0	0	0	0	0	3	9	0	1	0	0	0	0	0	0	0	0	0	0
9	0	0	0	0	0	0	0	1	0	11	1	0	0	0	0	0	0	0	0	0	0
10	0	0	0	0	0	0	0	0	2	0	11	0	0	0	0	0	0	0	0	0	0
11	0	0	0	0	0	0	0	0	0	0	0	12	1	0	0	0	0	0	0	0	0
12	0	0	0	0	0	0	0	0	0	0	0	0	13	0	0	0	0	0	0	0	0
13	0	0	0	0	0	0	0	0	0	0	0	0	0	13	0	0	0	0	0	0	0
14	0	0	0	0	0	0	0	0	0	0	0	0	0	0	13	0	0	0	0	0	0
15	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	13	0	0	0	0	0
16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	13	0	0	0	0
17	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	13	0	0	0
18	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	13	0	0
19	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	12	0
20	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	12

$$Accuracy = \ 95{,}60\%$$

Hellinger-Square classifier *Confusion Matrix:*

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
0	13	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	12	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
2	0	0	12	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
3	0	0	0	13	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	13	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5	0	0	0	0	0	13	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	13	0	0	0	0	0	0	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	13	0	0	0	0	0	0	0	0	0	0	0	0	0
8	0	0	0	0	0	0	0	3	9	0	1	0	0	0	0	0	0	0	0	0	0
9	0	0	0	0	0	0	0	1	0	11	1	0	0	0	0	0	0	0	0	0	0
10	0	0	0	0	0	0	0	0	2	0	11	0	0	0	0	0	0	0	0	0	0
11	0	0	0	0	0	0	0	0	0	0	0	12	1	0	0	0	0	0	0	0	0
12	0	0	0	0	0	0	0	0	0	0	0	0	13	0	0	0	0	0	0	0	0
13	0	0	0	0	0	0	0	0	0	0	0	0	0	13	0	0	0	0	0	0	0
14	0	0	0	0	0	0	0	0	0	0	0	0	0	0	13	0	0	0	0	0	0
15	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	13	0	0	0	0	0
16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	13	0	0	0	0
17	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	13	0	0	0
18	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	13	0	0
19	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	12	0
20	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	12

$$Accuracy = \ 95{,}23\%$$

Hamming classifier *Confusion Matrix*

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
0	13	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	13	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	0	0	12	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
3	0	0	0	13	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	13	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5	0	0	0	0	0	13	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	13	0	0	0	0	0	0	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	13	0	0	0	0	0	0	0	0	0	0	0	0	0
8	0	0	0	0	0	0	0	3	9	0	1	0	0	0	0	0	0	0	0	0	0
9	0	0	0	0	0	0	0	1	0	12	0	0	0	0	0	0	0	0	0	0	0
10	0	0	0	0	0	0	0	0	3	1	9	0	0	0	0	0	0	0	0	0	0
11	0	0	0	0	0	0	0	0	0	0	0	12	1	0	0	0	0	0	0	0	0
12	0	0	0	0	0	0	0	0	0	0	0	3	10	0	0	0	0	0	0	0	0
13	0	0	0	0	0	0	0	0	0	0	0	0	0	13	0	0	0	0	0	0	0
14	0	0	0	0	0	0	0	0	0	0	0	0	0	0	13	0	0	0	0	0	0
15	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	12	0	0	0	0	0
16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	13	0	0	0	0
17	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	13	0	0	0
18	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	13	0	0
19	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	12	0
20	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	12

$$Accuracy = \ 93{,}77\%$$

Bray-Curtis classifier *Confusion Matrix*

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
0	13	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	13	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	0	0	13	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3	0	0	0	13	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	13	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5	0	0	0	0	0	13	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	13	0	0	0	0	0	0	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	13	0	0	0	0	0	0	0	0	0	0	0	0	0
8	0	0	0	0	0	0	0	3	9	0	1	0	0	0	0	0	0	0	0	0	0
9	0	0	0	0	0	0	0	0	0	12	1	0	0	0	0	0	0	0	0	0	0
10	0	0	0	0	0	0	0	0	0	0	13	0	0	0	0	0	0	0	0	0	0
11	0	0	0	0	0	0	0	0	0	0	0	12	1	0	0	0	0	0	0	0	0
12	0	0	0	0	0	0	1	0	0	0	0	0	12	0	0	0	0	0	0	0	0
13	0	0	0	0	0	0	0	0	0	0	0	0	0	13	0	0	0	0	0	0	0
14	0	0	0	0	0	0	0	0	0	0	0	0	0	0	13	0	0	0	0	0	0
15	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	12	0	0	0	0	0
16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	13	0	0	0	0
17	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	13	0	0	0
18	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	13	0	0
19	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	13	0
20	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	12

$$Accuracy = 96{,}70\%$$

Min variance classifier *Confusion Matrix*

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
0	13	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	13	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	0	0	12	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
3	0	0	0	13	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	13	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5	0	0	0	0	0	13	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	13	0	0	0	0	0	0	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	13	0	0	0	0	0	0	0	0	0	0	0	0	0
8	0	0	0	0	0	0	0	3	9	0	1	0	0	0	0	0	0	0	0	0	0
9	0	0	0	0	0	0	0	1	0	12	0	0	0	0	0	0	0	0	0	0	0
10	0	0	0	0	0	0	0	0	2	1	10	0	0	0	0	0	0	0	0	0	0
11	0	0	0	0	0	0	0	0	0	0	0	12	1	0	0	0	0	0	0	0	0
12	0	0	0	0	0	0	0	0	0	0	0	3	10	0	0	0	0	0	0	0	0
13	0	0	0	0	0	0	0	0	0	0	0	0	0	13	0	0	0	0	0	0	0
14	0	0	0	0	0	0	0	0	0	0	0	0	0	0	13	0	0	0	0	0	0
15	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0	11	0	0	0	0	0
16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	13	0	0	0	0
17	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	13	0	0	0
18	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	13	0	0
19	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	12	0
20	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	12

$$Accuracy = 93{,}77\%$$

Pearson correlation classifier *Confusion Matrix*

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
0	13	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	12	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
2	0	0	12	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
3	0	0	0	13	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	13	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5	0	0	0	0	0	13	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	13	0	0	0	0	0	0	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	13	0	0	0	0	0	0	0	0	0	0	0	0	0
8	0	0	0	0	0	0	0	3	10	0	0	0	0	0	0	0	0	0	0	0	0
9	0	0	0	0	0	0	0	0	0	12	1	0	0	0	0	0	0	0	0	0	0
10	0	0	0	0	0	0	0	0	0	0	13	0	0	0	0	0	0	0	0	0	0
11	0	0	0	0	0	0	0	0	0	0	0	12	1	0	0	0	0	0	0	0	0
12	0	0	0	0	2	0	0	0	0	0	0	0	11	0	0	0	0	0	0	0	0
13	0	0	0	0	0	0	0	0	0	0	0	0	0	13	0	0	0	0	0	0	0
14	0	0	0	0	0	0	0	0	0	0	0	0	0	0	13	0	0	0	0	0	0
15	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	12	0	0	0	0	0
16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	13	0	0	0	0
17	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	13	0	0	0
18	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	13	0	0
19	0	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	11	0
20	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	12

$$Accuracy = 95{,}23\%$$

Test 1 – Results and First Considerations

The best classifier is ***Bray-Curtis*** with an **accuracy** of **96.70%**. Then *Tanimoto, Dice, Kulczynski 1, Nclassifier* and *Chi-square* with 95.60%. Then *Cosine Similarity, Braun-Blanquet, Otsuka, Euclidean Distance, Manhattan Distance, Hellinger-square* and *Pearson correlation* with 95.23%.

In general, we see most of the classifiers recognize more than 90% of actions, except for the *Ranking* and *Entropy*. A special case is the *Mountford* classifier with 20.14% , then *Kulczynski 2* and *Simpson* that reach zero accuracy. Most classifiers confuse sometimes 180˚ and 90˚ turns, but even if not correctly classified these errors fall in an area "semantically close" with respect to the Ground Truth.

We have to note that in the FFxIVFF space thi classifiers are very good to distinguish very similar actions like "going upstairs", "going downstairs" and "walking" performed by physically different people. We have also to consider that the presence of a relatively great number of types of actions in the Dictionary, usually tend to “confuse” the classifiers lowering the general accuracy, nonetheless the obtained accuracies are very high.

Test 2 - Majority Voting Combinations of All Classifiers with All the Sensors

The Majority Voting Combinations has been done usign the following classifiers:

- *Ranking*
- *Cosine Similarity*
- *Tanimoto*
- *Braun-blanquet*
- *Kulczynski 1*
- *Dice*
- *Otsuka*
- *NClassifier*
- *NWeightClassifier*
- *Entropy*
- *Distanza Euclidea*
- *Manhattan*
- *Camberra*
- *Chi-Quadro*
- *Hellinger-Quadro*
- *Hamming*
- *Bray-Curtis*
- *Min variance*
- *Sum variance*
- *Pearson correlation*

Best results use the following combinations:

1) *Cosine Similarity, Manhattan, Dice,Camberra* e *Bray-Curtis*, preference for *Cosine Similarity*
2) *Cosine Similarity, Manhattan, Braun-Blanquet,Camberra* e *Bray-Curtis* , preferenza *Cosine Similarity*
3) *Cosine Similarity, Manhattan,Kulczynski 1,Camberra* e *Bray-Curtis*, preferenza *Cosine Similarity*

With the following *Majority Confusion Matrix:*

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
0	13	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	13	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	0	0	13	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3	0	0	0	13	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	13	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5	0	0	0	0	0	13	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	13	0	0	0	0	0	0	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	13	0	0	0	0	0	0	0	0	0	0	0	0	0
8	0	0	0	0	0	0	0	2	10	0	1	0	0	0	0	0	0	0	0	0	0
9	0	0	0	0	0	0	0	0	0	12	1	0	0	0	0	0	0	0	0	0	0
10	0	0	0	0	0	0	0	0	0	0	13	0	0	0	0	0	0	0	0	0	0
11	0	0	0	0	0	0	0	0	0	0	0	12	1	0	0	0	0	0	0	0	0
12	0	0	0	0	0	0	0	0	0	0	0	0	13	0	0	0	0	0	0	0	0
13	0	0	0	0	0	0	0	0	0	0	0	0	0	13	0	0	0	0	0	0	0
14	0	0	0	0	0	0	0	0	0	0	0	0	0	0	13	0	0	0	0	0	0
15	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	12	0	0	0	0	0
16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	13	0	0	0	0
17	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	13	0	0	0
18	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	13	0	0
19	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	13	0
20	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	12

$$Accuracy = 97{,}43\%$$

Then:

1) *Camberra*, *Bray-Curtis* e *Nclassifier*, preference *Camberra*
2) *Tanimoto*, *Bray-Curtis* e *Nclassifier*, preference *Tanimoto*
3) *Dice*, *Bray-Curtis* e *Nclassifier*, preference *Dice*

Here the *Majority Confusion Matrix* of the case 1)*:*

	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**19**	**20**
0	13	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	13	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	0	0	13	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3	0	0	0	13	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	13	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5	0	0	0	0	0	13	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	13	0	0	0	0	0	0	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	13	0	0	0	0	0	0	0	0	0	0	0	0	0
8	0	0	0	0	0	0	0	3	9	0	1	0	0	0	0	0	0	0	0	0	0
9	0	0	0	0	0	0	0	0	0	12	1	0	0	0	0	0	0	0	0	0	0
10	0	0	0	0	0	0	0	0	0	0	13	0	0	0	0	0	0	0	0	0	0
11	0	0	0	0	0	0	0	0	0	0	0	12	1	0	0	0	0	0	0	0	0
12	0	0	0	0	0	0	0	0	0	0	0	0	13	0	0	0	0	0	0	0	0
13	0	0	0	0	0	0	0	0	0	0	0	0	0	13	0	0	0	0	0	0	0
14	0	0	0	0	0	0	0	0	0	0	0	0	0	0	13	0	0	0	0	0	0
15	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	12	0	0	0	0	0
16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	13	0	0	0	0
17	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	13	0	0	0
18	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	13	0	0
19	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	13	0
20	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	12

$$Accuracy = 97{,}04\%$$

Here the *Majority Confusion Matrix* of the cases 2) and 3)*:*

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
0	13	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	13	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	0	0	13	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3	0	0	0	13	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	13	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5	0	0	0	0	0	13	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	13	0	0	0	0	0	0	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	13	0	0	0	0	0	0	0	0	0	0	0	0	0
8	0	0	0	0	0	0	0	2	10	0	1	0	0	0	0	0	0	0	0	0	0
9	0	0	0	0	0	0	0	0	0	12	1	0	0	0	0	0	0	0	0	0	0
10	0	0	0	0	0	0	0	0	0	0	13	0	0	0	0	0	0	0	0	0	0
11	0	0	0	0	0	0	0	0	0	0	0	12	1	0	0	0	0	0	0	0	0
12	0	0	0	0	0	0	1	0	0	0	0	0	12	0	0	0	0	0	0	0	0
13	0	0	0	0	0	0	0	0	0	0	0	0	0	13	0	0	0	0	0	0	0
14	0	0	0	0	0	0	0	0	0	0	0	0	0	0	13	0	0	0	0	0	0
15	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	12	0	0	0	0	0
16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	13	0	0	0	0
17	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	13	0	0	0
18	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	13	0	0
19	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	13	0
20	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	12

$$Accuracy = 97{,}04\%$$

From the results, we note that using combinations the accuracy **increases** of about one percent compared to the best single classifier results. Clearly, we must find the right compromise between accuracy performance and computational costs.

*The following tests show the results obtained using the best classifier (**Bray-Curtis**), varying the number and positioning of sensors on the body.*

Test 3 –LOOCV hip, right wrist, right ankle

Bray-Curtis classifier *Confusion Matrix:*

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
0	13	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	11	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	0
2	0	0	13	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3	0	0	0	13	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	13	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5	0	0	0	0	0	13	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	13	0	0	0	0	0	0	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	13	0	0	0	0	0	0	0	0	0	0	0	0	0
8	0	0	0	0	0	0	0	4	8	0	1	0	0	0	0	0	0	0	0	0	0
9	0	0	0	0	0	0	0	0	0	12	1	0	0	0	0	0	0	0	0	0	0
10	0	0	0	0	0	0	0	0	1	0	12	0	0	0	0	0	0	0	0	0	0
11	0	0	0	0	0	0	0	0	0	0	0	12	1	0	0	0	0	0	0	0	0
12	0	0	0	0	0	0	0	0	0	0	0	2	11	0	0	0	0	0	0	0	0
13	0	0	0	0	0	0	0	0	0	0	0	0	0	13	0	0	0	0	0	0	0
14	0	0	0	0	0	0	0	0	0	0	0	0	0	0	13	0	0	0	0	0	0
15	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	12	0	0	0	0	0
16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	13	0	0	0	0
17	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	13	0	0	0
18	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	13	0	0
19	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	13	0
20	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	12

$$Accuracy = 94{,}87\%$$

From the results we see that with this configuration of sensors (three sensors), the accuracy *decreases* by about 2.5 percent points compared to the best results obtained using Bray-Curtis with all the sensors. In this case, the Bray- Curtis classifier tends to make more mistakes in recognizing actions as “rise from a chair”, “turn right 90°”, “left turn 180°”, “right turn 90°” and “karate side kick”. Despite these errors, the performances are still very high.

Test 4 - LOOCV with hip, left wrist, left ankle sensors

Bray-Curtis classifier *Confusion Matrix:*

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
0	13	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	13	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	0	0	13	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3	0	0	0	13	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	13	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5	0	0	0	0	0	13	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	13	0	0	0	0	0	0	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	12	0	1	0	0	0	0	0	0	0	0	0	0	0
8	0	0	0	0	0	0	0	2	9	0	2	0	0	0	0	0	0	0	0	0	0
9	0	0	0	0	0	0	0	0	0	12	1	0	0	0	0	0	0	0	0	0	0
10	0	0	0	0	0	0	0	0	1	0	12	0	0	0	0	0	0	0	0	0	0
11	0	0	0	0	0	0	0	0	0	0	0	10	1	2	0	0	0	0	0	0	0
12	0	0	0	0	0	0	1	0	0	0	0	0	12	0	0	0	0	0	0	0	0
13	0	0	0	0	0	0	0	0	0	0	0	1	1	11	0	0	0	0	0	0	0
14	0	0	0	0	0	0	0	0	0	0	0	0	0	0	13	0	0	0	0	0	0
15	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	12	0	0	0	0	0
16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	13	0	0	0	0
17	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	13	0	0	0
18	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	13	0	0
19	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	12	0
20	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	12

$$Accuracy = 94{,}13\%$$

With this configurations (three sensor) the accuracy decreases by about 2.7 percent compared to the best result obtained with Bray-Curtis and all the sensors. The Bray-Curtis classifier tends to make more mistakes in recognizing the actions "right turn 180°", "karate front kick", "karate punch" and "sitting on a chair"

Test 5 - LOOCV right wrist, right ankle

Bray-Curtis classifier *Confusion Matrix:*

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
0	13	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	1	10	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1
2	0	0	13	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3	0	0	0	13	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	10	0	0	0	0	0	0	0	2	0	0	1	0	0	0	0	0
5	0	0	0	0	0	13	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	13	0	0	0	0	0	0	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	13	0	0	0	0	0	0	0	0	0	0	0	0	0
8	0	0	0	0	0	0	0	4	8	0	1	0	0	0	0	0	0	0	0	0	0
9	0	0	0	0	0	0	0	0	0	12	1	0	0	0	0	0	0	0	0	0	0
10	0	0	0	0	0	0	0	0	0	1	12	0	0	0	0	0	0	0	0	0	0
11	0	0	0	0	0	0	0	0	0	0	0	12	1	0	0	0	0	0	0	0	0
12	0	0	0	0	1	0	0	0	0	0	0	4	8	0	0	0	0	0	0	0	0
13	0	0	0	0	0	0	0	0	0	0	0	0	0	13	0	0	0	0	0	0	0
14	0	0	0	0	0	0	0	0	0	0	0	0	0	0	13	0	0	0	0	0	0
15	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	12	0	0	0	0	0
16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	13	0	0	0	0
17	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	13	0	0	0
18	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	13	0	0
19	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	13	0
20	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	1	12

$$Accuracy = 91{,}94\%$$

With this configurations (three sensor) the accuracy decrease by about 5.5 percent points compared to the best result obtained with Bray-Curtis and all the sensors. The Bray-Curtis classifier tends to make more mistakes in recognizing actions "rising from the chair", "fall", "karate side kick", and “heavy sitting"

Test 6 - LOOCV left wirst, left ankle

Bray-Curtis classifier *Confusion Matrix:*

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
0	13	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	10	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	1
2	0	0	12	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
3	0	0	0	13	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	11	0	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5	0	0	0	0	0	13	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	13	0	0	0	0	0	0	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	12	1	0	0	0	0	0	0	0	0	0	0	0	0
8	0	0	0	0	0	0	0	2	9	0	2	0	0	0	0	0	0	0	0	0	0
9	0	0	0	0	0	0	0	0	1	12	0	0	0	0	0	0	0	0	0	0	0
10	0	1	0	0	0	0	0	0	0	1	11	0	0	0	0	0	0	0	0	0	0
11	0	2	0	1	0	0	0	0	0	0	0	7	0	3	0	0	0	0	0	0	0
12	0	0	0	0	1	0	1	0	0	0	0	0	10	0	0	1	0	0	0	0	0
13	0	0	0	0	1	0	0	0	0	0	0	1	0	10	0	0	0	0	0	0	1
14	0	0	0	0	0	0	0	0	0	0	0	0	0	0	13	0	0	0	0	0	0
15	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	12	0	0	0	0	0
16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	13	0	0	0	0
17	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	13	0	0	0
18	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	13	0	0
19	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	13	0
20	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	2	10

$$Accuracy = 89{,}01\%$$

With this configuration (two sensor) the performance decreases by about 7.5 percent points compared to the best result obtained with Bray-Curtis and all the sensors. The classifier tends to make more mistakes in recognizing actions "turn left 90°", "karate front kick," "karate side kick", and "karate punch."

Test 7 - LOOCV hip, right wrist, left ankle

Bray-Curtis classifier *Confusion Matrix:*

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
0	13	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	13	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	0	0	13	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3	0	0	0	13	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	13	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5	0	0	0	0	0	13	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	13	0	0	0	0	0	0	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	13	0	0	0	0	0	0	0	0	0	0	0	0	0
8	0	0	0	0	0	0	0	3	8	0	2	0	0	0	0	0	0	0	0	0	0
9	0	0	0	0	0	0	0	0	0	12	1	0	0	0	0	0	0	0	0	0	0
10	0	0	0	0	0	0	0	0	1	0	12	0	0	0	0	0	0	0	0	0	0
11	0	0	0	0	0	0	0	0	0	0	0	11	1	0	0	0	0	0	0	0	1
12	0	0	0	0	2	0	1	0	0	0	0	0	10	0	0	0	0	0	0	0	0
13	0	0	0	0	0	0	0	0	0	0	0	0	0	13	0	0	0	0	0	0	0
14	0	0	0	0	0	0	0	0	0	0	0	0	0	0	13	0	0	0	0	0	0
15	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	12	0	0	0	0	0
16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	13	0	0	0	0
17	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	13	0	0	0
18	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	13	0	0
19	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	13	0
20	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	12

$$Accuracy = 94{,}87\%$$

With this configuration (three sensors) accuracy decreases by about 2.8 percentage points than the best results with Bray-Curtis using all the sensors. The classifier tends to make more mistakes in recognizing the action "turn left 90 °", "karate front kick" and "karate side kick."

Test 8 - LOOCV hip, left wrist, right ankle

Bray-Curtis classifier *Confusion Matrix:*

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
0	13	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	11	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	0
2	0	0	13	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3	0	0	0	13	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	13	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5	0	0	0	0	0	13	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	13	0	0	0	0	0	0	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	13	0	0	0	0	0	0	0	0	0	0	0	0	0
8	0	0	0	0	0	0	0	4	9	0	0	0	0	0	0	0	0	0	0	0	0
9	0	0	0	0	0	0	0	0	0	13	0	0	0	0	0	0	0	0	0	0	0
10	0	0	0	0	0	0	0	0	0	1	12	0	0	0	0	0	0	0	0	0	0
11	0	0	0	0	0	0	0	0	0	0	0	12	1	0	0	0	0	0	0	0	0
12	0	0	0	0	0	0	0	0	0	0	0	0	13	0	0	0	0	0	0	0	0
13	0	0	0	0	0	0	0	0	0	0	0	0	0	13	0	0	0	0	0	0	0
14	0	0	0	0	0	0	0	0	0	0	0	0	0	0	13	0	0	0	0	0	0
15	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	12	0	0	0	0	0
16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	13	0	0	0	0
17	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	13	0	0	0
18	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	13	0	0
19	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	13	0
20	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	12

$$Accuracy = 96{,}33\%$$

With this configuration (three sensors) the performance decreases by about 0.4 percentage points compared to the best results obtained with Bray-Curtis using all sensors. *With this configuration, the accuracy is very close to the results obtained using all the sensors.*

Test 9 - LOOCV right wrist, left ankle

Bray-Curtis classifier *Confusion Matrix:*

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
0	13	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	11	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1
2	0	0	12	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
3	0	0	0	13	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	11	0	1	0	0	0	0	0	0	0	0	1	0	0	0	0	0
5	0	0	0	0	0	13	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	13	0	0	0	0	0	0	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	13	0	0	0	0	0	0	0	0	0	0	0	0	0
8	0	0	0	0	0	0	0	3	7	1	2	0	0	0	0	0	0	0	0	0	0
9	0	0	0	0	0	0	0	0	0	12	1	0	0	0	0	0	0	0	0	0	0
10	0	0	0	0	0	0	0	0	0	1	12	0	0	0	0	0	0	0	0	0	0
11	0	1	0	0	0	0	0	0	0	0	0	9	2	0	0	0	0	0	0	0	1
12	0	0	0	0	2	0	1	0	0	0	0	0	10	0	0	0	0	0	0	0	0
13	0	0	0	0	0	0	0	0	0	0	0	0	0	13	0	0	0	0	0	0	0
14	0	0	0	0	0	0	0	0	0	0	0	0	0	0	13	0	0	0	0	0	0
15	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	12	0	0	0	0	0
16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	13	0	0	0	0
17	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	13	0	0	0
18	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	13	0	0
19	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	13	0
20	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	2	10

$$Accuracy = 91{,}20\%$$

With this configuration (two sensors) accuracy decreases by about 5.3 percent points compared to the best result obtained with Bray-Curtis using all sensors.The classifier tends to make more mistakes in recognizing "rise from the chair", "turn right 90°", "karate front kick," "karate side kick" and " heavily sitting on a chair."

Test 10 - LOOCV right wrist, left ankle.

Bray-Curtis classifier *Confusion Matrix:*

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
0	13	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	1	10	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1
2	0	0	13	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3	0	0	0	13	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	11	0	0	0	0	0	0	0	2	0	0	0	0	0	0	0	0
5	0	0	0	0	0	13	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	13	0	0	0	0	0	0	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	12	1	0	0	0	0	0	0	0	0	0	0	0	0
8	0	0	0	0	0	0	0	3	8	0	2	0	0	0	0	0	0	0	0	0	0
9	0	0	0	0	0	0	0	0	0	13	0	0	0	0	0	0	0	0	0	0	0
10	0	0	0	0	0	0	0	0	0	2	11	0	0	0	0	0	0	0	0	0	0
11	0	0	0	0	0	0	0	0	0	0	0	12	1	0	0	0	0	0	0	0	0
12	0	0	0	0	0	0	0	0	0	0	0	0	13	0	0	0	0	0	0	0	0
13	0	0	0	0	0	0	0	0	0	0	0	0	0	12	0	0	0	0	0	0	1
14	0	0	0	0	0	0	0	0	0	0	0	0	0	0	13	0	0	0	0	0	0
15	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	12	0	0	0	0	0
16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	13	0	0	0	0
17	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	13	0	0	0
18	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	13	0	0
19	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	13	0
20	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	1	11

$$Accuracy = 93{,}40\%$$

With this configuration (two sensors) the accuracy decreases by about 3.3 percent points compared to the best result obtained with Bray-Curtis using all sensors. The Bray Curtis classifier tends to make more mistakes in recognizing the actions "rise from the chair", "turn right 180°", "turn right 90 °", "turn left 90°" and "heavily sitting on a chair".

Test 11 - LOOCV hip

Bray-Curtis classifier *Confusion Matrix:*

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
0	13	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	1	8	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	4	0
2	0	0	13	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3	0	0	1	12	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	13	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5	0	0	0	0	1	10	0	0	0	0	0	0	0	0	2	0	0	0	0	0	0
6	0	0	0	0	0	0	12	0	0	0	0	0	0	0	0	1	0	0	0	0	0
7	0	0	0	0	0	0	0	9	2	2	0	0	0	0	0	0	0	0	0	0	0
8	0	0	0	0	0	0	0	4	8	0	1	0	0	0	0	0	0	0	0	0	0
9	0	0	0	0	0	0	0	2	0	10	1	0	0	0	0	0	0	0	0	0	0
10	0	0	0	0	0	0	0	0	2	1	10	0	0	0	0	0	0	0	0	0	0
11	0	0	0	0	1	0	1	0	0	0	0	8	2	1	0	0	0	0	0	0	0
12	0	0	0	0	2	0	0	0	0	0	0	1	8	2	0	0	0	0	0	0	0
13	0	0	0	0	1	0	1	0	0	0	0	2	1	8	0	0	0	0	0	0	0
14	0	0	0	0	0	0	0	0	0	0	0	0	0	0	10	0	3	0	0	0	0
15	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	13	0	0	0	0	0
16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	3	0	10	0	0	0	0
17	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	13	0	0	0
18	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	13	0	0
19	0	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	11	0
20	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	10

$$Accuracy = 81{,}31\%$$

With this configuration (one sensor) the accuracy decreases by about 15 percent points compared to the best result obtained with Bray-Curtis using all sensors. The Bray Curtis classifier here tends to make more mistakes in almost every action, however, considering the dimensions of the dictionary of actions and that the test was run with only the sensor on the pelvis, the accuracies are still good.

Test 12 - LOOCV right wrist

Bray-Curtis classifier *Confusion Matrix:*

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
0	12	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	1	10	0	0	0	0	0	0	1	0	0	0	1	0	0	0	0	0	0	0	0
2	0	0	13	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3	0	0	0	13	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	10	0	0	0	0	0	0	0	0	2	0	0	0	0	0	0	1
5	0	0	0	0	0	10	0	0	0	0	1	0	0	0	0	0	2	0	0	0	0
6	0	0	0	0	0	0	12	0	0	0	0	0	0	1	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	12	0	1	0	0	0	0	0	0	0	0	0	0	0
8	0	0	0	0	0	0	0	2	8	0	3	0	0	0	0	0	0	0	0	0	0
9	0	0	0	0	0	0	0	1	0	11	1	0	0	0	0	0	0	0	0	0	0
10	0	0	0	0	0	0	0	0	1	1	11	0	0	0	0	0	0	0	0	0	0
11	0	0	0	0	0	0	0	0	0	0	0	9	3	0	0	0	0	1	0	0	0
12	0	0	1	0	3	0	0	1	0	0	0	3	4	1	0	0	0	0	0	0	0
13	0	0	0	0	0	0	0	0	0	0	0	0	0	13	0	0	0	0	0	0	0
14	0	0	0	0	0	0	0	0	0	0	0	0	0	0	13	0	0	0	0	0	0
15	0	0	0	0	0	0	1	0	0	0	0	1	0	4	0	7	0	0	0	0	0
16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	3	0	13	0	0	0	0
17	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	13	0	0	0
18	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	13	0	0
19	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	1	11	0
20	0	0	0	0	1	0	0	0	0	0	0	1	1	1	0	0	0	0	0	1	8

$$Accuracy = 82{,}78\%$$

With this configuration (one sensor) the accuracy decreases by about 14 percent points compared to the best result obtained with Bray-Curtis using all sensors. The Bray Curtis classifier here tends to make more mistakes in almost every action, however, considering the dimensions of the dictionary of actions and that the test was run with only the sensor on the wrist, the accuracies are still interesting.

Test 13 - LOOCV right ankle

Bray-Curtis classifier *Confusion Matrix:*

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
0	13	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	6	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	5	1
2	0	1	7	0	0	0	0	0	0	0	0	0	0	1	0	0	0	3	0	1	0
3	0	0	0	11	0	0	0	0	0	0	0	0	0	0	0	0	0	2	0	0	0
4	0	0	0	0	9	0	0	0	0	0	0	2	1	0	0	1	0	0	0	0	0
5	0	0	0	0	0	13	0	0	0	0	0	0	0	0	0	0	2	0	0	0	0
6	0	0	0	0	0	0	13	0	0	0	0	0	0	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	11	2	0	0	0	0	0	0	0	0	0	0	0	0
8	0	0	0	0	0	0	0	3	8	0	2	0	0	0	0	0	0	0	0	0	0
9	0	0	0	0	0	0	0	1	0	11	1	0	0	0	0	0	0	0	0	0	0
10	0	0	0	0	0	0	0	0	2	2	9	0	0	0	0	0	0	0	0	0	0
11	0	0	0	0	0	0	0	0	0	0	0	11	2	0	0	0	0	0	0	0	0
12	0	0	0	0	0	0	0	0	0	0	0	3	10	0	0	0	0	0	0	0	0
13	0	0	0	0	0	0	0	0	0	0	0	0	0	11	0	0	0	0	0	0	2
14	0	0	0	0	0	0	0	0	0	0	0	0	0	0	13	0	0	0	0	0	0
15	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	12	0	0	0	0	0
16	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	12	0	0	0	0
17	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	13	0	0	0
18	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	13	0	0
19	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	12	0
20	0	4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	9

$$Accuracy = 83{,}15\%$$

With this configuration (one sensor) the accuracy decrease by about 14 percentage points than the best result obtained using all sensors. The Bray Curtis classifier here tends to make more mistakes in almost every action, however, considering the dimensions of the dictionary of actions and that the test was run with only one sensor, the accuracies are still good.

NIDA Tests - First Considerations

Given the series of tests done using the NIDA dataset we can say that:

1. The accuracy is very high on several single classifiers using all the sensors: e.g. the *Bray-Curtis* with 96.70% (see Test 1)

2. The *Majority Voting Combination* with the best accuracies (e.g. Cosine, Manhattan, Dice, Camberra, with preference Cosine) enhance the results of about 1 percent point (97.43%) compared to Bray-Curtis (see Test2).

3. The results obtained with *Bray-Curtis* classifier using fewer sensors are really interesting because in fact, despite the size of the dictionary of actions, the performance of recognition remain high. For example, using *only three* sensors positioned on the right side of the body, the accuracy is high (94.87%) .

Test Results WARD

The publicly availabe Database WARD1.0 contains 1300 actions (see [5]). The tests performed on the WARD 1.0 database are as follows. At first all the described classifier has been taken and tested one at the time (Test 1) and then they are tested combing them (Test 2):

- Test 1 – LOOCV all classifiers with all the sensors
- Test 2 - Majority Voting Combinations of all classifiers with all the sensors

Then the single classifier with the best accuracy, the Braun-Blanquet, has been taken and its accuracy varying the number and position of sensors on the body has been tested:

- Test 3 - LOOCV with hip, right wrist, right ankle sensors
- Test 4 - LOOCV hip, left wrist, left ankle sensors
- Test 5 - LOOCV left wrist, keft ankle sensors
- Test 6 - LOOCV right wrist, right ankle sensors
- Test 7 - LOOCV hip, right wrist, left ankle sensors
- Test 8 - LOOCV hip, left wrist, right ankle sensors
- Test 9 - LOOCV right wrist, left ankle sensors
- Test 10 - LOOCV left wrist, right ankle sensors
- Test 11 - LOOCV hip sensor
- Test 12 - LOOCV right wrist sensor
- Test 13 - LOOCV right ankle sensor

The *legenda* to identify the actions is as follows:

Action	**Identifier**
Rest at standing	0
Rest at sitting	1
Rest at lying	2
Walk forward	3
Walk forward left-circle	4
Walk forward right-circle	5
Turn left	6
Turn right	7
Go upstairs	8
Go downstairs	9
Jog	10
Jump	11
Push wheelchair	12

Test 1 – LOOCV All Classifiers with All the Sensors

Ranking classifier *Confusion Matrix:*

	0	1	2	3	4	5	6	7	8	9	10	11	12
0	99	0	0	0	0	0	0	0	0	0	0	0	0
1	0	98	0	0	0	0	0	0	0	0	0	0	0
2	0	0	99	0	0	0	0	0	0	0	0	0	0
3	0	0	0	96	0	0	0	0	0	0	1	2	0
4	0	0	0	0	99	0	0	0	0	0	0	0	0
5	0	0	0	0	0	98	0	0	0	0	0	0	0
6	0	0	0	0	0	0	97	0	0	0	0	1	0
7	0	0	0	0	0	0	0	99	0	0	0	0	0
8	0	0	0	0	0	0	0	0	85	0	0	11	0
9	0	0	0	0	0	0	0	0	0	79	2	13	0
10	0	0	0	0	0	0	0	0	0	0	95	0	0
11	0	0	0	0	0	0	0	0	0	0	0	98	0
12	0	1	0	0	0	0	0	0	0	0	0	0	97

$$Accuracy = 97{,}55\%$$

Cosine Similarity classifier *Confusion Matrix:*

	0	1	2	3	4	5	6	7	8	9	10	11	12
0	99	0	0	0	0	0	0	0	0	0	0	0	0
1	0	98	0	0	0	0	0	0	0	0	0	0	0
2	0	0	99	0	0	0	0	0	0	0	0	0	0
3	0	0	0	96	0	0	0	0	0	0	1	2	0
4	0	0	0	0	99	0	0	0	0	0	0	0	0
5	0	0	0	0	0	97	0	0	0	0	0	1	0
6	0	0	0	0	0	0	96	0	0	0	0	2	0
7	0	0	0	0	0	0	0	99	0	0	0	0	0
8	0	0	0	0	0	0	0	0	85	0	0	11	0
9	0	0	0	0	0	0	0	0	0	81	0	13	0
10	0	0	0	0	0	0	0	0	0	0	95	0	0
11	0	0	0	0	0	0	0	0	0	0	0	98	0
12	0	0	0	0	0	0	0	0	0	0	0	0	98

$$Accuracy = 97{,}63\%$$

Tanimoto classifier *Confusion Matrix:*

	0	1	2	3	4	5	6	7	8	9	10	11	12
0	99	0	0	0	0	0	0	0	0	0	0	0	0
1	0	98	0	0	0	0	0	0	0	0	0	0	0
2	0	0	99	0	0	0	0	0	0	0	0	0	0
3	0	0	0	96	0	0	0	0	0	0	0	3	0
4	0	0	0	0	99	0	0	0	0	0	0	0	0
5	0	0	0	0	0	98	0	0	0	0	0	0	0
6	0	0	0	0	0	0	98	0	0	0	0	0	0
7	0	0	0	0	0	0	0	99	0	0	0	0	0
8	0	0	0	0	0	0	0	0	87	0	0	9	0
9	0	0	0	0	0	0	0	0	0	86	0	8	0
10	0	0	0	0	0	0	0	0	0	0	95	0	0
11	0	0	0	0	0	0	0	0	0	0	0	98	0
12	0	0	0	0	0	0	0	0	0	0	0	0	98

$$Accuracy = 98{,}42\%$$

Simpson classifier *Confusion Matrix:*

	0	1	2	3	4	5	6	7	8	9	10	11	12
0	0	0	0	0	0	0	88	11	0	0	0	0	0
1	0	0	0	0	0	0	78	20	0	0	0	0	0
2	0	0	0	0	0	1	2	0	0	0	10	86	0
3	0	0	0	0	0	0	0	0	71	27	1	0	0
4	0	0	0	0	0	2	0	1	0	2	83	1	10
5	0	0	0	0	0	0	0	0	36	17	0	0	45
6	0	0	0	0	0	0	0	1	3	0	80	14	0
7	0	0	0	0	0	0	4	0	66	0	0	0	29
8	0	0	1	0	0	10	0	0	0	0	34	51	0
9	0	0	0	2	0	10	0	0	0	0	77	4	1
10	0	0	0	0	0	0	0	0	14	81	0	0	0
11	0	0	0	0	0	0	0	0	24	74	0	0	0
12	0	0	0	0	0	42	1	8	0	1	46	0	0

$$Accuracy = 0\%$$

BraunBlanquet classifier *Confusion Matrix:*

	0	1	2	3	4	5	6	7	8	9	10	11	12
0	99	0	0	0	0	0	0	0	0	0	0	0	0
1	0	98	0	0	0	0	0	0	0	0	0	0	0
2	0	0	99	0	0	0	0	0	0	0	0	0	0
3	0	0	0	96	0	0	0	0	0	0	0	3	0
4	0	0	0	0	99	0	0	0	0	0	0	0	0
5	0	0	0	0	0	98	0	0	0	0	0	0	0
6	0	0	0	0	0	0	98	0	0	0	0	0	0
7	0	0	0	0	0	0	0	99	0	0	0	0	0
8	0	0	0	0	0	0	0	0	87	0	0	9	0
9	0	0	0	0	0	0	0	0	0	86	0	8	0
10	0	0	0	0	0	0	0	0	0	0	95	0	0
11	0	0	0	0	0	0	0	0	0	0	0	98	0
12	0	0	0	0	0	0	0	0	0	0	0	0	98

$$Accuracy = 98{,}42\%$$

Kulczynski 1 classifier *Confusion Matrix:*

	0	1	2	3	4	5	6	7	8	9	10	11	12
0	99	0	0	0	0	0	0	0	0	0	0	0	0
1	0	98	0	0	0	0	0	0	0	0	0	0	0
2	0	0	99	0	0	0	0	0	0	0	0	0	0
3	0	0	0	96	0	0	0	0	0	0	0	3	0
4	0	0	0	0	99	0	0	0	0	0	0	0	0
5	0	0	0	0	0	98	0	0	0	0	0	0	0
6	0	0	0	0	0	0	98	0	0	0	0	0	0
7	0	0	0	0	0	0	0	99	0	0	0	0	0
8	0	0	0	0	0	0	0	0	87	0	0	9	0
9	0	0	0	0	0	0	0	0	0	86	0	8	0
10	0	0	0	0	0	0	0	0	0	0	95	0	0
11	0	0	0	0	0	0	0	0	0	0	0	98	0
12	0	0	0	0	0	0	0	0	0	0	0	0	98

$$Accuracy = 98{,}42\%$$

Kulczynski 2 classifier *Confusion Matrix:*

	0	1	2	3	4	5	6	7	8	9	10	11	12
0	2	0	0	0	0	0	88	9	0	0	0	0	0
1	0	0	0	0	0	0	78	20	0	0	0	0	0
2	0	0	0	0	0	1	2	0	0	0	10	86	0
3	0	0	0	0	0	0	0	0	79	20	0	0	0
4	0	0	0	0	1	3	0	1	0	4	78	1	11
5	0	0	0	0	0	0	0	0	45	20	0	0	33
6	0	0	0	0	0	0	38	2	3	0	46	9	0
7	0	0	0	0	0	0	5	9	60	1	0	0	24
8	0	0	1	0	0	7	0	0	0	0	27	61	0
9	0	0	0	5	0	6	0	0	0	0	74	9	0
10	0	0	0	0	0	0	0	0	12	83	0	0	0
11	0	0	0	0	0	0	0	0	23	72	0	3	0
12	0	0	0	0	0	40	1	8	0	2	47	0	0

$$Accuracy = 4{,}17\%$$

Dice classifier *Confusion Matrix:*

	0	1	2	3	4	5	6	7	8	9	10	11	12
0	99	0	0	0	0	0	0	0	0	0	0	0	0
1	0	98	0	0	0	0	0	0	0	0	0	0	0
2	0	0	99	0	0	0	0	0	0	0	0	0	0
3	0	0	0	96	0	0	0	0	0	0	0	3	0
4	0	0	0	0	99	0	0	0	0	0	0	0	0
5	0	0	0	0	0	98	0	0	0	0	0	0	0
6	0	0	0	0	0	0	98	0	0	0	0	0	0
7	0	0	0	0	0	0	0	99	0	0	0	0	0
8	0	0	0	0	0	0	0	0	87	0	0	9	0
9	0	0	0	0	0	0	0	0	0	86	0	8	0
10	0	0	0	0	0	0	0	0	0	0	95	0	0
11	0	0	0	0	0	0	0	0	0	0	0	98	0
12	0	0	0	0	0	0	0	0	0	0	0	0	98

$$Accuracy = 98{,}42\%$$

Otsuka classifier *Confusion Matrix:*

	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**
0	99	0	0	0	0	0	0	0	0	0	0	0	0
1	0	98	0	0	0	0	0	0	0	0	0	0	0
2	0	0	99	0	0	0	0	0	0	0	0	0	0
3	0	0	0	96	0	0	0	0	0	0	1	2	0
4	0	0	0	0	99	0	0	0	0	0	0	0	0
5	0	0	0	0	0	97	0	0	0	0	0	1	0
6	0	0	0	0	0	0	96	0	0	0	0	2	0
7	0	0	0	0	0	0	0	99	0	0	0	0	0
8	0	0	0	0	0	0	0	0	85	0	0	11	0
9	0	0	0	0	0	0	0	0	0	81	0	13	0
10	0	0	0	0	0	0	0	0	0	0	95	0	0
11	0	0	0	0	0	0	0	0	0	0	0	98	0
12	0	0	0	0	0	0	0	0	0	0	0	0	98

$$Accuracy = 97{,}63\%$$

Mountford classifier *Confusion Matrix:*

	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**
0	97	0	0	0	0	0	1	1	0	0	0	0	0
1	0	78	0	0	0	0	15	2	3	0	0	0	0
2	0	0	0	0	0	1	8	1	17	0	0	72	0
3	0	0	0	24	0	0	0	0	68	7	0	0	0
4	0	0	0	0	56	0	0	2	38	3	0	0	0
5	0	0	0	0	0	1	0	0	92	5	0	0	0
6	0	0	0	0	0	0	74	7	17	0	0	0	0
7	0	0	0	0	1	0	11	39	47	0	0	0	1
8	0	0	0	6	0	0	0	0	62	12	0	13	3
9	0	0	0	5	0	0	0	0	18	54	0	17	0
10	0	0	0	0	0	0	0	0	11	84	0	0	0
11	0	0	0	0	0	0	0	0	36	39	0	23	0
12	0	0	0	0	0	0	1	0	32	1	0	0	64

$$Accuracy = 45{,}03\%$$

Nclassifier classifier *Confusion Matrix:*

	0	1	2	3	4	5	6	7	8	9	10	11	12
0	99	0	0	0	0	0	0	0	0	0	0	0	0
1	0	97	0	0	0	0	0	0	0	0	0	0	1
2	0	0	99	0	0	0	0	0	0	0	0	0	0
3	0	0	0	91	0	0	0	0	7	1	0	0	0
4	0	0	0	0	95	0	0	0	2	2	0	0	0
5	0	0	0	0	0	93	0	2	1	0	0	2	0
6	0	0	0	0	1	0	94	0	2	0	0	1	0
7	0	0	0	0	0	0	0	98	1	0	0	0	0
8	0	0	0	0	0	0	0	0	87	0	0	9	0
9	0	0	0	2	0	0	0	0	5	80	0	7	0
10	0	0	0	0	0	0	0	0	0	3	92	0	0
11	0	0	0	0	0	0	0	0	2	0	0	96	0
12	0	0	0	0	0	0	0	0	2	0	0	0	96

$$Accuracy = 95{,}82\%$$

NWeightclassifier classifier *Confusion Matrix:*

	0	1	2	3	4	5	6	7	8	9	10	11	12
0	99	0	0	0	0	0	0	0	0	0	0	0	0
1	0	97	0	0	0	0	0	0	0	0	0	0	1
2	0	0	98	0	0	0	0	0	1	0	0	0	0
3	0	0	0	87	0	0	0	0	11	1	0	0	0
4	0	0	0	0	93	0	0	0	5	1	0	0	0
5	0	0	0	0	0	85	0	3	8	2	0	0	0
6	0	0	0	0	2	0	93	0	3	0	0	0	0
7	0	0	0	0	0	4	0	93	2	0	0	0	0
8	0	0	0	0	0	0	0	0	88	0	0	8	0
9	0	0	0	0	0	0	0	0	10	81	0	3	0
10	0	0	0	0	0	0	0	0	0	5	90	0	0
11	0	0	0	0	0	0	0	0	5	0	0	93	0
12	0	0	0	0	0	0	0	0	4	0	0	0	94

$$Accuracy = 93{,}77\%$$

Entropy classifier *Confusion Matrix:*

	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**
0	98	0	1	0	0	0	0	0	0	0	0	0	0
1	19	0	79	0	0	0	0	0	0	0	0	0	0
2	0	97	0	0	0	0	0	0	0	0	0	2	0
3	0	0	0	92	0	0	0	0	0	4	1	2	0
4	0	0	0	45	26	1	19	0	3	2	0	3	0
5	0	0	0	57	7	0	0	23	1	6	0	4	0
6	0	0	0	0	33	0	1	60	1	0	0	3	0
7	0	0	0	0	2	19	77	0	0	0	0	1	0
8	0	0	0	14	2	0	3	0	49	11	0	16	1
9	0	0	0	15	0	0	0	0	10	58	5	15	1
10	0	0	0	0	0	0	0	0	0	3	88	4	0
11	0	0	0	0	0	0	0	0	0	1	0	97	0
12	0	0	0	0	0	0	0	0	5	0	0	0	93

$$Accuracy = 47{,}40\%$$

Euclidean Distance classifier *Confusion Matrix:*

	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**
0	99	0	0	0	0	0	0	0	0	0	0	0	0
1	0	97	0	0	0	0	0	0	0	0	0	0	1
2	0	0	99	0	0	0	0	0	0	0	0	0	0
3	0	0	0	91	0	0	0	0	7	1	0	0	0
4	0	0	0	0	95	0	0	0	2	2	0	0	0
5	0	0	0	0	0	92	0	1	3	1	0	1	0
6	0	0	0	0	1	0	94	0	2	0	0	1	0
7	0	0	0	0	0	1	0	97	1	0	0	0	0
8	0	0	0	0	0	0	0	0	87	0	0	9	0
9	0	0	0	0	0	0	0	0	6	81	0	7	0
10	0	0	0	0	0	0	0	0	0	3	92	0	0
11	0	0	0	0	0	0	0	0	2	0	0	96	0
12	0	0	0	0	0	0	0	0	2	0	0	0	96

$$Accuracy = 95{,}74\%$$

Manhattan classifier *Confusion Matrix:*

	0	1	2	3	4	5	6	7	8	9	10	11	12
0	99	0	0	0	0	0	0	0	0	0	0	0	0
1	0	97	0	0	0	0	0	0	0	0	0	0	1
2	0	0	98	0	0	0	0	0	1	0	0	0	0
3	0	0	0	92	0	0	0	0	6	1	0	0	0
4	0	0	0	0	95	0	0	0	2	2	0	0	0
5	0	0	0	0	0	94	0	1	2	0	0	1	0
6	0	0	0	0	1	0	94	0	2	0	0	1	0
7	0	0	0	0	0	0	0	98	1	0	0	0	0
8	0	0	0	0	0	0	0	0	86	0	0	10	0
9	0	0	0	0	0	0	0	0	7	80	0	7	0
10	0	0	0	0	0	0	0	0	0	3	92	0	0
11	0	0	0	0	0	0	0	0	2	0	0	96	0
12	0	0	0	0	0	0	0	0	2	0	0	0	96

$$Accuracy = 95{,}82\%$$

Camberra classifier *Confusion Matrix:*

	0	1	2	3	4	5	6	7	8	9	10	11	12
0	99	0	0	0	0	0	0	0	0	0	0	0	0
1	0	98	0	0	0	0	0	0	0	0	0	0	0
2	0	0	99	0	0	0	0	0	0	0	0	0	0
3	0	0	0	96	0	0	0	0	0	0	1	2	0
4	0	0	0	0	92	0	5	0	1	1	0	0	0
5	0	0	0	0	0	89	0	6	0	0	0	3	0
6	0	0	0	0	0	0	95	0	0	0	0	3	0
7	0	0	0	0	0	1	0	98	0	0	0	0	0
8	0	0	0	0	0	2	0	0	83	0	0	11	0
9	0	0	0	10	0	0	0	0	4	64	2	13	1
10	0	0	0	0	0	0	0	0	0	0	95	0	0
11	0	0	0	0	0	0	0	0	0	0	0	98	0
12	0	0	0	0	0	0	0	0	0	0	0	0	98

$$Accuracy = 94{,}80\%$$

Chi-Square classifier *Confusion Matrix:*

	0	1	2	3	4	5	6	7	8	9	10	11	12
0	99	0	0	0	0	0	0	0	0	0	0	0	0
1	0	97	0	0	0	0	0	0	0	0	0	0	1
2	0	0	98	0	0	0	0	0	1	0	0	0	0
3	0	0	0	91	0	0	0	0	5	3	0	0	0
4	0	0	0	0	95	0	0	0	2	2	0	0	0
5	0	0	0	0	0	94	0	1	2	0	0	1	0
6	0	0	0	0	1	0	94	0	2	0	0	1	0
7	0	0	0	0	0	0	0	98	1	0	0	0	0
8	0	0	0	0	0	0	0	0	85	0	0	11	0
9	0	0	0	0	0	0	0	0	6	81	0	7	0
10	0	0	0	0	0	0	0	0	0	3	92	0	0
11	0	0	0	0	0	0	0	0	2	0	0	96	0
12	0	0	0	0	0	0	0	0	2	0	0	0	96

$$Accuracy = 95{,}74\%$$

Hellinger-Square classifier *Confusion Matrix:*

	0	1	2	3	4	5	6	7	8	9	10	11	12
0	99	0	0	0	0	0	0	0	0	0	0	0	0
1	0	97	0	0	0	0	0	0	0	0	0	0	1
2	0	0	99	0	0	0	0	0	0	0	0	0	0
3	0	0	0	91	0	0	0	0	5	3	0	0	0
4	0	0	0	0	95	0	0	0	2	2	0	0	0
5	0	0	0	0	0	94	0	1	2	0	0	1	0
6	0	0	0	0	1	0	94	0	2	0	0	1	0
7	0	0	0	0	0	0	0	98	1	0	0	0	0
8	0	0	0	0	0	0	0	0	85	0	0	11	0
9	0	0	0	0	0	0	0	0	6	81	0	7	0
10	0	0	0	0	0	0	0	0	0	3	92	0	0
11	0	0	0	0	0	0	0	0	2	0	0	96	0
12	0	0	0	0	0	0	0	0	2	0	0	0	96

$$Accuracy = 95{,}82\%$$

Hamming classifier *Confusion Matrix:*

	0	1	2	3	4	5	6	7	8	9	10	11	12
0	99	0	0	0	0	0	0	0	0	0	0	0	0
1	0	97	0	0	0	0	0	0	0	0	0	0	1
2	0	0	98	0	0	0	0	0	1	0	0	0	0
3	0	0	0	91	0	0	0	0	6	2	0	0	0
4	0	0	0	0	95	0	0	0	3	1	0	0	0
5	0	0	0	0	0	91	0	1	4	2	0	0	0
6	0	0	0	0	1	0	94	0	3	0	0	0	0
7	0	0	0	0	0	4	0	94	1	0	0	0	0
8	0	0	0	0	0	0	0	0	87	0	0	9	0
9	0	0	0	0	0	0	0	0	6	82	0	6	0
10	0	0	0	0	0	0	0	0	0	5	90	0	0
11	0	0	0	0	0	0	0	0	3	0	0	95	0
12	0	0	0	0	0	0	0	0	2	0	0	0	96

$$Accuracy = 95{,}19\%$$

Bray-Curtis classifier *Confusion Matrix:*

	0	1	2	3	4	5	6	7	8	9	10	11	12
0	99	0	0	0	0	0	0	0	0	0	0	0	0
1	0	98	0	0	0	0	0	0	0	0	0	0	0
2	0	0	99	0	0	0	0	0	0	0	0	0	0
3	0	0	0	96	0	0	0	0	0	0	1	2	0
4	0	0	0	0	99	0	0	0	0	0	0	0	0
5	0	0	0	0	0	97	0	0	0	0	0	1	0
6	0	0	0	0	0	0	95	0	0	0	0	3	0
7	0	0	0	0	0	0	0	99	0	0	0	0	0
8	0	0	0	0	0	0	0	0	85	0	0	11	0
9	0	0	0	0	0	0	0	0	1	82	0	11	0
10	0	0	0	0	0	0	0	0	0	0	95	0	0
11	0	0	0	0	0	0	0	0	0	0	0	98	0
12	0	0	0	0	0	0	0	0	0	0	0	0	98

$$Accuracy = 97{,}63\%$$

Min variance classifier *Confusion Matrix:*

	0	1	2	3	4	5	6	7	8	9	10	11	12
0	99	0	0	0	0	0	0	0	0	0	0	0	0
1	0	97	0	0	0	0	0	0	0	0	0	0	1
2	0	0	98	0	0	0	0	0	1	0	0	0	0
3	0	0	0	91	0	0	0	0	7	1	0	0	0
4	0	0	0	0	94	0	0	0	3	2	0	0	0
5	0	0	0	0	0	87	0	3	7	1	0	0	0
6	0	0	0	0	2	0	93	0	3	0	0	0	0
7	0	0	0	0	0	4	0	94	1	0	0	0	0
8	0	0	0	0	0	0	0	0	87	0	0	9	0
9	0	0	0	0	0	0	0	0	8	80	0	6	0
10	0	0	0	0	0	0	0	0	0	5	90	0	0
11	0	0	0	0	0	0	0	0	4	0	0	94	0
12	0	0	0	0	0	0	0	0	2	0	0	0	96

$$Accuracy = 94{,}48\%$$

Pearson correlation classifier *Confusion Matrix:*

	0	1	2	3	4	5	6	7	8	9	10	11	12
0	99	0	0	0	0	0	0	0	0	0	0	0	0
1	0	98	0	0	0	0	0	0	0	0	0	0	0
2	0	0	99	0	0	0	0	0	0	0	0	0	0
3	0	0	0	96	0	0	0	0	0	0	0	3	0
4	0	0	0	0	99	0	0	0	0	0	0	0	0
5	0	0	0	0	0	97	0	0	0	0	0	1	0
6	0	0	0	0	0	0	98	0	0	0	0	0	0
7	0	0	0	0	0	0	0	99	0	0	0	0	0
8	0	0	0	0	0	0	0	0	86	0	0	10	0
9	0	0	0	0	0	0	0	0	0	83	0	11	0
10	0	0	0	0	0	0	0	0	0	0	95	0	0
11	0	0	0	0	0	0	0	0	0	0	0	98	0
12	0	0	0	0	0	0	0	0	0	0	0	0	98

$$Accuracy = 98{,}03\%$$

Test 1 – Results and First Considerations

The classifiers with the best accuracy are ***Braun-Blanquet***, ***Tanimoto***, ***Dice*** and ***Kulczynski 1*** with **98.43% accuracy**. Then ***Bray-Curtis***, *Ranking*, *Cosine Similarity* and *Pearson correlation* with an accuracy ranging from 97.7% to 98%. In general we can say most of the classifiers recognizes more than 95% of the actions.
A special case is the *Mountford* classifier with 64.02%, and Entropy with 47.40%, and finally *Kulczynski 2*, *Simpson* that both reach accuracy zero.
Most of the classifiers commit a few errors in recognizing "Go downstairs" and "Go upstairs," confusing these actions with "Jump."

Test 2 - Majority Voting Combination of All Classifiers with All the Sensors

These series of tests was run with the same classifiers described in Test 2 of “Test Results NIDA, p. 59”. The test shows that combinations *does not* improve the accuracy, because classifiers make mistakes on the same Actions, making fruitless the policy of *Majority Voting*. Generally speaking, the classifiers tend to commit errors in the recognition of the following actions: "go downstairs" and "go upstairs". Accuracy is the same obtained using single classifiers. Hence, ***BraunBlanquet, Tanimoto, Dice, Kulczynski 1*** used as single classifiers give the best performance on the WARD database with an accuracy of **98.42%.**

Test 3 - LOOCV hip, right wrist, right ankle.

BraunBlanquet classifier *Confusion Matrix:*

	0	1	2	3	4	5	6	7	8	9	10	11	12
0	99	0	0	0	0	0	0	0	0	0	0	0	0
1	0	98	0	0	0	0	0	0	0	0	0	0	0
2	0	0	99	0	0	0	0	0	0	0	0	0	0
3	0	0	0	96	0	0	0	0	0	0	1	2	0
4	0	0	0	0	99	0	0	0	0	0	0	0	0
5	0	0	0	0	0	98	0	0	0	0	0	0	0
6	0	0	0	0	0	0	98	0	0	0	0	0	0
7	0	0	0	0	0	0	0	99	0	0	0	0	0
8	0	0	0	0	0	0	0	0	87	0	0	9	0
9	0	0	0	0	0	0	0	0	0	84	1	9	0
10	0	0	0	0	0	0	0	0	0	0	95	0	0
11	0	0	0	0	0	0	0	0	0	0	0	98	0
12	0	0	0	0	0	0	0	0	0	0	0	0	98

$$Accuracy = 98{,}26\%$$

We see that with this configuration of sensors (three sensors), the accuracy remains pratically unchanged compared to the results obtained using the best classifier (*BraunBlanquet*) with all the sensors (98.42%.) In this case, the classifier tends to make more mistakes in recognizing "go upstairs" and "go downstairs."

Test 4- LOOCV hip, left wrist, left ankle

BraunBlanquet classifier *Confusion Matrix:*

	0	1	2	3	4	5	6	7	8	9	10	11	12
0	99	0	0	0	0	0	0	0	0	0	0	0	0
1	0	98	0	0	0	0	0	0	0	0	0	0	0
2	0	0	99	0	0	0	0	0	0	0	0	0	0
3	0	0	0	96	0	0	0	0	0	0	1	2	0
4	0	0	0	0	99	0	0	0	0	0	0	0	0
5	0	0	0	0	0	98	0	0	0	0	0	0	0
6	0	0	0	0	0	0	98	0	0	0	0	0	0
7	0	0	0	0	0	0	0	99	0	0	0	0	0
8	0	0	0	0	0	0	0	0	87	0	0	9	0
9	0	0	0	0	0	0	0	0	0	84	1	9	0
10	0	0	0	0	0	0	0	0	0	0	95	0	0
11	0	0	0	0	0	0	0	0	0	0	0	98	0
12	0	0	0	0	0	0	0	0	0	0	0	0	98

$$Accuracy = 98{,}26\%$$

This configuration of sensors is similar to the former configuration but wrist and ankle sensors are on the left part of the body. Again with this configuration of sensors (three sensors), the accuracy remains practically unchanged compared to the results obtained using the best classifier (*BraunBlanquet*) with all the sensors (98.42%.) In this case, the classifier tends to make more mistakes in recognizing "go upstairs" and "go downstairs."

Test 5- LOOCV right wrist, right ankle

BraunBlanquet classifier *Confusion Matrix:*

	0	1	2	3	4	5	6	7	8	9	10	11	12
0	99	0	0	0	0	0	0	0	0	0	0	0	0
1	0	98	0	0	0	0	0	0	0	0	0	0	0
2	0	0	99	0	0	0	0	0	0	0	0	0	0
3	0	0	0	95	1	0	0	0	0	0	0	3	0
4	0	0	0	2	97	0	0	0	0	0	0	0	0
5	0	0	0	0	0	98	0	0	0	0	0	0	0
6	0	0	0	0	0	0	97	0	0	0	0	1	0
7	0	0	0	0	0	0	0	99	0	0	0	0	0
8	0	0	0	0	0	0	0	0	88	0	0	8	0
9	0	0	0	0	0	0	0	0	8	81	0	5	0
10	0	0	0	0	0	0	0	0	0	0	95	0	0
11	0	0	0	0	0	0	0	0	0	0	0	98	0
12	0	0	0	0	0	0	0	0	0	0	0	0	98

$$Accuracy = 97{,}79\%$$

This configuration of sensors is the same of the Test 3 configuration but without the hip sensor. With this configuration of sensors (2 sensors) accuracy decreases only by about 0.6 percent points than the result obtained using the best classifier (*BraunBlanquet*) wirth all sensors. If we compare the results of this test with Test 3, to eliminate the hip sensor costs only 0.4 percent points in accuracy.

Test 6- LOOCV left wrist, left ankle

BraunBlanquet classifier *Confusion Matrix:*

	0	1	2	3	4	5	6	7	8	9	10	11	12
0	99	0	0	0	0	0	0	0	0	0	0	0	0
1	0	98	0	0	0	0	0	0	0	0	0	0	0
2	0	0	99	0	0	0	0	0	0	0	0	0	0
3	0	0	0	94	0	0	0	0	0	1	0	4	0
4	0	0	0	0	99	0	0	0	0	0	0	0	0
5	0	0	0	0	0	97	0	0	0	0	0	1	0
6	0	0	0	0	0	0	98	0	0	0	0	0	0
7	0	0	0	0	0	0	0	99	0	0	0	0	0
8	0	0	0	0	0	0	0	0	86	0	0	10	0
9	0	0	0	0	0	0	0	0	8	84	0	9	1
10	0	0	0	0	0	0	0	0	0	0	95	0	0
11	0	0	0	0	0	0	0	0	0	0	0	98	0
12	0	0	0	1	0	0	0	0	0	0	0	0	97

$$Accuracy = 97{,}87\%$$

This configuration of sensors here is the same of the Test 4 configuration but without the hip sensor. With this configuration of sensors (2 sensors) accuracy decreases only by about 0.6 percent points compared to the result obtained using the best classifier (*BraunBlanquet*) using all sensors. If we compare the results of this test with Test 3, eliminating the hip sensor costs only 0.4 percent points in accuracy.

Test 7- LOOCV hip, right wrist, left ankle

BraunBlanquet classifier *Confusion Matrix:*

	0	1	2	3	4	5	6	7	8	9	10	11	12
0	99	0	0	0	0	0	0	0	0	0	0	0	0
1	0	98	0	0	0	0	0	0	0	0	0	0	0
2	0	0	99	0	0	0	0	0	0	0	0	0	0
3	0	0	0	96	0	0	0	0	0	0	1	2	0
4	0	0	0	0	99	0	0	0	0	0	0	0	0
5	0	0	0	0	0	98	0	0	0	0	0	0	0
6	0	0	0	0	0	0	98	0	0	0	0	0	0
7	0	0	0	0	0	0	0	99	0	0	0	0	0
8	0	0	0	0	0	0	0	0	88	0	0	8	0
9	0	0	0	0	0	0	0	0	0	83	0	11	0
10	0	0	0	0	0	0	0	0	0	0	95	0	0
11	0	0	0	0	0	0	0	0	0	0	0	98	0
12	0	0	0	0	0	0	0	0	0	0	0	0	98

$$Accuracy = 98{,}26\%$$

Again we see that with this configuration of sensors (three sensors) accuracy remains pratically unchanged compared to the results of *BraunBlanquet* classifier using all the sensors (98.42%.)

The classifier tends to make more mistakes in recognizing "go upstairs" and "go downstairs."

Test 8 - LOOCV hip, left wrist, right ankle

BraunBlanquet classifier *Confusion Matrix:*

	0	1	2	3	4	5	6	7	8	9	10	11	12
0	99	0	0	0	0	0	0	0	0	0	0	0	0
1	0	98	0	0	0	0	0	0	0	0	0	0	0
2	0	0	99	0	0	0	0	0	0	0	0	0	0
3	0	0	0	96	0	0	0	0	0	0	0	3	0
4	0	0	0	0	99	0	0	0	0	0	0	0	0
5	0	0	0	0	0	98	0	0	0	0	0	0	0
6	0	0	0	0	0	0	97	0	0	0	0	1	0
7	0	0	0	0	0	0	0	99	0	0	0	0	0
8	0	0	0	0	0	0	0	0	86	0	1	9	0
9	0	0	0	0	0	0	0	0	12	69	1	12	0
10	0	0	0	0	0	0	0	0	0	0	95	0	0
11	0	0	0	0	0	0	0	0	0	0	0	98	0
12	0	0	0	0	0	0	0	0	0	0	0	0	98

$$Accuracy = 96{,}92\%$$

With this configuration of sensors (two sensors) accuracy decreases by about 1.5 percent points compared to results of the best single classifier (*BraunBlanquet*) using all the sensors. The classifier tends to make more mistakes in recognizing the actions "go upstairs" and "go downstairs."

Test 9 - LOOCV right wrist, left ankle

BraunBlanquet classifier *Confusion Matrix:*

	0	1	2	3	4	5	6	7	8	9	10	11	12
0	99	0	0	0	0	0	0	0	0	0	0	0	0
1	0	98	0	0	0	0	0	0	0	0	0	0	0
2	0	0	99	0	0	0	0	0	0	0	0	0	0
3	0	0	0	96	0	0	0	0	0	0	0	3	0
4	0	0	0	0	99	0	0	0	0	0	0	0	0
5	0	0	0	0	0	98	0	0	0	0	0	0	0
6	0	0	0	0	0	0	97	0	0	0	0	1	0
7	0	0	0	0	0	0	0	99	0	0	0	0	0
8	0	0	0	0	0	0	0	0	86	0	1	9	0
9	0	0	0	0	0	0	0	0	12	69	1	12	0
10	0	0	0	0	0	0	0	0	0	0	95	0	0
11	0	0	0	0	0	0	0	0	0	0	0	98	0
12	0	0	0	0	0	0	0	0	0	0	0	0	98

$$Accuracy = 96{,}92\%$$

With this configuration of sensors (two sensors) accuracy decreases by about 1.5 percent points compared to the results obtained using the best single classifier (*BraunBlanquet*) using all the sensors. The classifier tends to make more mistakes in recognizing the actions "Go upstairs" and "Go downstairs." Recognition gives the same results of Test 8.

Test 10 - LOOCV left wrist, right ankle

BraunBlanquet classifier *Confusion Matrix:*

	0	1	2	3	4	5	6	7	8	9	10	11	12
0	99	0	0	0	0	0	0	0	0	0	0	0	0
1	0	98	0	0	0	0	0	0	0	0	0	0	0
2	0	0	99	0	0	0	0	0	0	0	0	0	0
3	0	0	0	95	0	0	0	0	0	0	1	3	0
4	0	0	0	0	99	0	0	0	0	0	0	0	0
5	0	0	0	0	0	97	0	0	0	0	0	1	0
6	0	0	0	0	0	0	96	0	0	0	0	2	0
7	0	0	0	0	0	0	0	99	0	0	0	0	0
8	0	0	0	0	0	0	0	0	85	0	0	11	0
9	0	0	0	0	0	0	0	0	0	85	0	9	0
10	0	0	0	0	0	0	0	0	0	0	95	0	0
11	0	0	0	0	0	0	0	0	0	0	0	98	0
12	0	0	0	1	0	0	0	0	0	0	0	0	97

$$Accuracy = 97{,}79\%$$

From the results we see that the accuracy decreases only by about 0.6 percent points compared to the results obtained using the best single classifier (*BraunBlanquet*) with all the sensors.

Test 11 - LOOCV hip

BraunBlanquet classifier *Confusion Matrix:*

	0	1	2	3	4	5	6	7	8	9	10	11	12
0	99	0	0	0	0	0	0	0	0	0	0	0	0
1	0	98	0	0	0	0	0	0	0	0	0	0	0
2	0	0	99	0	0	0	0	0	0	0	0	0	0
3	0	0	0	95	0	0	0	0	1	0	3	0	0
4	0	0	0	0	99	0	0	0	0	0	0	0	0
5	0	0	0	0	0	97	0	0	0	0	1	0	0
6	0	0	0	0	2	0	94	0	0	0	0	1	0
7	0	0	0	0	0	1	0	98	0	0	0	0	0
8	0	0	0	0	0	0	0	0	71	12	4	8	1
9	0	0	0	0	0	0	0	0	16	59	12	7	0
10	0	0	0	0	0	0	0	0	0	0	86	9	0
11	0	0	0	0	0	0	0	0	0	0	2	96	0
12	0	0	0	0	0	0	0	0	0	0	0	0	98

$$Accuracy = 93{,}62\%$$

Using only hip sensors accuracy decreases by about 4.8 percent points compared to the result obtained using the best classifier (*BraunBlanquet*) with all the sensors. The classifier tends to make more mistakes in recognizing "go upstairs," "go downstairs," and "jog".

Test 12 - LOOCV right wrist

BraunBlanquet classifier *Confusion Matrix:*

	0	1	2	3	4	5	6	7	8	9	10	11	12
0	99	0	0	0	0	0	0	0	0	0	0	0	0
1	0	98	0	0	0	0	0	0	0	0	0	0	0
2	0	0	99	0	0	0	0	0	0	0	0	0	0
3	0	0	0	95	1	0	0	0	0	0	0	3	0
4	0	0	0	19	79	0	0	0	0	0	0	1	0
5	0	0	0	12	0	84	0	0	0	0	0	2	0
6	0	0	0	1	3	0	92	0	0	0	0	2	0
7	0	0	0	0	0	18	0	81	0	0	0	0	0
8	0	0	0	0	0	0	0	0	70	21	0	5	0
9	0	0	0	0	0	0	0	0	19	71	0	4	0
10	0	0	0	0	0	0	0	0	0	0	95	0	0
11	0	0	0	0	0	0	0	0	0	0	0	98	0
12	0	0	0	0	0	0	0	0	0	0	0	0	98

$$Accuracy = 91{,}25\%$$

Using only the right wrist sensors accuracy decreases by about 7 percent points compared to the results obtained using the best classifier (*BraunBlanquet*) with all the sensors. The classifier tends to make more mistakes in recognizing “walk forward left-circle”, “walk forward right-circle”, “turn right” ,“go upstairs” ,“go downstairs”.

Test 13 - LOOCV right ankle

BraunBlanquet classifier *Confusion Matrix:*

	0	1	2	3	4	5	6	7	8	9	10	11	12
0	92	7	0	0	0	0	0	0	0	0	0	0	0
1	45	53	0	0	0	0	0	0	0	0	0	0	0
2	0	0	99	0	0	0	0	0	0	0	0	0	0
3	0	0	0	74	0	0	0	0	14	2	2	0	7
4	0	0	0	0	96	0	3	0	0	0	0	0	0
5	0	0	0	0	0	98	0	0	0	0	0	0	0
6	0	0	0	0	0	0	98	0	0	0	0	0	0
7	0	0	0	0	0	0	0	99	0	0	0	0	0
8	0	0	0	0	0	1	0	0	71	0	0	18	6
9	0	0	0	0	0	0	0	0	5	70	2	16	1
10	0	0	0	0	0	0	0	0	1	4	87	3	0
11	0	0	0	0	0	0	0	0	7	0	0	91	0
12	0	2	0	10	0	0	0	3	20	0	1	2	60

$$Accuracy = 85{,}66\%$$

Using only the right ankle sensors accuracy decreases by about 12 percent points compared to the results obtained using the best classifier (*BraunBlanquet*) with all the sensors. The classifier tends to make more mistakes in recognizing “rest at sitting”, “walk forward”, “walk forward”, “go upstairs” ,“go downstairs”, ”push wheelchair”.

WARD DatabaseTest – First Considerations

The WARD 1.0 database was created at UC. Berkeley by Yang. et al. in 2009, the reason to use this database is mainly to compare our test results with the results of Berekely researchers on the *same database.*

From the tests we see that the accuracy is very high, achieving **an accuracy 98.43%** with the ***BraunBlanquet, Tanimoto, Dice*** and ***Kulczynski 1*** classifiers. Using *Majority Voting* combination the accuracy remains unchanged.
If we compare the best accuracy obtained at U.C. Berkeley (see "State of the art" reference [2]) i.e. **93.4%,** with the accuracy here measured **98.43%,** these results are about 5 % higher using the same database. The **error rate** has been lowered from **6.6%** to **1.57%**, i.e. more than **4 times.**

The results obtained with **fewer sensors** say that even with one sensor we can recognize more than 85% of the actions. In particular, with **three sensors** the accuracy here measured is **98.26%** (Test 7), that is just 0.17% less then best result obtained with 5 sensors. Using only **one sensor** on the hip and ***BraunBlanquet*** the accuracy of **93.62%** was obtained, that is sligthly higher than **93.4 %** obtained by [A.Y.Yang et al.] with all the **five sensors using this database.**
In addition, in the Test 5 and Test 6 we see that the **right** and the **left** side of the body are involved in the same way in the recognition process; using just three sensors in both cases accuracy is very similar to that obtained when using all the 5 sensors.

Intracluster and Intercluster Similarity

The high accuracy obtained with classifiers that do not use discrimination surface to classify, suggests that the *FFxIVFF* operation affects dimension, density and distance of the clusters of data. Hence, the Intracluster and Intercluster similarity has been calculated on both databases before and after the *FFxIVFF* operation.
Intracluster and Intercluster similarity has been measured in the values space, in the *FF* space and finally in the *FFxIVFF* space; results have been compared (Table 11). The cosine similarity has been used as a similarity measure.

Database	**Type**	**Value Space (Cos)**	**FF space (Cos)**	**FFxIVFF space (Cos)**
Nida	Intercluster	**0,3506425**	0,1710151	**0,0286604**
	Intracluster	**0,7077682**	0,8759209	**0,9260479**
Ward	Intercluster	**0,2476419**	0,2367639	**0,0605941**
	Intracluster	**0,6038969**	0,8455223	**0,9311265**

Table 11 – Intracluster Extracluster Similarity before and after FFxIVFF

Generally speaking, it is possible to note that *FF* and *IVFF* actually enhance cluster density and cluster separation. In particular using the NIDA database we can note that:

- the *IVFF* enhance the Intercluster similarity **5.96 times** respect to *FF*, and **12.23 times** respect to the values space
- *FF* enhance the Intracluster similarity, while thanks to *FFxIVFF* Intracluster similarity passes the critical threshold of .9

using WARD database we can note that:

- the *IVFF* enhance the Intercluster similarity **4.08 times** respect to the values space
- the FF enhance the Intracluster similarity, while thanks to FFxIVFF Intracluster similarity passes the critical threshold of .9

WARD and NIDA Dataset Representation

The former results show that FF and *IVFF* enhance clusters density and clusters separation. These results are also confirmed by a visual analysis of the WARD and NIDA datasets.

To see the effects of the *FFxIVFF* on the WARD and NIDA dataset the original dimension of the space has been reduced to a three dimension space using a Principal Componente Analysis (PCA) technique. The three dimensional vectors obtained have been represented using Matlab.

In Fig. 16 and Fig. 17 we can see the **WARD** dataset before the *FFxIVFF* operation: centroids and variances of the clusters of actions have been calculated and represented. It is possible to note that the centroids are very close (Fig. 16) and the clusters are intersected (Fig. 17).

In Fig. 18 we can see the **WARD** dataset *after* the *FFxIVFF* operation: centroids are well separated, and clusters are further apart and more dense as suggested by Intracluster and Intercluster similarity values

In Fig. 19 we can see the **NIDA** dataset *before* the *FFxIVFF* operation: centroids and variances of the cluster of actions have been calculated and represented. Again it is possible to note that the centroids are very close and the clusters are quite mixed. In Fig. 20 and Fig. 21 we can see the **NIDA** dataset *after* the *FFxIVFF* operation: centroids are well separated and clusters are further apart and more dense as suggested by Intracluster and Intercluster similarity values.

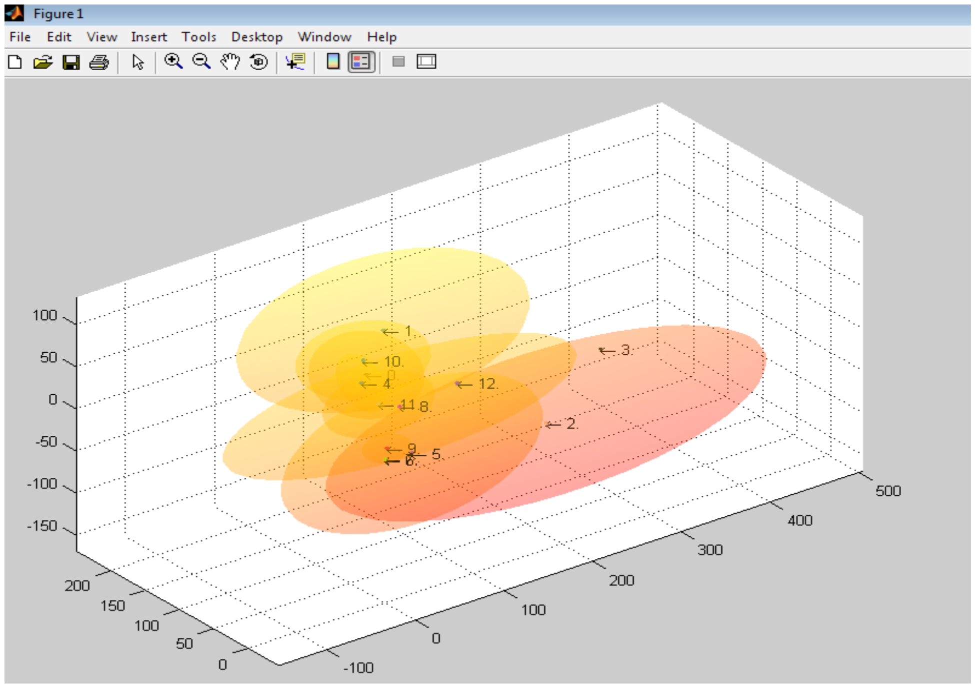


Fig. 16 – WARD dataset before the FFxIVFF operation

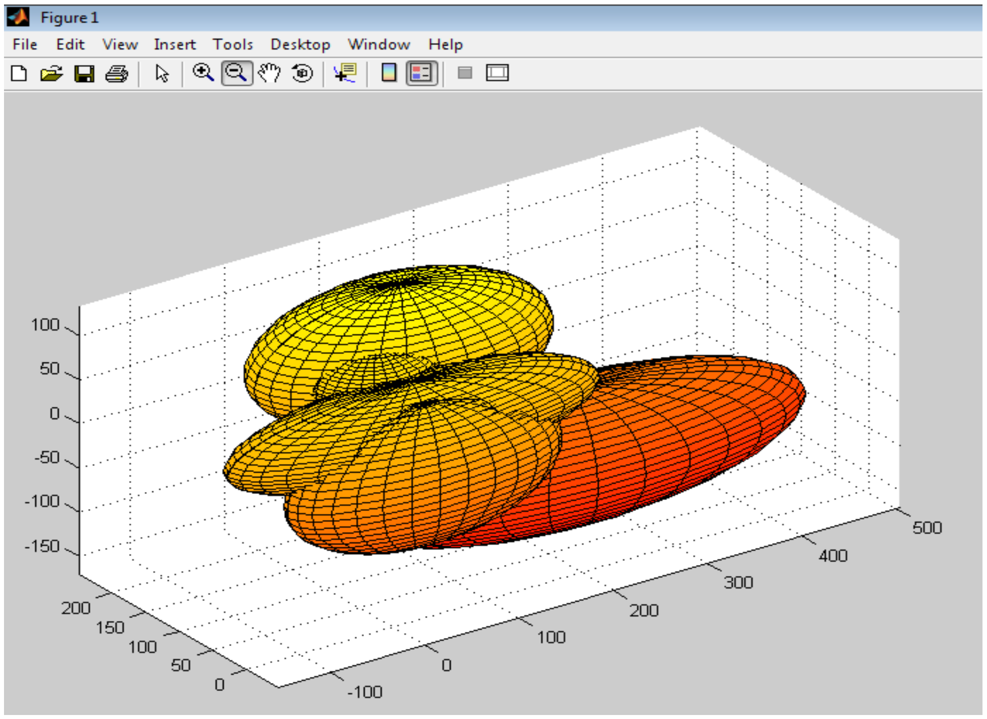


Fig. 17 – WARD database: before the FFxIVFF operation clusters are intersected

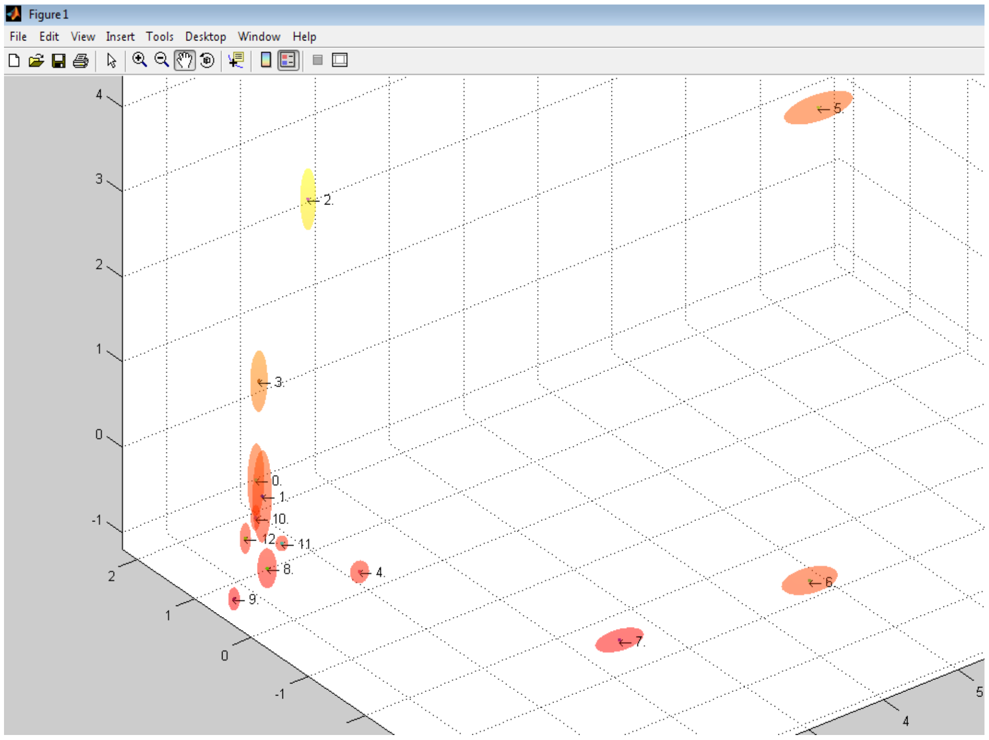


Fig. 18 – WARD database: after the FFxIVFF operation clusters are well separated

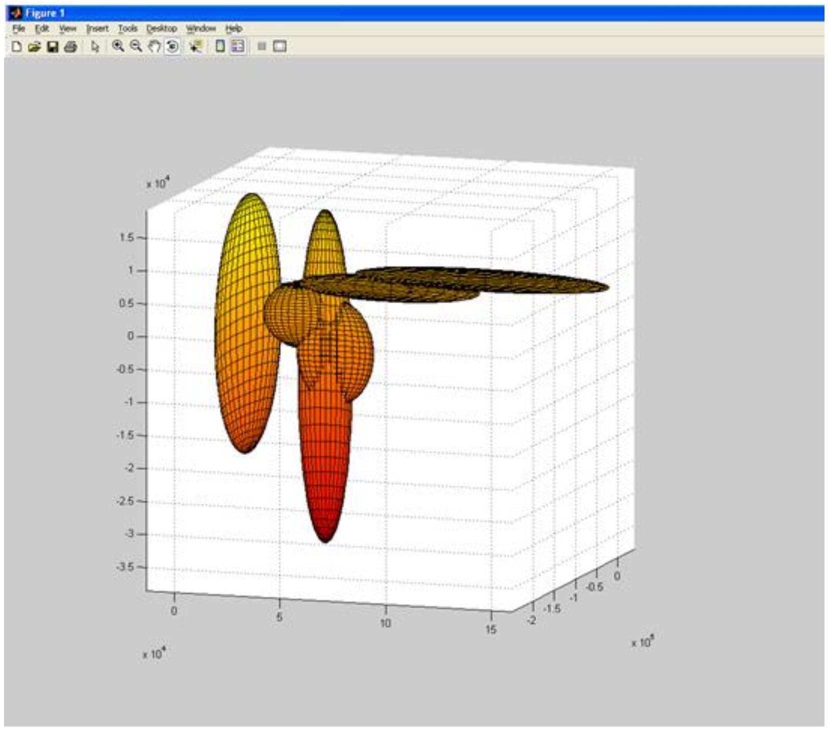


Fig. 19– NIDA dataset: before the FFxIVFF operation clusters are intersected

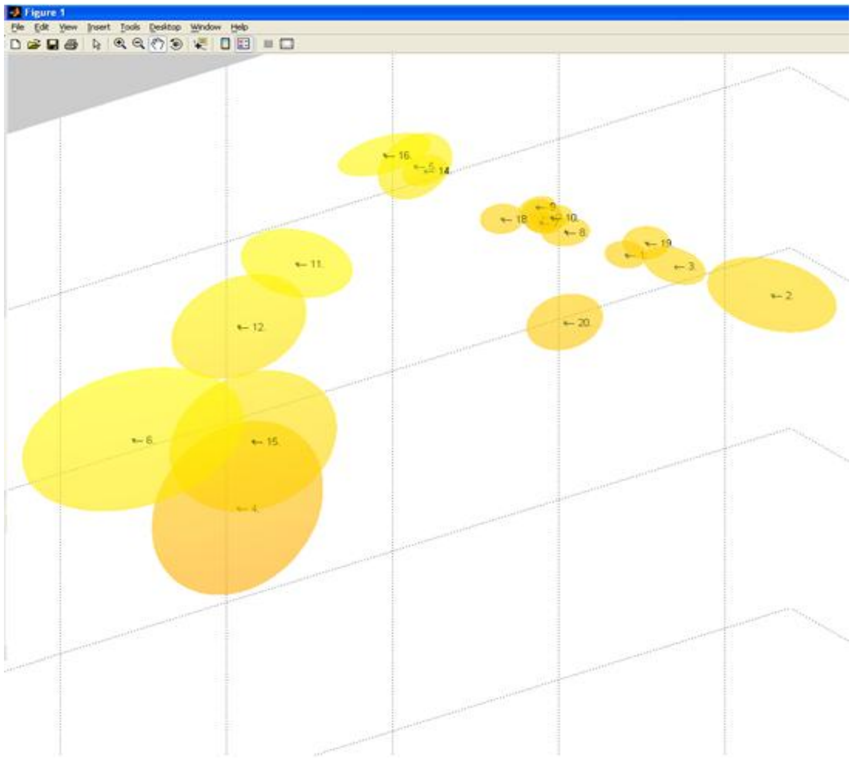


Fig. 20 – NIDA dataset: after the FFxIVFF operation centroids are further apart

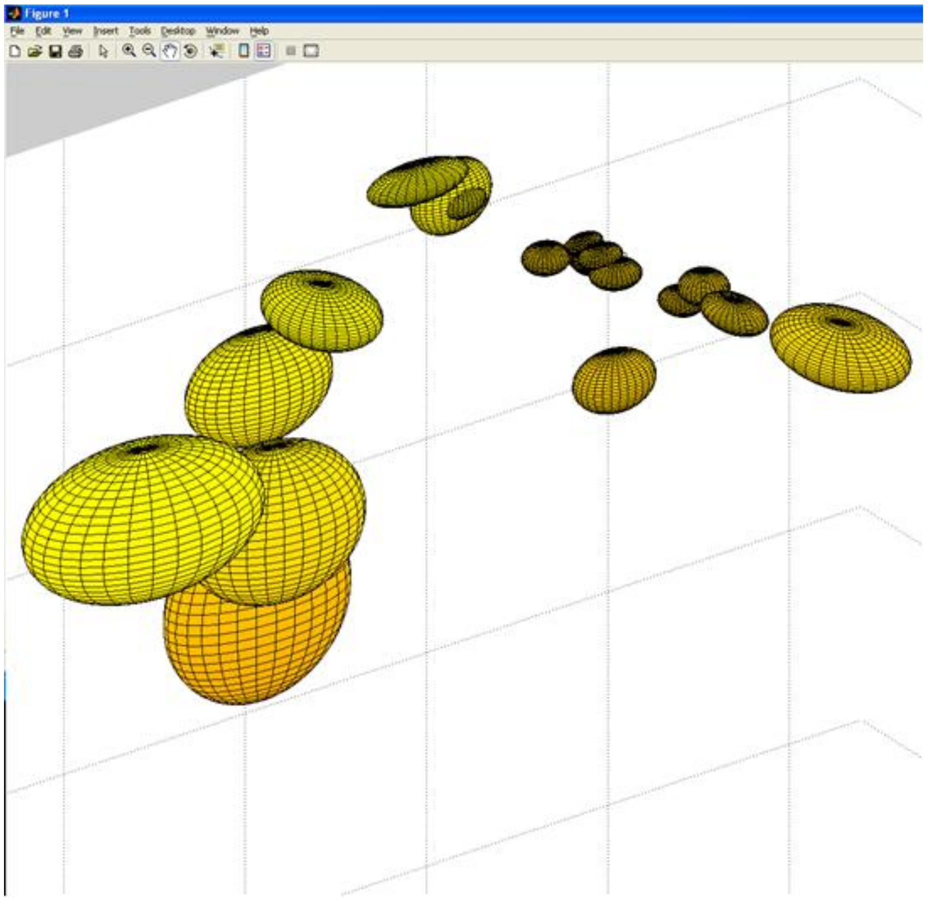


Fig. 21 – NIDA dataset: after the FFxIVFF operation clusters are well separated

Conclusions

In this chapter I summarize the results of the tests done on both NIDA and WARD datasets in order to compare the accuracy and perfomance, I show and compare the most interesting results, varying the number of sensors, and I show the Intracluster Intercluster similarity values before and after the *FFxIVFF* . Finally, the conclusions.

Tests with NIDA and WARD Databases

Using the **NIDA 1.0** database with 21 actions, executed 13 times (10 male, 3 female) with similar "Human and Body Profile", for a total of 273 actions, the best results are:

- Single classifier
    - *Bray-curtis:* accuracy 96.70 %
    - *Tanimoto, Dice, Kulczynski 1, NClassifier, Chi-Square:* accuracy 95.60 %
    - *Cosine* accuracy: 95.23%
- Majority Voting:
    - *Cosine Similarity, Manhattan, Dice, Camberra, Bray-Curtis:* accuracy 97.43 %

Using the **WARD 1.0** databse created at U.C.Berkeley with 13 Actions 5 ripetitions each; 20 People (13 male, 7 female) ranging from 20 to 79 year-oldSensor Profile similar to NIDA; calibration and normalitation done as suggested by the authors; the best results are:

- Single classifier:
    - *Braun-blanquet, Tanimoto, Dice, Kulczynski 1:* accuracy 98.43 %. Error Rate: 1.57%
    - *Bray-Curtis, Ranking, Cosine Similarity, Pearson correlation:* Accuracy 97.6-98.0 %
- Majority Voting*:*
    - Identical accuracy of single classifiers, in the best case.

Comparing these with the results of [A.Y. Yang et al. 2009] on the **WARD 1.0** database, their best result is:

- Distributed Sparsity Classifier: accuracy 93.4 %. Error rate 6.6%

We can see the **error rate** was **lowered** by more than 4 times.

We have to note that in the literature related to activity recognition with inertial sensors it is not usual to test and confront results using public datasets. Given the diversity of the domains (dictionaries), test approaches are usually quite specific. Hence, sometimes it is very difficult to evaluate and compare the results.
As my proposed methodology is “technology independent” it becomes natural to test these methods on both databases. The accuracies obtained are very interesting. In fact, the *FFxIVFF* method perform much better than the DSC method proposed by A.Y.Yang et al.. using the same dataset. Moreover, these results have been obtained using classifiers that are very simple from a computational point of view.
The accuracy results obtained using the WARD database clearly shows that the *FFxIVFF* improves the accuracy. That suggests that the *FFxIVFF* could affects dimension and reciprocal distances of the cluster, as confirmed by the Intracluster Intercluster similarity values, and by PCA analysis.

Tests Varying the Number of Sensors

Table 12 shows a synoptic view of the most interesting results obtained using both databases varying the number of sensors. The *Bray-Curtis* classifier has been used for the NIDA database, and the *BraunBlanquet* classifier has been used for the WARD database.

# of sensors	**Sensors**	**NIDA accuracy**	**WARD accuracy**
5	hip, left/right wrist, left/right ankle	97. 43 %	98.43 %
3	hip, right wrist, right ankle	94.87 %	98.26 %
3	hip, left wrist, left ankle	94.13 %	98.26 %
3	hip, letf wrist, right ankle	96. 33 %	98.26 %
3	hip, right wrist, left ankle	91.20 %	96.92 %
1	hip	81.31 %	93.62 %
1	right wrist	82.78 %	91.25 %
1	right ankle	83.15 %	85.66 %
5	*A.Y .Yang, et al.* (UC Berkeley)	-	93.4 %

Table 12 – Accuracy of FFxIVFF varying the number of sensors compared.

On both databases we can observe that using **3 sensors** (left or right side of the body) results are quite accurate and comparable with the same results using **5 sensors**. That is particularly true for the **WARD** dataset where with 3 sensors the accuracy is lowered by only 0.17 percentage points.

Note that using the **WARD** dataset the accuracy obtained with the *FFxIVFF* with **3 sensors** (98,26 %) and with **1 sensor** (93,62 %) is *better* than [A.Y. Yang et al. 2009] with all the **5 sensors** (93,4 %) on the same dataset.

NIDA and WARD Results Compared

The tests clearly demonstrate that the classifiers which have the highest accuracy on databases are *Bray-Curtis* for NIDA and *BraunBlanquet* for WARD 1.0. To compare these results we must consider them also in relation to the number of types of actions present in the dictionary. An accuracy of 50% on a database with only 2 types of actions, does not have the same value than an accuracy of 50% on a database containing 100 types of actions. To compare the obtained accuracies, first we calculate the N_1 value for WARD dictionary:

$$\frac{1}{13} * N_1 = 98{,}42 \rightarrow N_1 = 1279{,}46$$

then the N_2 value for NIDA dictionary :

$$\frac{1}{21} * N_2 = 97{,}43 \rightarrow N_2 = 2046{,}03$$

Now the two quantities are compared:

$$Q = \frac{N_2}{N_1} = 1{,}60$$

where Q indicates the ratio of the accuracy obtained with NIDA and WARD in the best cases.
The Q factor shows that the quality of test results on the NIDA is greater by a factor of 1.60 compared to the results of tests on the WARD database. Since the number of type of actions in the NIDA database are more numerous respect to the number of types of actions of the WARD database, we can say that the accuracy obtained for NIDA are more accurate and reliable.

Intracluster and Intercluster Similarity

The high accuracy obtained with classifiers, suggests that the *FFxIVFF* operation affects dimension, density and distance of the clusters of data. This is confirmed by the values of Intracluster and Intercluster similarity calculated on both the databases before and after the *FFxIVFF* operation (see Table 11). Generally speaking is possible to note that *FF* and *IVFF* actually enhance cluster density and cluster separation. In particular, using NIDA database we can note that:

- the *IVFF* enhance the Intercluster similarity **5.96 times** respect to *FF*, and **12.23 times** respect to the values space
- *FF* enhance the Intracluster similarity, while thanks to *FFxIVFF* Intracluster similarity passes the critical threshold of .9

using WARD database we can note that:

- the *IVFF* enhance the Intercluster similarity **4.08 times** respect to the values space
- the FF enhance the Intracluster similarity, while thanks to FFxIVFF Intracluster similarity passes the critical threshold of .9

Final Conclusions

The Goal was to create a general methodology that could be used on different technologies, and domains without changing the given framework for activity reconigtion with inertial sensors.
Given a set of action samples, an interative process of feature extraction that uses very generic transformations and very generic features transforms the executed actions in their vector representation in the value space. Quantization transforms the feature-value vectors, in binary feature-interval vectors.

In order to create a flexible methodology usable in different domains, the features have been weighted using two different criteria, the generality of a feature in the Population (intra-class frequency), the ability to discriminate the Actions of the Dictionary (inter-class frequency). This has been done by the *FFxIVFF* weighting operation of the features. *FFxIVFF* values are context dependent: they depend both on Dictionary and Population. However, given the database, their values can be very easily automatically calculated. This assures that the

methodology is usable in *different domains*, with vocabularies of different dimensions using an automatic operation.
The *FFxIVFF* transforms the original space, into a different space where clusters are denser and more separated, as the measure of Intracluster and Intercluster similarity suggests. Thanks to this operation the accuracy is greatly increased and we can use very simple classifiers which have low computational costs. Twentytwo classifiers have been chosen (e.g. Cosine, Euclidean Distance, Tanimoto, etc).

I used two different databases to tests the effects of the *FFxIVFF,* and the accuracy of given classifiers: the NIDA 1.0 database, with 21 types of actions that use the XSensor Technolgy, 5 inertial unit with 3-axial accelerometers, 3-axial gyroscopes, and a 3-axial magnetometers. The WARD 1.0 is a public database created at U.C.Berkeley with 5 wireless inertial unit with 3-axial accelerometers, 2-axial gyroscopes and no magnetomers.

The results show that accuracy is very high on both databases (96.70% NIDA, 98.43% WARD) outperforming similar results present in the literature on the same database (see [A.Y. Yang et al. 2009]). The method was tested in both the databases without changing the architecture, the feature extracting procedure, the testing methodology, obtaining high accuracies. That confirms that the method is technology independent.
With the given the methodology you do not need to be a domain expert, or an inertial sensors expert. In particular, a biomechanics competence on movements or posture for the given domain is not needed.
Also, the method does not have a strong dependency on the position of sensors or on their number, a very high accuracy has been achieved using just 3 sensors on both database, in particular on the WARD database (having only 3 accelerometers, and 2 gyroscopes per sensor) outperforming the results given in literature with 5 sensors in the same dataset [2]. It has been obtained an high accuracy by placing sensors in diagonal (pelvis, right wrist and left ankle). Good results have been obtained using just one inertial sensor on the hip.

Conceptual clearness, high accuracy, technology independency, low sensitivity to the number of sensor variations, and the use of fast algorithms, give this methodology a great appeal. Also, the FFxIVFF operation generates a space where clusters are denser and well separated, allowing the use of very simple algorithms. The method opens a new and interesting approach to activity recognition with inertial sensors, which is general, flexible, technologically independent, accurate, and also creates a bridge between the instance based techniques, the application domain and the semantic domain.

Appendix A - Instrumentation

The instrumentation used for movements acquisition is the X-Bus Kit, produced by the X-Sens company that contains an X-Bus Master and 5 MTx Sensors ([3]). TheMTx Sensor are inertial unit with three mounted devices on board, a 3 dimensional accelerometer, a 3 dimensional gyroscope and a 3 dimensional magnetometer. When positioned on the body, the inertial units returns data about the movements of the body segments where the inertial units have been positioned.

The Xbus Master is a device that can control up to 10 inertial units with two different buses (maximum 5 units per bus). It can be connected via serial port or Bluetooth to a personal computer (PC) or to a mobile device (e.g. a Personal Digital Assistant), where data can be analyzed and interpreted. As we can see in Fig. 38, sensors are connected in serial configuration using a cable, than they are connected to the X-Bus Master. Thanks to this configuration and using the two buses we can place inertial units practically on every segment of the body. A Hewlett Packard Personal Digital Assistant with a Bluetooth and a Wifi device is used to send the data from the Xbus to the personal computer for runtime analysis (see Fig.47). A specifically developed software called *Motus* has been used to drive the acquisition process, and to send data to the Personal PC (see Fig. 39).

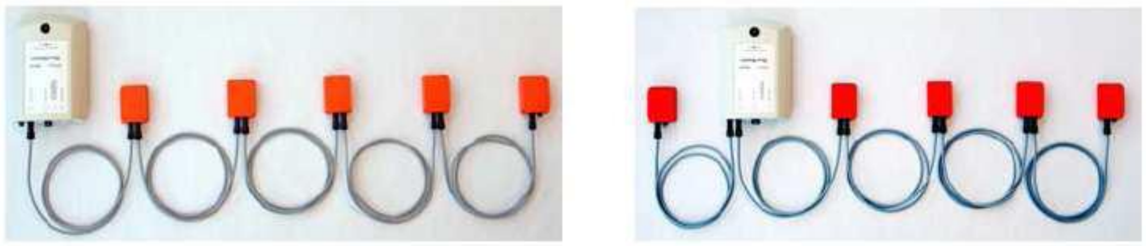
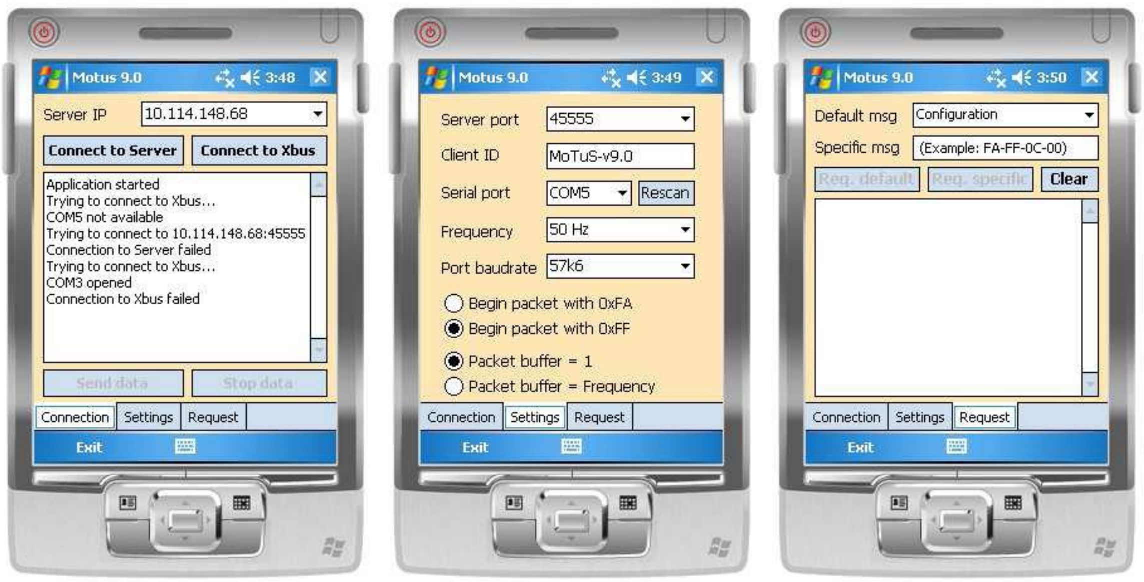


Fig. 38 - "XBus Master" and the "MTx" sensors

The reason to develop this application is the need to have a lightweight device that can be worn on body to acquire and store the data. The second reason is to create a common interface for data analysis: the server acquires data on a specific port, if we want to change the sensor technology, we just have to rewrite part of the Motus code if necessary, then we just have to direct data from the new sensors to the given TCP/IP port to start the analysis process on the server side, without changing the server side architecture.

Also, *Motus* is used to send configuration messages to the *XBus Master* (e.g. to set the sample frequency), to start/stop the data reading process, and generically to manage low level messages.

We can see in Fig. 39 the *Motus* application running, and three of its “navigation” interfaces.



```
Server IP 10.114.148.68

Application started
Trying to connect to Xbus...
COM5 not available
Trying to connect to 10.114.148.68:45555
Connection to Server failed
Trying to connect to Xbus...
COM3 opened
Connection to Xbus failed
```

```
Server port    45555
Client ID      MoTuS-v9.0
Serial port    COM5
Frequency      50 Hz
Port baudrate  57k6
( ) Begin packet with 0xFA
(●) Begin packet with 0xFF
(●) Packet buffer = 1
( ) Packet buffer = Frequency
```

```
Default msg   Configuration
Specific msg  (Example: FA-FF-0C-00)
```

Fig. 39 – Motus and its navigation interfaces

MTx Sensors

The MTx Sensor are inertial sensors units. Every unit comes with three mounted devices: an accelerometer, a gyroscope, and a magnetometer. Every single device has three degree of freedom and gives information respectively about change of velocity (m/sec^2) , rate of turn (rad/sec), intensity of the magnetic north pole vector (milliTesla).

The inertial unit and the embedded devices

Every MTx inertial unit contains three sensors devices. Here a specific description of the devices, the output data and a first explanation of how to interpret data.

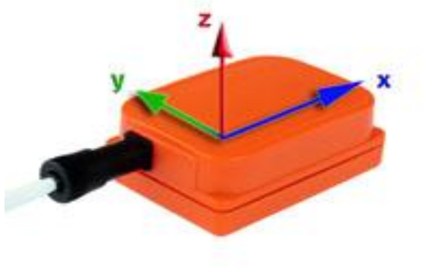


Fig. 40 - “MTx” sensor and its reference system (S)

In Fig. 40 has been represented the coordinate system of the sensor (S) . This reference system (S) is conventionally related to a fixed reference system (G). In the (G) reference system, the X axis is positive when it is parallel the magnetic north and point to it, Z is positive when parrallel to the gravity acceleration vector **g** and opposite to it, and conventionally Y is positive when ponting to west. In Fig. 41 we can see repesented the two reference systems (S) and (G).

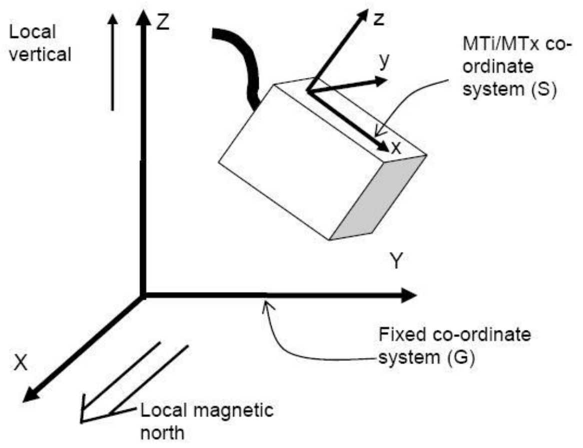


Fig. 41 The reference systems (S) and the fixed reference system (G)

It is possibile to pass from the (S) to (G) coordinates system using a rotation matrix that is calculated using quaternions, but as pinpointed before quaternions were avoided in order to test the classification method in the most general way.

We have to note that even if it is possible to obtain the Euler angles of *MTx* sensor in respect to the (G), and to know the tilting orientation of sensors, it is not possible to know the position of the MTx Sensor in respect to the the fixed triple of coordinates (0,0,0) of the reference system (G). Consequently the information we will use for the recognition are relative only to change of velocity, rate of turn, and north pole direction in the three-dimensional reference system (S) of to the sensors itself.

Accelerometer

The accelerometer is an instrument that measures the acceleration of a body, the operating principle is based on the inertia of a mass when subjected to acceleration. The device contains a mass attached to an elastic element, which detects the displacement relative to the fixed structure that supports it.

What we measure is three spatial components X, Y and Z of the sum of the acceleration of sensor and the force of gravity, expressed in $\mathrm{m/s^2}$.

Due to the sensitivity of the MTx sensors, we can see important information about the movements. In the Fig. 5 we can see the X, Y, Z components of acceleration, and magnitude, of a person walking wearing a single sensor placed on the hip.

In this example the Y axis is parallel to spinal cord, the X axis is parallel to the direction of walking, the Z axis is external to the saggital plane. The subject walks three times, and remains sit before every walking activity. The magnitude of the three components (yellow), is very close to the values of the Y component, since the other two spatial components tend to reciprocally eliminate.

The Y component (in blue), gives probably the most significant information to “categorize” this type of movement. But also the other two components, X and Z depict a repetitive pattern. It is easy to see also that when the person is sitting, there is a significant change in the relative value of the three components. This first simple example gives at a glance the power, and the problems of detecting an activity using an inertial sensor. The information is simple, but obviously is not very intuitive. In order to understand what is going on, we should know very well the a priori pattern we are searching for, where to search it. Also there is a lot of “noise” in the information given by physiological and pathological differences in the ability to move of people. Also, the position and number of sensor, change greatly the type, importance and interpretation of the given data.

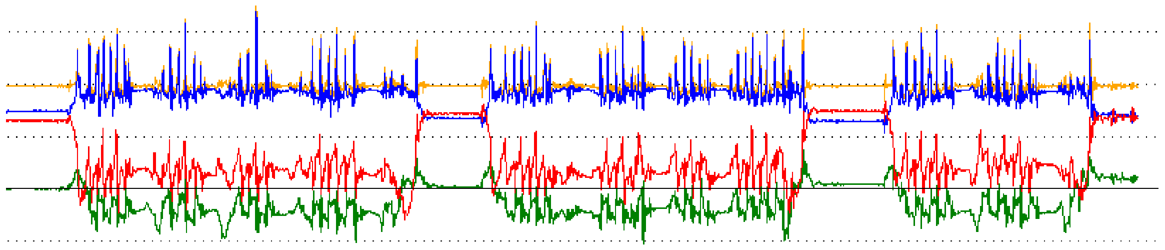


Fig. 42 – Accelerometer data about 3 walking

Gyroscope

The gyroscope provides the angular velocity of the sensor along the three spatial axes expressed in rad/s. A gyroscope contains a spherical shape, mounted on a gimbal that can rotate in any direction. Due to the law of conservation of angular momentum, it tends to maintain its axis of rotation oriented in a fixed direction. The high inertia keeps the axis of rotation orthogonal to the plane formed by the rotation axis itself and the applied force. These properties are common to any body in rotation around an axis of symmetry, including Earth.

When steady, the gyroscope values tend to stay close to zero, if the gyro is rotated a peak appears in the positive or negative values (depending on the direction of rotation). The higher is the rate of turn, the more significant is the peak.

In Fig. 43 we see an output sample of the gyroscope. The executed action is a person rotating on itself 6 times.

In this example, one sensor has been placed on the right side of hip: the X axis is parallel to spinal cord, theY axis is parallel to the direction of walking, the Z axis is external to the saggital plan (no filters were applied to the curve).

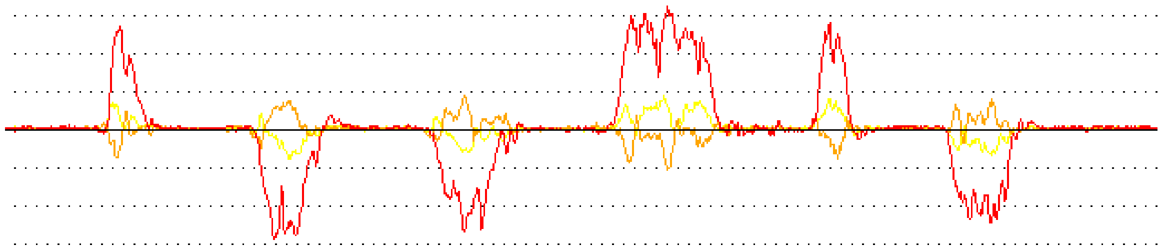


Fig. 43 – Gyroscope data, 6 turns

You may notice that the red component (rotation around the X axis) is the most significant, because the sensor was positioned with the X axis pointing up. The result is a postive peak when the person turns counterclockwise, and negative if the rotation is clockwise (our reference plane is |YZ|, the floor)

Magnetometer

The magnetometer measures the Earth's magnetic field, i.e. the values of the vector pointing to the Earth's magnetic north, in the three axial sensor reference system (S).

While the magnetic north does not move in respect to an ideal fixed reference system (G), the body (or a segment of the body) moves in respect to the north pole, and since the reference frame of the sensor is in solid with the body, when the subject turns on itself you can observe the variation of the vector pointing the north magnetic pole into the three-axial reference system (S).

In Fig. 44, we see the three components of the magnetometer relative to the phenomenon described in the former section (Gyroscopes)

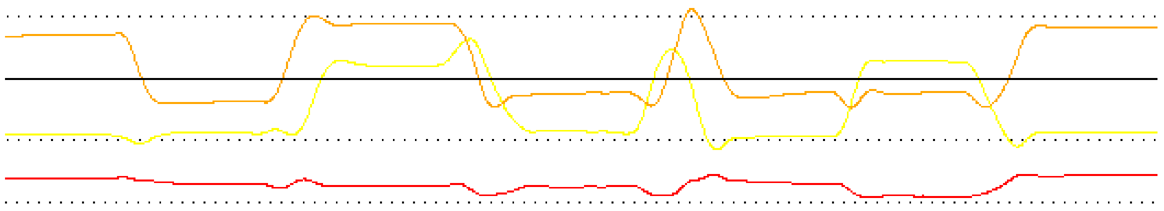


Fig. 44 - 3 axial data of the magnetometer during turns

This graphical representation may appear less intuitive to be understood at a first glance, but we may consider the data more interesting when represented in the X-Y, Y-Z, X-Z planes. For example, if you need to know the rate of turn of a person and specifically his/her angle of rotation, we can consider the plan Y-Z data and calculate the turn angle in this plan using the antitangent function ($\tan^{-1}$) and the Y and Z axis information.

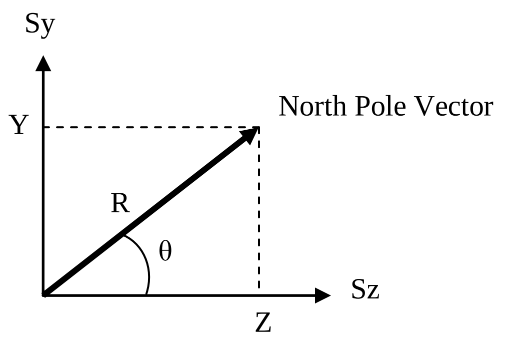


Fig. 45 – North pole vector in the |YZ| plane of the sensor framework

Is well know by trigonometry that

$$Z = R * \cos(\theta)$$
$$Y = R * \sin(\theta)$$
$$\tan(\theta) = \frac{Y}{Z}$$
$$\theta = \tan^{-1}\left(\frac{Y}{Z}\right)$$

so it is also useful to represent the heading of the sensor using the θ angle. Note thar since the tangent is defined just in the domain $(-\pi/2, \pi/2)$, if the angle θ becomes greater than $|\pi/2|$ the function has a discontinuity and value passes form π to $-\pi$ and vice versa (the tangent funcion tends to +/- ∞ when domain values approaches +/- $\pi/2$ values or its multiples).

Other Useful Data Representations

As explained above, each device generates data for each axis of the sensor system. It has been noted that the information generated by the magnitude of the axs provides additional information. For example, if a person

turns he generates data that are more probably related to the orizontal plan, this plan can contain suitable information indicative of the phenomenon. Not only magnitude representation can increase the amount of data useful for analysis, but also could provide information that are less dependent on the physical disposition of the sensors, because magnitude is a scalar quantity. For this reason it was chosen to introduce the magnitude of the two-dimensional planar informations and the magnitude of the three-dimensional space information for each device and sensor.
The formula to calculate the component magnitude is trivial and is as follows:

$$\|V\| = \sqrt{v_1^2 + v_2^2 + v_3^2 + \cdots + v_n^2}$$

Where v_1 , v_2 , ... v_n are the *n* component values of the vector **V.**

In summary, each sensor returns the three dimensional values of each components, than their magnitude plan representation |XY|, |XZ|, |YZ| and the magnitude of spatial representation |XYZ| .
The total number of data generated by every sensor can be calculated using the following formula:

$$SignalsNumber = S * D * C$$

Where *S* is the number of sensors worn on the body, *D* is the number of devices mounted on inertial units and *C* is the number of information given by each device. The formula implies that each sensor has the same number of devices and that each device generates information about the same number of components. In our test, for every action the generated information are 105 (45 “physically” generated, 60 “derived”).

Low level description of data

The MTx sensors are able to provide data in different way. They can be configured to send accelerometer, gyroscope and magnetometer data in "Calibrated mode" (i.e. single-precision floating-point values of 4 bytes each), or in "Un-Calibrated mode” (unsigned integer, 2 bytes each). They can also provide Euler angles (roll, pitch, yaw, single-precision floating-point values of 4 bytes each), quaternions (single precision floating point values), or the rotation matrix (floating point values single precision).

The Calibrated mode was prefereed for its generality, in this mode data are represented as a "strip" of 13 values of 4 bytes each for each sensor connected to Xbus Master. Three record of three values each are used for accelerometer, gyroscope, magnetometer, and four values are used for quaternions. Packets sent by X-Bus Master use a format that depends not only on the selected output mode (e.g. “Calibrated” “Uncalibrated”), but also by the number of sensors connected to X-Bus Master.
Each transmission begins with a series of data that provides information about the sender, the number of connected sensors, and then the inertial data. The format is as follows:

Field	**Description**
Preamble	Starting packet
BID (Bus identifier)	The address of the device
MID (Message identifier)	Type of message sent by *“XBus Master”*
Length	Amount of data contained in a single data packet
Extended Length	Used only if the number of sensors connected to *“Xbus Master”* is grater than four, and is used to count the number of additional data sent from *"Xbus Master"*
Data	It contains the serialized data sent by the sensors in the format specified in the MID.
Checksum	Checksum for sent data

Table 5

The Fig. 46 gives a schematic representation of data packet sent by the XBus Master.

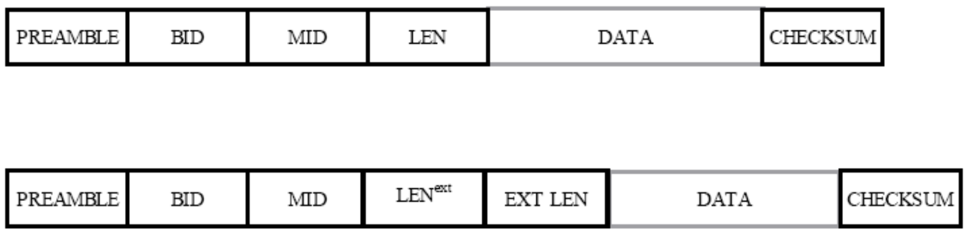


Fig. 46 - “XBus Master” packet

The expected format of the data is the following.

Field	Dimension	Comment
Preamble	1 byte	Constant values 0xFA
BID	1 byte	Constant value 0XFF
MID	1 byte	Expected 0x32
Length	1 byte	If the value is grater than 0xFF, more than 4 sensor has been connected to *Xbus Master*
Extended Length	2 byte	Length of the additional data sent from *"Xbus Master"* (this field may not be present)
Data	Not fixed	Formatted data
Checksum	1 byte	Checksum

As already mentioned, the format of the data in “Data” field depends on the type of MID field. The MID has the expected value “0x32” which specifies that the sensors are ready for measurements and *X-Bus Master* is ready to send data. The values in the “Data” can have different configurations: the used mode is the forementioned "Calibrated mode" with quaternions. Each sample consists of 13 single precision floating-point data, following the "IEEE 754" standard that uses 4 bytes each (see [4]). Fig. 47 we can see represented the communication architecture.

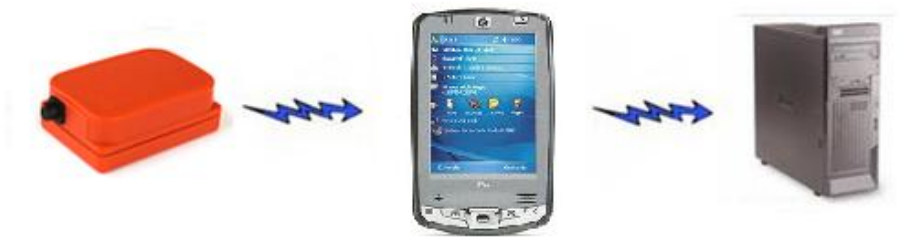
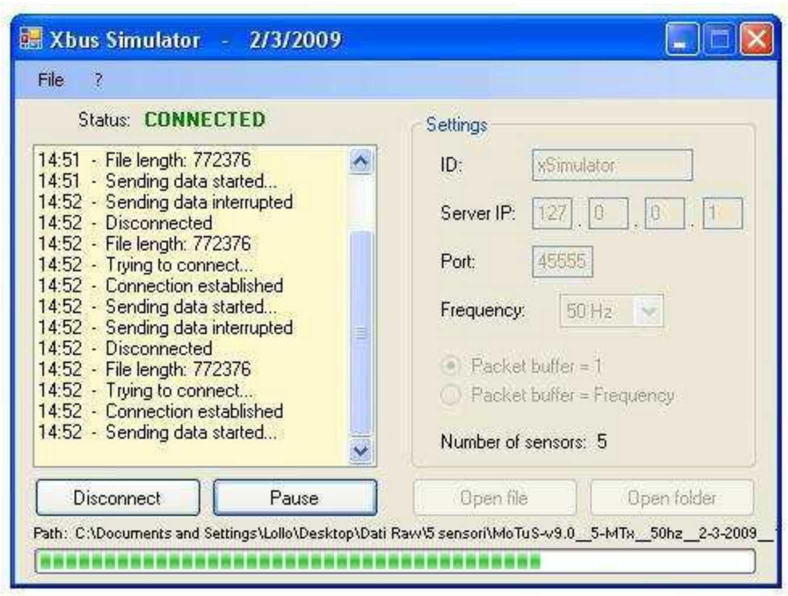


Fig. 47 - “XBus Master” communication chain

XBus Simulator

Xbus Simulator is a C# application that emulates the Xbus Master. The purpose of this emulator is to send data to the server for analysis, using the previously acquired sample of movements saved into files. This allows to run the acquisition, whenever it becomes necessary instead of wearing the sensors and repeat over and over again the same motions. Fig. 48 shows the program running.



Status: CONNECTED

```
14:51 - File length: 772376
14:51 - Sending data started...
14:52 - Sending data interrupted
14:52 - Disconnected
14:52 - File length: 772376
14:52 - Trying to connect...
14:52 - Connection established
14:52 - Sending data started...
14:52 - Sending data interrupted
14:52 - Disconnected
14:52 - File length: 772376
14:52 - Trying to connect...
14:52 - Connection established
14:52 - Sending data started...
```

Settings

ID: xSimulator

Server IP: 127 . 0 . 0 . 1

Port: 45555

Frequency: 50 Hz

[✓ Packet buffer = 1]

Packet buffer = Frequency

Number of sensors: 5

Path: C:\Documents and Settings\Lollo\Desktop\Dati Raw\5 sensori\MoTuS-v9.0__5-MTx__50hz__2-3-2009__

Fig. 48 - "XBus Simulator" running

Appendix B - Filters

Here is a detailed description of Filters used for the process of *feature extraction.*

Smoothing

This filter is designed to flatten signal variations usually correlated to signal noise. The defect of this function is that it also flattens the peaks, even those who do not represent noise in the signal. The function calculates the arithmetic mean of values over a mobile window, which means that the window is shifted one value at a time on the set of values to be filtered. The greater the window the greater is the smoothing effect of the filtered signal. For the feature extraction process the correct dimension of the window has been choosen with heuristically. Below in Fig. 49, an example of a raw signal (red) and a smoothed signal (blue) with a window of 20 values.

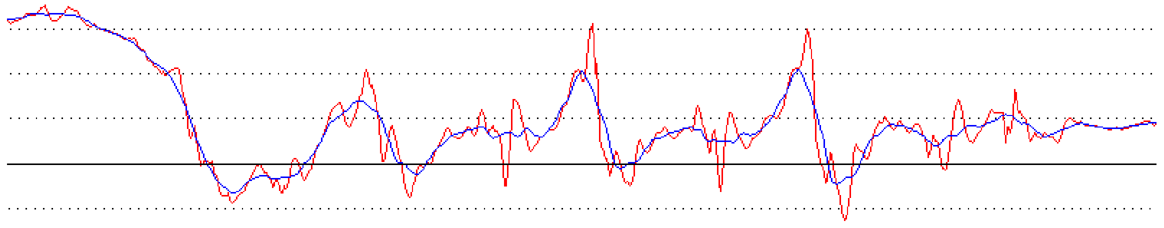


Fig. 49 - In blue data smoothed with a windows of 20 values

In Fig. 50 an example of a signal (red) with a smoothin filter (blue) with a window of 100 values. In the second example, the smooth effect is excessive.

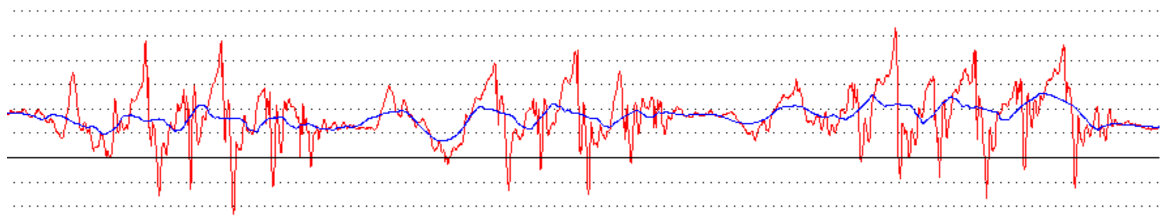


Fig. 50 - In blue data smoothed with a windows of 100 values

Low Pass Filter

This filter has been implemented using using the Discret Fourier Transforms (DFT). The purpose of this filter is to eliminate the noise in signals without altering the height of the most significant peaks. This is achieved by moving the signal from time domain to the frequency domain, and cutting the frequencies above a certain threshold. In Fig. 51 we can see an example of an acquisition where frequencies above 3 Hz have been eliminated (in blue). Peaks are flattened but mostly preserved.

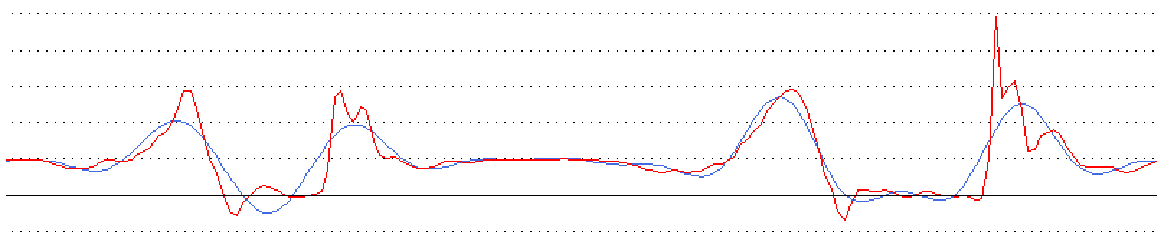


Fig. 51 - Signal filtered cutting the frequencies above 3 Hz

The Fig. 52 represent a signal filtered using a low-pass filter with a cut frequency of 10 Hz. For the feature extraction process the cut frequency has been choosen using an heuristic, and has been set to 4 Hz.

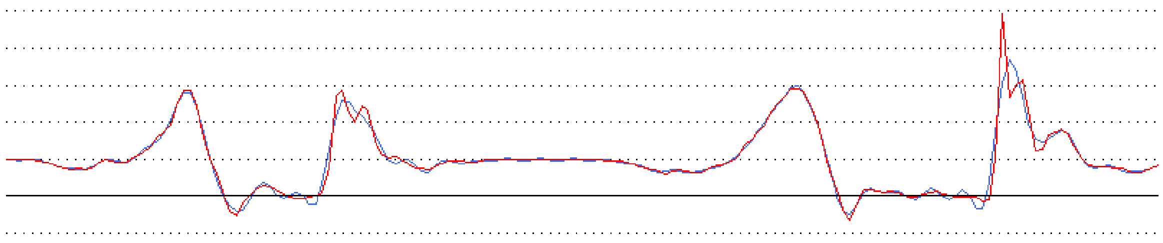


Fig. 52 – Signal filtered cutting the frequencies above 10 Hz

Variance

The *variance* is a measure of statistical dispersion useful to understand how the values of a population deviate from the mean. In the analysis of sensory data it is very useful to identify the start of an event, because a change in the value of variance indicates a change of a certain intensity in the signal [4]. The implementation of the filter uses the same logic used in the smoothing filter, i.e. the variance is calculated on a sliding window. For each value it identifies *n* samples (where *n* is the size of the window), then calculates the variance of this interval and repeats the procedure for all samples of the signal. What you will get is a curve whose values is function of the variance of each window.

Newton's Difference Quotient (First Derivative)

The Newton's Difference Quotient of a function is calculated as the ratio of the increment of the dependent variable (the increment of the values of the function) and the increment of the independent variable.
The well known formula of Newton's Difference Quotient is the following:

$$R_h(x) = \frac{f(x+h) - f(x)}{h} \qquad (Eq. 1.0)$$

The formula is used as a numerical approximation of the first derivative of the original function. Formally, the derivative of the function f at x is the limit of the Newton's Difference Quotient $R_h(x)$ when h tends to zero. Obviously a numerical approximation is necessary when the function f is not a formula, but a set of values.
The implementation calculates the quotient of each sample returning a function that closely approximates the first derivative of the signal.

Newton's Difference Quotient (Second Derivative)

Applying the difference quotient of the prevously described function it is possible to approximate the second derivative.

Appendix C - Features

Here is a detailed description of features used for the process of *feature extraction.*

Maximum/Minimum Values Starting and Ending Values

Usually an acquisition (a sample) lasts for less than a couple of seconds. Given a sample, we simply extract the maximum, the minimum value, the first and the last value of the given sample.

Skewness

Skewness is a statistical measure of the asymmetry of the probability distribution of a real-valued random variable. It is able to quantify if data are "distributed" on one side or the other of the arithmetic mean. Qualitatively, a negative (positive) skew indicates that the *tail* on the left (right) side of the probability density function is *longer* than the right (lieft) side and the bulk of the values including the median lie to the right (left) of the mean. A zero value indicates that the values are relatively evenly distributed on both sides of the mean, typically but not necessarily implying a symmetric distribution. The formula of *Skewness* is as follows:

$$\gamma_1 = \frac{\mu_3}{\sigma^3} \qquad (Eq.1.0.1.)$$

where μ_3 is the third moment of the mean μ, and σ is the standard deviation.

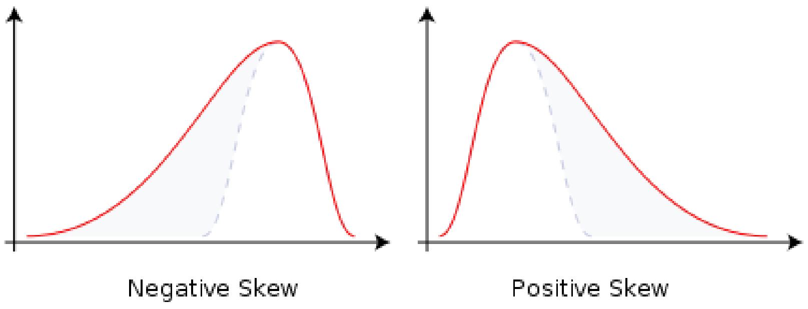


Fig. 16 - "Skewness" in a Gaussian distribution

Kurtosis

In probability theory and statistics, *kurtosis* (from the Greek word κυρτός) is a measure of the "peakedness" of the probability distribution of a real-valued random variable. Higher kurtosis means the variance is the result of infrequent extreme deviations from mean. The formula of *Skewness* is as follows:

$$\gamma_2 = \frac{\mu_4}{\sigma^4} \qquad (Eq. 1.0.2)$$

where μ_4 is the fourth moment about the mean μ, and σ is the standard deviation.

Zero Crossing Rate

Zero Crossing Rate (ZCR) is the number of times a signal crosses the zero value, normalized by the number of samples contained in the acquired set. The formula is as follows:

$$Z.C.R. = \frac{|\{x : x \in S \wedge x = 0\}|}{|S|} \qquad (Eq. 1.0.3)$$

Where S is the set of samples, |S| is the cardinality of S, and |{x : x ∈ S ∧ x = 0}| is the number of samples S equal to zero.

Mean Crossing Rate

The *Mean Crossing Rate* (MCR) represents the number of times a signal crosses the value of the arithmetic mean of its values, normalized with the number of samples contained in the acquired set. The formula is as follows:

$$M.C.R. = \frac{|\{x : x \in S \wedge x = \mu\}|}{|S|} \qquad (Eq. 1.0.4)$$

Where S is the set of samples, $|S|$ is the cardinality of S, and $|\{x : x \in S \wedge x = \mu\}|$ is the number of samples S equal to the mean μ.

Mean

The "Mean" *feature* is the arithmetic average of the window of data on which the analysis is carried out. The average formula is:

$$\mu = \frac{1}{n}\sum_{i=1}^{n} x_i \qquad (Eq. 1.0.5)$$

Unlike the mean and smoothing *filters* where a mobile windows is used to filter the signals, this feature is applied to the complete set of signal values.

Variance

The "Variance" is computed exploting the definition of statistical variance. Statistical Variance is a dispersion index indicating the statistical measure of variation from the average. The formula for the variance is as follows:

$$\sigma^2 = \frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x})^2 \quad (Eq. 1.0.6)$$

where $\bar{x}$ represents the average of the complete set of values, and σ is the standard deviation.
Unlike the variance filter, which calculates the variance on a window of values, this function calculates the extent of variance of all samples that are present in the acquisition.

Appendix D - Classifiers

Here is a detailed description of the used classifiers.

Similarity Classifiers

Rank Classifier

The Rank classifier assigns a degree of similarity summing up the feature values of the given vectors and sorting the results. In this case, the degree of similarity depends on the value of *FF* and *IVFF*. The formula used to calculate the similarity of every action is as follows:

$$ranking_j = \sum_{i=1}^{N} x_{i,j} \qquad (Eq. 1.1.1)$$

where $x_{i,j}$ is the *i-th* interval-feature hit by the query, of the *j-th* Action in the *Training Set* table.

The obtained $ranking_j$ values are ordered from greatest to smallest. The *j* action that has the highest $ranking_j$ value is the action that is considered more similar to the given *"query"*,. The function cost is linear in time, the main algorithmic costs of the Rank classifier is the cost of ordering, that prevails on the others (see Computational Costs, p.42).

Cosine Similarity Classifier

Cosine Similarity calculate the cosine of the angle between the query vector and each Actions class (each column vector) of the Training-Set table. The *j-th* Action that has a “smaller angle” is considered the most similar to the query [9].

The formula used to calculate the *Cosine Similarity* is the following:

$$\cos \theta_j = \frac{A_j \cdot B_j}{\|A\|_j \|B\|_j} \qquad (Eq. 1.2.1)$$

Hence:

$$\theta_j = \cos^{-1} \frac{A_j \cdot B_j}{\|A\|_j \|B\|_j} \qquad (Eq. 1.2.2)$$

Where $A \cdot B$ represent the dot product

$$A_j \cdot B_j = \sum_{i=1}^{N} x_{i,j} * y_i \qquad (Eq. 1.2.3)$$

$\|A\|, \ \|B\|$ are the magnitudes calculated as follows

$$\|A\|_{\ j} = \sqrt{\sum_{i=1}^{n} {x_{i,j}}^2} \qquad (Eq. 1.2.4)$$

$$\|B\|_{\ j} = \sqrt{\sum_{i=1}^{n} {y_i}^2} \qquad (Eq. 1.2.5)$$

where A and B generically represent two column vectors; A_j is the *j-th* colum vector of the *Training-Set* table and B is the *query vector*. B_j is the query vector that dipends on the *"n"* intervals problem that depends on A_j. Here $x_{i,j}$ is the *i-th interval-feature* of the *j-th* Action in the *Training Set* table, and y_i is the *i-th interval-feature* hit by the query.

We have to note that $x_{i,j}$, y_i are positive values given that $Ff_{i,j}$ and $IVFf_i$ are positive, hence the calculated similarity values are in the domain [0,1] .

The algorithm calculates similarity values of every actions of the *Training Set* to the given query and returns the results in ascending order.

Tanimoto Classifier

The *Tanimoto* classifier is a variant of the *Cosine Similarity* classifier and is an extension of the *Jaccard* coefficient that is usually related only to binary vectors.

The formula is as follows:

$$\cos\theta_j = \frac{A_j \cdot B_j}{\|A\|^2{}_j + \|B\|^2{}_j - A_j \cdot B_j} \qquad (Eq. 1.3.1)$$

Hence:

$$\theta_j = \cos^{-1}\frac{A_j \cdot B_j}{\|A\|^2{}_j + \|B\|^2{}_j - A_j \cdot B_j} \qquad (Eq. 1.3.2)$$

$A \cdot B$, $\|A\|^2$ e $\|B\|^2$ are calculated according to the equations Eq 1.2.3 , Eq 1.2.4 e Eq 1.2.5.
The algorithm calculate similarity values, and orders the actions in ascending order.

Note: Many similarity functions are defined on binary vectors; if used with real numbers they can loose metric properties. Therefore, a transformation of the formulas is necessary to be used by algorithms that operate with non-binary vectors.

For example, given the two vectors $X\ Y$, with $X_i \in \{0,1\}$ and $Y_i \in \{0,1\}\ \forall\ 1 \le i \le C$, where $C = |A| = |B|$, and N is the number of elements of X and Y. We can write the formula of *Jaccard* coefficient for binary vectors as following:

$$Jaccard = \frac{a}{a+b+c} \qquad (Eq. 1.3.3)$$

Where a is the number of times the pair (X_i, Y_i) = (1,1) , b is the number of times the pair (X_i, Y_i) = (0,1), and c is the number of times the pair (X_i, Y_i) = (1,0).

We can generalize the above expression to non binary cases redifining a, b, c terms of the equation as follows:

$$a = A \cdot B \qquad (Eq\ \ 1.3.4)$$
$$b = \|A\|^2 - A \cdot B \qquad (Eq\ \ 1.3.5)$$
$$c = \|B\|^2 - A \cdot B \qquad (Eq\ \ 1.3.6)$$

Using the above equations, is possibile to trasform many similarity algorithm from binary to non-binary vectors cases (see [16]).

Simpson Classifier

Simpson classifier measure the similarity between two vectors in an *N*-dimensional space. The similarity is calculated between the query vector and each vector column of the table of the Training-Set. The formula used to calculate *Simpson* classifier is as follows:

$$Simpson = \frac{a}{min(a+b, a+c)} \qquad (Eq\ 1.4.1)$$

Using the transform equations $Eq\ 1.3.4,\ Eq\ 1.3.5,\ Eq\ 1.3.6$ the formula becomes:

$$Simpson_j = \frac{A_j \cdot B_j}{min\left(\|A\|^2{}_j, \|B\|^2{}_j\right)} \qquad (Eq\ 1.4.2)$$

Given $Ff_{i,j} \in (0,1]$ and $IVFf_i \in [0, \log|D|]$, then $Ff_{i,j} * IVFf_i < IVFf_i \quad \forall Ff_{i,j},\ \forall\ IVFf_i$
where $1 \leq i \leq N$ e $1 \leq j \leq M$, and *N* is the number of features, *M* is the number of actions.
It follows:

$$\|A\|^2{}_j < \|B\|^2{}_j \quad \forall\ j\ \ 1 \leq j \leq M \qquad (Eq\ 1.4.3)$$

Hence the *Simpson* classifier can be simplified as follows:

$$Simpson_j = \frac{A_j \cdot B_j}{\|A\|^2{}_j} \qquad (Eq\ 1.4.2)$$

The algorithm calculates the similarity using Simpson function, and returns the results in descending order.

Braun-Blanquet Classifier

The formula of *BraunBlanquet* classifier is as follows:

$$BraunBlanquet = \frac{a}{max(a+b, a+c)} \qquad (Eq\ 1.5.1)$$

Using the usual transform equations the formula becomes:

$$BraunBlanquet_j = \frac{A_j \cdot B_j}{max\left(\|A\|^2{}_j, \|B\|^2{}_j\right)} \qquad (Eq\ 1.5.2)$$

According to Eq. 1.4.3 and simplifying:

$$BraunBlanquet_j = \frac{A_j \cdot B_j}{\|B\|^2{}_j} \qquad (Eq\ 1.5.2)$$

The algorithm calculates the *BraunBlanquet* similarity and returns the results in descending order.

Kulczynski 1 Classifier

The formula of *Kulczynski 1* classifier is as follows:

$$Kulczynski\ 1 = \frac{a}{b+c} \qquad (Eq\ 1.6.1)$$

Using the transform equations the formula becomes:

$$Kulczynski\ 1_j = \frac{A_j \cdot B_j}{\|A\|^2{}_j + \|B\|^2{}_j - 2\ A_j \cdot B_j} \qquad (Eq\ 1.6.2)$$

The algorithm calculates the *Kulczynski 1* similarity and returns the results in descending order.

Kulczynski 2 Classifier

The formula of Kulczynski 2 classifier is as follows:

$$Kulczynski\ 2 = \frac{1}{2} * \left[\left(\frac{a}{a+b}\right) + \left(\frac{a}{a+c}\right)\right] \qquad (Eq\ 1.7.1)$$

Using the transform equations the formula becomes:

$$Kulczynski\ 2_j = \frac{1}{2} * \left[\left(\frac{A_j \cdot B_j}{\|A\|^2{}_j}\right) + \left(\frac{A_j \cdot B_j}{\|B\|^2{}_j}\right)\right] \qquad (Eq\ 1.7.2)$$

The algorithm calculates the *Kulczynski 2* similarity and returns the results in descending order.

Dice-Sorensen Classifier

The formula of *Dice-Sorensen* classifier is as follows:

$$Dice\_Sorensen = \frac{2a}{b + c} \qquad (Eq\ 1.8.1)$$

using the transform equations the formula becomes:

$$Dice\_Sorensen_j = \frac{2\ \ A_j \cdot B_j}{\|A\|^2{}_j + \|B\|^2{}_j} \qquad (Eq\ 1.8.2)$$

The algorithm calculates the *Dice-Sorensen* similarity and returns the results in descending order.

Otsuka Classifier

The formula of *Otsuka* classifier is as follows:

$$\text{Otsuka} = \frac{a}{\sqrt{(a + b) * (a + c)}} \qquad (Eq\ 1.9.1)$$

using the transform equations the formula becomes:

$$\text{Otsuka}_j = \frac{A_j \cdot B_j}{\sqrt{\left(\|A\|^2{}_j * \|B\|^2{}_j\right)}} \qquad (Eq\ 1.9.2)$$

The algorithm calculates the *Otsuka* similarity and returns the results in descending order.

Mountford Classifier

The formula of *Mountford* classifier is as follows:

$$\text{Mountford} = \frac{2a}{2\,bc + ab + ac} \qquad (Eq\ 1.10.1)$$

using the transform equations the formula becomes:

$$\text{Mountford}_j = \frac{2\,A_j \cdot B_j}{2\left(\|A\|^2{}_j * \|B\|^2{}_j\right) - \left(A_j \cdot B_j\right)^2} \qquad (Eq\ 1.10.2)$$

The algorithm calculates the *Mountford* similarity and returns the results in descending order.

N-Classifier

This classifier measures the similarity between the *query* vector and the *j-th* vector of *Training Set*, counting how many intervals the query and action have in common. Given a *query*, the algorithm sum the number of interval set to *"n"* (null), for every actions of *Training Set* and then sorts the actions from the smallest to largest values. The action with the smallest value is the more similar. The higher the value, and the more an action is dissimilar from the query, conversely, the smaller the value, and the more the action is similar to the query. Roughly speaking this classifier take into account how much subspace the query have in common with the actions.

NWeight-Classifier

The *NweightClassifier* is a variant of the above described *NClassifier*. For each interval of the training set, this classifier calculates the "horizontal frequency" of the null "n" interval (*nFrequency*), interval by interval. For each Action of the *Training* Set, given a query, the algorithm counts how many terms set to "n" are present in a given Action and instead of take an increment of 1, uses an increment of *nFrequency* value:

$$nFrequency_i = \frac{nValues_i}{|D|} \qquad (Eq\ 1.11.1)$$

where $nValues_i$ is the number of occurrences of the n terms in the *interval-feature* i, while $|D|$ is the cardinality dictionary.

The formula used to calculate *NWeightClassifier* is as follows:

$$NWeightClassifier_j = \sum_{i=1}^{N} nFrequency_i \qquad if\ \ Ff_{i,j} = n \qquad (Eq\ 1.11.2)$$

The algorithm calculates the *NWeightClassifier* similarity and returns the results in ascending order.

Entropy Classifier

In information theory, Entropy is defined as measure of the uncertainty of random variable associated with a probability distribution $X = \{x_{1,}x_{2,}x_3, \dots . x_d\}$ having observed a set of symbols $S = \{s_{1,}s_{2,}s_3, \dots . s_d\},\ \ where\ d = |X| = |S|$.

Here the distribituion of probability is $P_j = \{Ff_{1,j} * IVFf_1, Ff_{2,j} * IVFf_2, \dots, Ff_{N,j} * IVFf_N\}$ having observed the *features* $F_j = \{f_1, f_2, \dots, f_N\}$, where *N* is the number of given features.

Hence, the used formula of *Entropy* is as follows:

$$Entropy_j = \sum_{i=1}^{N} x_{i,j} * \log \frac{1}{x_{i,j}} \qquad (Eq\ 1.12.1)$$

When $Ff_{i,j} = n$, we sum to $Entropy_j$ the quantity 0.

Given the query, agorithm calculate the *Entropy* of every action of the *Traing Set* table then returns the result in descending order.

Distance Classifiers

These classifiers use the distance in *N*-dimensional space of *features* to measure the similarity between the *query* vector and a specific vector column of *Training Set*. The algorithms calculate the distance between the *query* vector with each column vector of the table of the *Training-Set* and the the action with the shortest distance is considered the most probable action to be performed.
All the algorithms calculate the distance of the query with all the vector of the *Training Set* then order the results in ascending order (smaller to greater).

Euclidean Distance Classifier

The formula of *euclidean distance* is as follows

$$EuclideanDistance_j = \sqrt{\sum_{i=1}^{N} \left(x_{i,j} - y_i\right)^2} \quad (Eq. 2.1.1)$$

Manhattan Distance Classifier

The formula of *Manhattan* distance is as follows:

$$Manhattan_j = \sum_{i=1}^{N} abs(x_{i,j} - y_i) \quad (Eq. 2.2.1)$$

Camberra Classifier

The formula of *Camberra* distance is as follows:

$$Camberra_j = \sum_{i=1}^{N} \frac{abs(x_{i,j} - y_i)}{x_{i,j} - y_i} \quad (Eq. 2.3.1)$$

Where $x_{i,j} - y_i$ must not be zero.

Chi-square Distance Classifier

The formula of *Chi-square* distance is as follows:

$$Chi\_square_j = \sum_{i=1}^{N} \frac{(x_{i,j} - y_i)^2}{(x_{i,j} - y_i)} \quad (Eq. 2.4.1)$$

Where $x_{i,j} - y_i$ must not be zero.

Hellinger-square Distance Classifier

The used formula of *Hellinger-square* distance is as follows:

$$Hellinger\_square_j = \sum_{i=1}^{N} \left(\sqrt{x_{i,j}} - \sqrt{y_i}\right)^2 \quad (Eq. 2.5.1)$$

Hamming Distance Classifier

The *Hamming* distance for binary values has been defined as follows:

$$Hamming = \frac{b + c}{C} \quad (Eq. 2.6.1)$$

Using the transform equations $Eq. 1.3.4$, $Eq. 1.3.5$, $Eq. 1.3.6$ the formula becomes:

$$Hamming_j = \frac{\|A\|^2{}_j + \|B\|^2{}_j - A_j \cdot B_j}{N} \quad (Eq. 2.6.2)$$

Where *N* is the number of analyzed features and is used as a normalization factor for the calculated distance. *N* is not necessary to classify, as only relative ordering is important to say which action is closest to the query.

Bray-Curtis Classifier

La formula of *Bray-Curtis* is as follows:

$$Bray\_Curtis_j = \frac{\sum_{i=1}^{N} abs(x_{i,j} - y_i)}{\sum_{i=1}^{N} x_{i,j} + y_i} \quad (Eq. 2.7.1)$$

Min-variance Classifier

This classifier has been introduced as an alternative method to caluclate the distance between two vectors in our context. The formula of the *min variance* is as follows:

$$Min\_variance_j = abs\left(\sum_{i=1}^{N}\left(x_{i,j} - \sum_{i=1}^{N}\frac{x_{i,j}}{N}\right)^2 - \sum_{i=1}^{N}\left(y_i - \sum_{i=1}^{N}\frac{y_i}{N}\right)^2\right) \quad (Eq. 2.8.1)$$

Where $x_{i,j}$ is the *i-th interval-feature* of the *j-th* Action in the *Training Set* table, and y_i is the *i-th interval-feature* hit by the query, and where $1 \leq i \leq N$ e $1 \leq j \leq M$ vary on the number of *N features* and *M* Actions (classes) present in the vocabulary.

Given the j-*th* Action A_j, with feature values $x_{i,j}$ and the B_j query with feature values y_i, *the min variance* takes into account the variance of the features values, in respect to the "vertical average", i.e. the mean of the *FFxIVFF* of the given action and query. Then these values are respectively subtracted.

Correlation Classifier

Pearson Correlation Classifier

Only one correlation classifier has been used, the *Pearson correlation* to test if relevant differences arise. This classifier measures the correlation between the *query* vector and each column vector (Action class) of the *Training Set* table. The *Pearson correlation* has values in the interval [1, -1], and in particular:

- If the *query* vector and the action class of the *Training Set* are *directly correlated*, the *Pearson correlation* value is *1*
- If the *query* vector and the action class of the *Training Set* are *uncorrelated*, the *Pearson correlation* value is *0*. The inverse implication is not generally true.
- If the *query* vector and the action class of the *Training Set* are *inversely correlated*, the *Pearson correlation* value is -*1*

The formula of *Pearson correlation* is as follows:

$$Pearson\_Correlation_j = \frac{\sum_{i=1}^{N}\left(x_{i,j} - \sum_{i=1}^{N}\frac{x_{i,j}}{N}\right) * \left(y_i - \sum_{i=1}^{N}\frac{y_i}{N}\right)}{\sqrt{\frac{\sum_{i=1}^{N}\left(x_{i,j} - \sum_{i=1}^{N}\frac{x_{i,j}}{N}\right)^2}{N}} * \sqrt{\frac{\sum_{i=1}^{N}\left(y_i - \sum_{i=1}^{N}\frac{y_i}{N}\right)^2}{N}}} \qquad (Eq. 3.1.1)$$

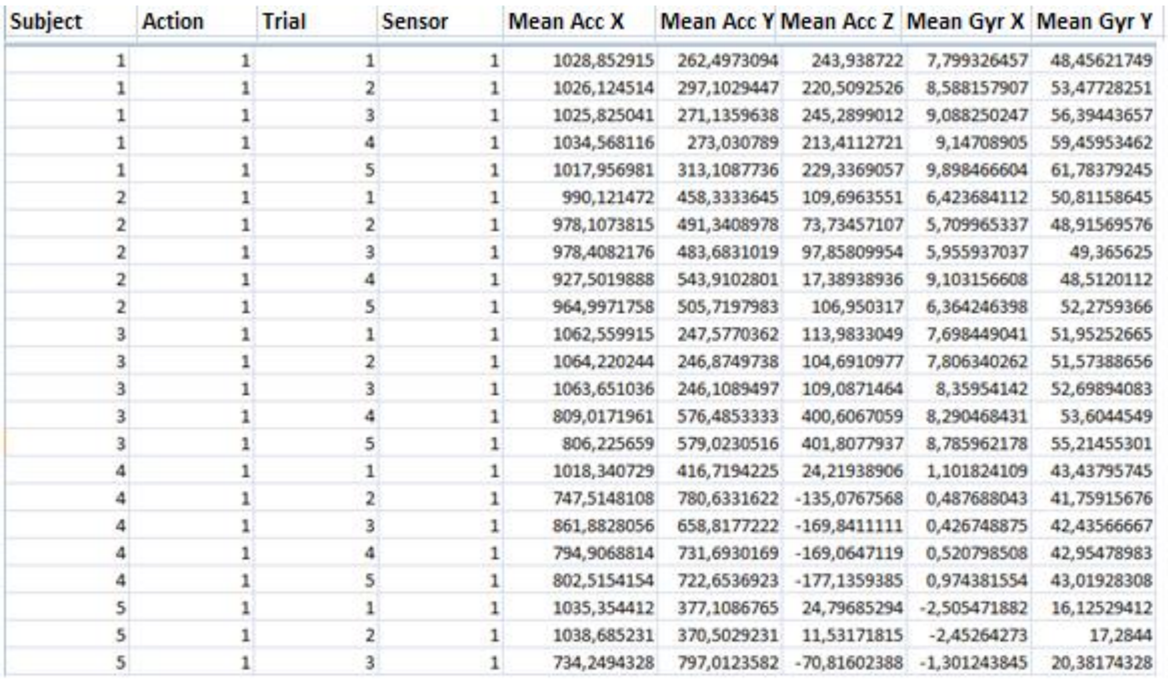
The algorithm return the action in descending order (greater to smaller).

The action directly correlated to *query* vector with the best value, is considered the most probable executed action.

Appendix E - Calibration

Calibration of WARD Data

As suggested by the U.C. Berkely researchers, the WARD database was Calibrated and Normalized (i.e. mapped to to correct useful range of values). Calibration requires to“center the data” on a general average (mean shifting).



Subject	Action	Trial	Sensor	Mean Acc X	Mean Acc Y	Mean Acc Z	Mean Gyr X	Mean Gyr Y
1	1	1	1	1028,852915	262,4973094	243,938722	7,799326457	48,45621749
1	1	2	1	1026,124514	297,1029447	220,5092526	8,588157907	53,47728251
1	1	3	1	1025,825041	271,1359638	245,2899012	9,088250247	56,39443657
1	1	4	1	1034,568116	273,030789	213,4112721	9,14708905	59,45953462
1	1	5	1	1017,956981	313,1087736	229,3369057	9,898466604	61,78379245
2	1	1	1	990,121472	458,3333645	109,6963551	6,423684112	50,81158645
2	1	2	1	978,1073815	491,3408978	73,73457107	5,709965337	48,91569576
2	1	3	1	978,4082176	483,6831019	97,85809954	5,955937037	49,365625
2	1	4	1	927,5019888	543,9102801	17,38938936	9,103156608	48,5120112
2	1	5	1	964,9971758	505,7197983	106,950317	6,364246398	52,2759366
3	1	1	1	1062,559915	247,5770362	113,9833049	7,698449041	51,95252665
3	1	2	1	1064,220244	246,8749738	104,6910977	7,806340262	51,57388656
3	1	3	1	1063,651036	246,1089497	109,0871464	8,35954142	52,69894083
3	1	4	1	809,0171961	576,4853333	400,6067059	8,290468431	53,6044549
3	1	5	1	806,225659	579,0230516	401,8077937	8,785962178	55,21455301
4	1	1	1	1018,340729	416,7194225	24,21938906	1,101824109	43,43795745
4	1	2	1	747,5148108	780,6331622	-135,0767568	0,487688043	41,75915676
4	1	3	1	861,8828056	658,8177222	-169,8411111	0,426748875	42,43566667
4	1	4	1	794,9068814	731,6930169	-169,0647119	0,520798508	42,95478983
4	1	5	1	802,5154154	722,6536923	-177,1359385	0,974381554	43,01928308
5	1	1	1	1035,354412	377,1086765	24,79685294	-2,505471882	16,12529412
5	1	2	1	1038,685231	370,5029231	11,53171815	-2,45264273	17,2844
5	1	3	1	734,2494328	797,0123582	-70,81602388	-1,301243845	20,38174328

Fig. 54 – Averages of accelerometer and magnetometer data component by component.

In Fig. 54 we can see that in every sensor the averages of each component (Accelerometer X, Accelerometer Y, Accelerometer Z, Gyroscope X, Gyroscope Y) have different values.

To align the averages, given the action *a*, the sensor *s*, and component *c* the general average is calculated, then is calculated the difference between the obtained average and the average given component by component. Finally, the difference is added to each sample in order to shift the samples of the right quantity.

In particular:

1. For every sensor s , action a and component c, the general mean $GenMean_{c,s,a}$ is calculated as follows:

$$GenMean_{s,a,c} = \frac{\sum_{S=1}^{NumSubject} \sum_{t=1}^{NunTrial} Mean_{S,a,t,s,c}}{(numSubject * numTrial)}$$

Where $Mean_{c,S,a,t,s}$ is the avarage fixing the subject S, action a, trial t, sensor s and componente c, and where $numSubject$ and $numTrial$ are respectively the number of subjects and the number of trials.

2. For every subject S, action a, trial t, sensor s and componente c the shifting factor $\Delta_{c,S,a,t,s}$ is calculated as follows:

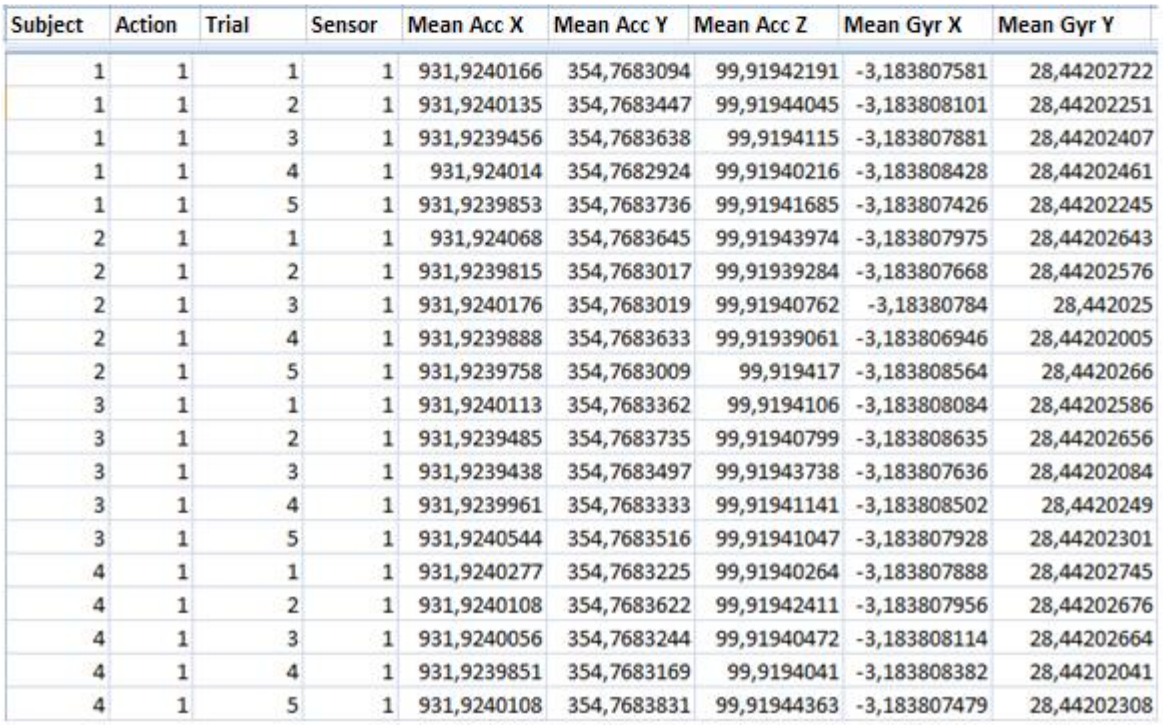
$$\Delta_{S,a,t,s,c} = Gen_{Means,a,c} - Mean_{S,a,t,s,c}$$

3.The values of every component is shifted using the following formula:

$$NewValue_{S,a,t,s,c,i} = Value_{S,a,t,s,c,i} + \Delta_{S,a,t,s,c}$$

Where $Value_{S,a,t,s,c,i}$ is the single i data sample given the subject S , action a, trial t, sensor s of the component c.

In Fig. 55 we can see an example of the results of the operation.



Subject	Action	Trial	Sensor	Mean Acc X	Mean Acc Y	Mean Acc Z	Mean Gyr X	Mean Gyr Y
1	1	1	1	931,9240166	354,7683094	99,91942191	-3,183807581	28,44202722
1	1	2	1	931,9240135	354,7683447	99,91944045	-3,183808101	28,44202251
1	1	3	1	931,9239456	354,7683638	99,9194115	-3,183807881	28,44202407
1	1	4	1	931,924014	354,7682924	99,91940216	-3,183808428	28,44202461
1	1	5	1	931,9239853	354,7683736	99,91941685	-3,183807426	28,44202245
2	1	1	1	931,924068	354,7683645	99,91943974	-3,183807975	28,44202643
2	1	2	1	931,9239815	354,7683017	99,91939284	-3,183807668	28,44202576
2	1	3	1	931,9240176	354,7683019	99,91940762	-3,18380784	28,442025
2	1	4	1	931,9239888	354,7683633	99,91939061	-3,183806946	28,44202005
2	1	5	1	931,9239758	354,7683009	99,919417	-3,183808564	28,4420266
3	1	1	1	931,9240113	354,7683362	99,9194106	-3,183808084	28,44202586
3	1	2	1	931,9239485	354,7683735	99,91940799	-3,183808635	28,44202656
3	1	3	1	931,9239438	354,7683497	99,91943738	-3,183807636	28,44202084
3	1	4	1	931,9239961	354,7683333	99,91941141	-3,183808502	28,4420249
3	1	5	1	931,9240544	354,7683516	99,91941047	-3,183807928	28,44202301
4	1	1	1	931,9240277	354,7683225	99,91940264	-3,183807888	28,44202745
4	1	2	1	931,9240108	354,7683622	99,91942411	-3,183807956	28,44202676
4	1	3	1	931,9240056	354,7683244	99,91940472	-3,183808114	28,44202664
4	1	4	1	931,9239851	354,7683169	99,9194041	-3,183808382	28,44202041
4	1	5	1	931,9240108	354,7683831	99,91944363	-3,183807479	28,44202308

Fig. 55 – Averages of accelerometer and magnetometer data after the shifting component by component

Normalization of the Calibrated Data.

Normalization (mapping of the data to a correct codomain of values) of WARD data is necessary as we need the data to be within the correct range of values before to start the feature extraction process. If this is not done we would have a discretization thta could get incorrect results. The normalization and mapping of data is different for accelerometers and gyroscopes. This is consistent since the two devices works differently.

Accelerometer

For each subject, action, trial and sensor, the average of the magnitude of accelerometer is calculated. Although the X, Y and Z averages are all aligned to the accelerometer sensor-action, the average of the magnitude may be not 9.81 m/sec^2, as the formual of magnitude is a non-linear operation.
Data are normalized as follows:

1. For every action a and sensor s the action-sensor mean of the average of magnitude of accelerometer is calculated as follows:

$$MeanActionSensorMagAcc_{a,s} = \frac{\sum_{S=1}^{NumSubject}\sum_{t=1}^{NumTrial} MeanModAccCal_{S,a,t,s}}{(numSubject * NumTrial)}$$

   where $MeanModAccCal_{S,a,t,s}$ is the average of the magnitude of accelerometers calculate from the calibrated data given the subject S, action a, trial t e sensore s, and where $numSubject$ e $numTrial$ sono rispettivamente il numero dei soggetti e il numero dei trial.

2. Given the subject S, action a, trial t e sensor s, to every sample i of the component c of accelerometer has been applied the formula:

$$NewValue_{S,a,t,s,c,i} = \frac{9{,}81}{MeanActionSensorMagAcc_{a,s}} * Value_{S,a,t,s,c,i}$$

   Where $Value_{S,a,t,s,c,i}$ is the single *i* data sample given the subject *S* , action *a*, trial *t* and sensor *s*. of the component *c*.

.

Gyroscope

The normalization (mapping) of calibrated data of the Gyroscope is done in the following way.
For each sample of the X and Y gyroscope, given the subject *S*, action *a*, trial *t*, sensor *s*, the following formula is applied:

$$NewGirXValue_{S,a,t,s,Xi} = \left(GyrYValue_{S,a,t,s,Yi} + MeanGirYCal_{a,s}\right) * \frac{\pi/2}{200}$$

$$NewGirYValue_{S,a,t,s,Yi} = \left(GyrXValue_{S,a,t,s,Xi} + MeanGirXCal_{a,s}\right) * \frac{\pi/2}{200}$$

Where $MeanGirXCal_{a,s}$ and $MeanGirYCal_{a,s}$ are respectively the *X* axis and *Y* axis average of the calibrated data given subject *S*, action *a*, while $GyrYValue_{S,a,t,s,Yi}$ e $GyirXValue_{S,a,t,s,Xi}$ are the *i-th* data respectively of *X* and *Y* axes of the given acquisition, and 200 is a normalization parameter heuristically choosen.

References

[1] A.Mileo,S.Pinardi, R.Bisian. Movement recognition using context: a lexical approach based on coherence, *Proceedings of MRC2010*, Lisbon, Portugal, August 2010

[2] S.Pinardi and R.Bisiani. Movement recognition with intelligent multisensor analysis, a lexical approach, *Proceedings of AITAMi'10*, Kuala Lumpur, Malaysia, July 2010

[3] C.Yang Y.Hsu. A Review of accelerometry-based wearable motion detectors for physical activity monitoring. *Sensors* 10, 10, 8: 7772-7788; doi:10.3390/s100807772, 2010

[4] R. Bisiani, D. Merico, A. Mileo, S. Pinardi. A logical approach to home healthcare with intelligent sensor-network support. *The Computer Journal ;* doi: 10.1093/comjnl/bxn074, 2009.

[5] A.Y. Yang, R. Jafari, S.S. Sastry, R. Bajcsy. Distributed recognition of human actions using wearable motion sensor networks, *Journal of Ambient Intelligence and Smart Environments*, 2009.

[6] J. Cameron, J.Lasenby, Estimating human skeleton parameters and configuration in real-time from markered optical motion capture, *Proceedings of the 5th international conference on Articulated Motion and Deformable Objects*, Port d'Andratx, Mallorca, Spain, pp. 92-101, 2008

[7] Z. Gan, M. Jiang. Articulated body tracking by immune particle filter. *Proceedings ICWL 2006*: 93-104. SEAL 2006 853-857, 2008.

[8] M. Hahn, L. Krüger, C. Wöhler. 3D action recognition and long-term prediction of human motion. *Proceedings of ICVS 2008*, 23-32, 2008

[9] J. Han, M. Kamber. Data Mining: Concept and Techniques, *Morgan Kaufman 2nd ed.*, 2008 ISBN: 978-1-55860-901-3.

[10] M.Marin-Perianu, C. Lombriser, O. Amft, P. J. M. Havinga, G. Tröster, Distributed activity recognition with Fuzzy-enabled Wireless Sensor Networks. *Proceedings of the 4th IEEE international conference on Distributed Computing in Sensor Systems*, Santorini, Greece, volume 5067/June 2008, 296-313, 2008

[11] R.Okada, B.Stenger. A single camera motion capture system for human-computer interaction, *ICICE(E91-D)*, No. 7, pp. 1855-1862, 2008

[12] S. Ohgi, S. Morita, K.K. Loo, C. Mizuike, Time series analysis of spontaneous upper-extremity movements of premature infants with brain injuries, *PHYS. THER.*, Vol. 88, No. 9, September 2008, pp. 1022-1033, 2008

[13] V.Osmani, S. Balasubramaniam, D. Botvich. Human activity recognition in pervasive health-care: supporting efficient remote collaboration. J. *Netw. Comput. Appl. 31*, 4 (Nov. 2008), 628-655, 2008.

[14] K. Rieck, P. Laskov. Linear-Time Computation of similarity measures for sequential data. *Journal of Machine Learning Research.* Vol. 9, Number Jan, pp. 23-48, 2008.

[15] C.Wan, B.Yuan, Z.Miao. Markerless human body motion capture using markov random field and dynamic graph cuts, *Visual Computer* (24), No. 5, May 2008

[16] G. Castellano, S.D. Villalba, A.Camurri. Recognizing human emotions from body movement and gesture dynamics, *Proc. ACII07, Lecture Notes In Computer Science*, vol. 4738. Springer-Verlag, Berlin, Heidelberg, 2007

[17] G. Guerra-Filho, Y.Aloimonos. A language for human action, *IEEE Computer Magazine*, 40:60–69, 2007

[18] G. Guerra-Filho, Y. Aloimonos. A sensory-motor linguistic framework for human activity understanding, *Doctoral Thesis, University of Maryland at College Park*, 2007.

[19] A. Kleinsmith, N. Bianchi-Berthouze. Recognizing affective dimensions from body posture, *Proc. of ACII07, Lecture Notes In Computer Science*, vol. 4738. Springer-Verlag, Berlin, Heidelberg, 48-58, 2007

[20] H. Wang, H. Lenz, A. Szabo, J. Bamberger, U. D. Hanebeck. WLAN-Based pedestrian tracking using particle filters and Low-Cost MEMS Sensors, *Proceedings of 4th Workshop on Positioning, Navigation and Communication* 2007 *(WPNC 2007)*, Hannover, Germany, 2007

[21] T. Choudhury, M. Philipose, D. Wyatt, J. Lester. Towards activity database: using sensor and statistical models to summarize people's lives, *IEEE Data Engineering Bulletin*, 2006.

[22] S. Knoop, S. Vacek, R. Dillmann. Sensor fusion for 3d human body tracking with an articulated 3d body model, *Proceedings of the 2006 IEEE International Conference on Robotics and Automation (ICRA)*, Orlando, Florida, 2006.

[23] J. Lester, T. Choudhury, G. Borriello. A practical approach to recognizing physical activities, *Proceedings of Pervasive 2006*, LNCS 3968, pp. 1 – 16, 2006.

[24] M.S. Ryoo, J.K. Aggarwal, Recognition of composite human activities through context-free grammar based representation. *Computer & Vision Research Center / Department of ECE University of Texas at Austin (CVPR'06)*, pp. 1709-1718, 2006

[25] H.Sushant, S. Rahul, B. Waylon, G.Borriello, R. Sumit. Exploiting mobility for energy efficient data collection in wireless sensor networks. *Mobile networks and applications*, Vol. 11, No. 3. (June 2006), pp. 327-339, 2006

[26] T. Choudhury, N. Kern, G. Borriello, B. Hannaford, J. Lester. A hybrid discriminative/generative approach for modeling human activities. *Proceedings of the Nineteenth IJCAI*, pp. 766 - 722, Edinburgh, Scotland, 2005.

[27] E. F. Desserée, L. R. Legrand. First results of a complete marker-free methodology for human gait analysis. *Proceedings of the 2005 IEEE Engineering in Medicine and Biology* , 27th Annual Conference Shanghai, China, September 1-4, 2005

[28] G .Guerra-Filho, Y.Aloimonos. Discovering a language for human activity. *AAAI Workshop on Anticipation in Cognitive Systems*, October, 2005

[29] N. Ravi, N. Dandekar, P. Mysore, and M. L. Littman. Activity recognition from accelerometer data, *American Association for Artificial Intelligence* (IAAI05), pp.1541-1546, 2005

[30] A.M. Sabatini. Quaternion-based strap-down integration method for applications of inertial sensing to gait analysis. 43(1):94-101, Jan 2005.

[31] Y. Tao, H. Hu, and H. Zhou. Integration of vision and inertial sensors for home-based rehabilitation. *IEEE International Conference on Robotics and Automation*, 2005.

[32] D. Wilson, S. Consolvo, K. Fishkin, and M. Philipose, In-home assessment of the activities of daily living of the elderly. *CHI2005: Workshops - HCI Challenges in Health Assessment*, 2005.

[33] L. Bao, S. Intille. Activity recognition from user-annotated acceleration data. *Proceedings of the International Conference on Pervasive Computing*, pp. 1-17, 2004.

[34] L.Snidaro, G.Luca, F. Ruixin Niu, P.K. Varshney. Sensor fusion for video surveillance. *7th Int. Conf. on Information Fusion* , 2004.

[35] S.Park, J.K. Aggarwal. Semantic-level understanding of human actions and interactions using event hierarchy. *IEEE Workshop on Articulated and nonrigid otion (ANM2004)*, Whashington, DC, USA, 2004

[36] H. Zhou, H. Hu. A survey - human movement tracking and stroke rehabilitation. *Technical Report CSM*-420, ISSN 1744 - 8050, University of Essex, UK, 2004.

[37] L. Bao, Physical activity recognition from acceleration data under semi-naturalistic conditions. *MIT master thesis*, 2003.

[38] L. Wang, W.M. Hu and T.N. Tan, Recent developments in human motion analysis. *Pattern Recognition*, vol. 36, no. 3, pp. 585-601, 2003.

[39] G.S. Chambers, S.Venkatesh, G.A.W.West, H.H.Bui. Hierarchical recognition of intentional human gestures for sports video annotation. *International Conference on Pattern Recognition*. vol.2 pp. 1082–5, Quebec City, 2002.

[40] A.M. Glenberg, M.P. Kascha. Grounding language in action. *Psychonomic Bulletin & Review*, 9, 558-565, 2002.

[41] S. Lee, K. Mase, Activity and location recognition using wearable, *IEEE Pervasive Computing*, 2002. Vol. 1 (2002), pp. 24-32, 2002

[42] T.B. Moeslund, E. Granum. A Survey of computer vision-based human motion capture. *Computer Vision and Image Understanding*, vol. 1, 3, pp. 24 – 32, 2002, doi: 10.1109/MPRV.2002.1037719

[43] C. Sminchisescu. Estimation Algorithms for ambiguous visual models--three-dimensional human modeling and motion reconstruction in monocular video Sequences. *Ph.D. thesis, Institute National Politechnique de Grenoble (INRIA)*, 2002.

[44] K. Mase, S.-W. Lee (2001). Recognition of walking behaviors for pedestrian navigation. *Proceedings of 2001 IEEE Conference on Control Applications*, (CCA01) p. 1152–5. Mexico City, 2001

[45] J. Mantyjarvi, J. Himberg, T. Seppanen. Recognizing human motion with multiple acceleration sensors. *Transactions on Systems, Man, and Cybernetics*, 2001 IEEE International Conference, Vol. 2 (2001), pp. 747-752, 2001

[46] T.B. Moeslund, E. Granum. A Survey of Computer vision-based human motion capture. *Computer Vision and Image Understanding*, 2001

[47] J. Deutscher, A. Blake, and I. Reid. Articulated body motion capture by annealed particle filtering. *IProc. Conf. Computer Vision and Pattern Recognition*, vol. 2, pp. 1144-1149, 2000.

[48] C. Randell, H. Muller, Context awareness by analyzing accelerometer data. *In: MacIntyre, B., and Iannucci, B., eds.. The Fourth International Symposium on Wearable Computers*, pp.175– 176. IEEE Computer Society, 2000.

[49] C.Randell, H.Muller. Context awareness by analyzing accelerometer data. *The Fourth International Symposium on Wearable Computers*, pp.175-176, 2000

[50] K. Van-Laerhoven O. Cakmakci. What shall we teach our pants? *The Fourth International Symposium on Wearable Computers*, pp. 77–83, 2000

[51] K.Amian, Ph. Robert, E.E. Buchser, B. Rutschmann, D. Haoyz, M.Deparion, Physical activity monitoring based on accelerometry, validation ad comparison with video observation. *Medical and biological engineering and computing,* Vol.7 n., 304-308 (1999), doi: 10.1007/BF02513304, 1999

[52] F. Foerster, M. Smeja; J. Fahrenberg,. M. S. Detection of posture and motion by accelerometry: a validation in ambulatory monitoring. *Computers in Human Behavior* 15(5), 571—583, 1999

[53] D. Gavrila. The visual analysis of human Movement: a survey. *Computer Vision and Image Understanding*, 73(1):82-98,. 1999.

[54] G. Welch. Hybrid self-tracker: an inertial/optical hybrid three-dimensional tracking system. *Tech. report TR95-048*, pp. 52-61. Univ. of North Carolina at Chapel Hill, 1995

[55] Y. Bar-Shalom ,T. B. Fortman. Tracking and data association. *Academic Press, New York*, 1988.

[57] S. Katz. Assessing self-maintenance: activities of daily living, mobility, and instrumental activities of daily living. *Journal of the American Geriatrics Society*, vol. 31, no. 12, pp. 721–726, 1983.

[58] H. Becquerel. Sur le radiation invisible émises par le corps phosphorescent. *L'Académie des Sciences*, *Paris*, *Comptes Rendus*, vol. 122: 501-502,24 Feb. 1896.

[59] www.xsens.com; visited October 2010

[60] www.charndyn.com; visted October 2010