Urbano Miguel Gonçalves Nunes

Keyframe Based Human Activity Recognition Using Random Forests

Coimbra, September 2016

Universidade de Coimbra

• U C •

FCTUC FACULDADE DE CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE DE COIMBRA

Keyframe Based Human Activity Recognition Using Random Forests

Urbano Miguel Gonçalves Nunes

Coimbra, September 2016

Keyframe Based Human Activity Recognition Using Random Forests

Supervisor:

Professor Doutor Paulo José Monteiro Peixoto

Jury:

Professor Doutor Carlos Alberto Henggeler de Carvalho Antunes

Professor Doutor Paulo José Monteiro Peixoto

Professor Doutor Rui Alexandre de Matos Araújo

Dissertation submitted in partial fulfillment for the degree of Master of Science in Electrical and Computer Engineering.

Coimbra, September 2016

"Science is the belief in the ignorance of experts."

— Richard Feynman

Acknowledgements

With the development and conclusion of this dissertation, great years have come to an end, which grant me many memorable moments of joy, but also lots of sacrifice. It is time to be grateful for all people whom I have come across and made this an easier and more pleasant journey.

A very distinctive acknowledgement to my supervisor, Prof. Paulo Peixoto. His inspirational ideas, support and motivational guidance brought me excitement to keep working with increasingly enthusiasm throughout this period. I am also grateful to Prof. Diego Faria, whose remarkable insights and knowledge in the studied field helped me overcome many adversities. I would also like to thank ISR and its personnel for hosting me, grating all the necessary conditions for the accomplishment of this work. To my laboratory colleagues, who shared, in a way or another, their practical experience, I also express my gratitude. I would also like to thank my closest family: my parents, sister and brother, who have unconditionally stand by me. A special thank to my friends and colleagues, with whom I shared magnificent moments. And last, but not least, my expression of gratefulness is to Joana Pinto, for her patient and enduring support.

Resumo

As atividades humanas são caracterizadas como sendo complexas, não só pelos seus padrões espaço-temporais, mas também devido ao facto de que cada ser humano é fisicamente diferente. A mesma atividade é efetuada diferentemente por várias pessoa e até o mesmo ser humano pode executar a mesma atividade com movimentos distintos. Neste sentido, existe a necessidade de incorporar métodos de aprendizagem máquina capazes de lidar com esta variabilidade. No entanto, mesmo com o aumento de atenção que esta área de investigação está a ter, a maior parte destas técnicas de aprendizagem, embora precisas/exatas, são bastante lentas para treinar e classificar. Esta característica torna tais ferramentas inviáveis para aplicações em tempo-real, onde um robô tem de identificar correta e rapidamente uma dada atividade, por forma a responder adequadamente. Neste contexto, é proposta uma metodologia de reconhecimento de atividades humanas que é treinada rapidamente e requer poucos exemplos de treino, consistindo em duas componentes base: uma componente de extração de características (isto é, extração de informação relevante que descreve uma certa actividade) e uma componente de aprendizagem, baseada no classificador de florestas aleatórias (*random forests*). Inicialmente, é explorado o conceito baseado na segmentação de cada atividade numa sequência de janelas de tamanho fixo (isto é, a classificação não é feita quadro a quadro), extraindo apenas valores máximos e mínimos de cada característica obtida baseada no esqueleto humano. Após a sua validação experimental, foi apresentada e testada uma segunda abordagem, considerando a divisão de cada atividade em janelas de tamanho variável, baseadas em poses chave. Cada janela de ação é delimitada por duas poses chave consecutivas que são identificadas automaticamente, sendo extraídas características estáticas (isto é, geométricas) e dinâmicas (isto é, temporais). Primeiro, estas abordagens foram testadas usando o *Cornell Activity Dataset* e o classificador de florestas aleatórias disponibilizado pelo programa *Weka*, obtendo-se resultados médios globais relevantes. Em seguida, foi desenvolvido de raíz um classificador de florestas aleatórias, usando o algoritmo de evolução diferencial como parte do seu núcleo. O classificador de florestas aleatórias desenvolvido foi testado e os seus resultados comparados com os previamente obtidos com o programa *Weka*, observando-se uma ligeira melhoria em termos dos indicadores de desempenho considerados. Foi construído um *dataset*, usando um sensor RGB-D, com base no qual foram testados ambos os classificadores de florestas aleatórias. Toda a estrutura foi implementada numa plataforma robótica real (em C++), após a sua validação em ambiente *Matlab*. Após algumas observações finais, destacam-se novas direcções de pesquisa, por forma a melhorar as características da metodologia proposta de reconhecimento proposta e colmatar as suas limitações.

Palavras-chave: Reconhecimento de Atividades Humanas, Florestas Aleatórias, Características Max-Min Baseadas em Esqueleto, Poses Chave, Características Estáticas e Dinâmicas.

Abstract

Human activities are characterized as being complex, not only for their temporal and spacial patterns, but also because of the fact humans are physically different. When the same activity is performed by different persons, or even by the same one, they can execute the same activity with distinct motion properties. In this sense, there is the need to incorporate machine learning approaches able to integrate all this variability. However, even with the increasingly attention this field of research is getting, current learning approaches, although accurate, are very slow to train and classify. These limitations turn infeasible their use in real-time applications, where a robot must rapidly and correctly identify a performed activity, in order to respond accordingly. A human activity recognition framework, featuring fast training and requiring few training examples, is therefore proposed, consisting of two main components: a features extraction approach component (i.e. extraction of relevant information describing a certain activity) and a machine learning component, based on the random forest classifier. An initial concept of segmenting each activity into a sequence of fixed-size actions is employed (i.e. no frame-by-frame classification), extracting just maximum and minimum values of each extracted skeleton-based feature. After some experimental validation, a second approach was developed and tested, considering the division of each activity into variable-size windows, based on key poses. Each action window is delimited by two consecutive and automatically identified key poses, where static (i.e. geometrical) and maximum and minimum dynamic (i.e. temporal) features are extracted. First, these approaches were tested using the Cornell Activity Dataset and the random forest classifier provided by Weka Software, obtaining relevant overall average results. Then a custom random forest classifier was developed from scratch, using a differential evolution algorithm, as part of its core. The developed random forest was tested and its results compared to the ones obtained with Weka's random forest, suggesting a slight increase in terms of the considered performance indicators. A custom dataset was built, using data from a RGB-D sensor, and both random forest classifiers were tested. The proposed framework, after being validated using Matlab, was implemented in a real robotic platform (in C++). After some concluding remarks, new open research directions are highlighted, in order to improve the framework's characteristics and to bridge its drawbacks.

Keywords: Human Activity Recognition, Random Forests, Max-Min Skeleton-based Features, Key Poses, Static and Dynamic Features.

Contents

List of Figures

List of Tables

List of Acronyms

3D	Three Dimensional. 2, 5, 9–11, 19, 20, 22, 36, 42, 43, 45–47
CAD-60	Cornell Activity Dataset. 7, 35–37, 39, 46
CART	Classification and Regression Trees. 3, 7, 12, 14, 15, 29, 33, 34
DD	Developed Dataset. 7, 42–44
DE	Differential Evolution. 5–7, 9, 15, 16, 30, 31, 34, 39, 45
DRF	Developed Random Forest. 6, 7, 38–45
DT	Decision Trees. 3, 6, 11–15, 30–34, 39, 46
RF	Random Forest. 2–7, 9, 14, 15, 24, 29, 30, 33–35, 37–39, 41–44, 46, 48
RGB-D	Red, Green, Blue, Depth. 2, 9, 36, 42, 43
ROS	Robotic Operating System. 4, 5, 7, 35, 42, 43, 46

1 Introduction

Robot perception is still an open research area, which combines research endeavors from many fields, such as computer vision, machine learning and pattern recognition. It is a very complex subject due to the dynamic nature of the environment in real-world application scenarios. This is specially true when there is the need for robots to interact with humans, which adds another layer of complexity, but also excitement. In this sense, robots must know their surrounding environment, so that they can behave accordingly.

Human activity recognition is just a small part of the global field of robot perception, but plays a huge role in it. There are many areas in which the accurate response of the robot to certain situations is very relevant and even, in extreme cases, of life or death.

For example, as the world population is increasingly aging [1], many people will be unable to live independently. In this case, an autonomous system could provide quality of life to those people, guaranteeing their basic needs, as well as comfort, security, safety, commodity and health monitoring. For this purpose, first, the robot must perceive and understand what are the person intentions, but also detect and identify crucial actions that are unpredictable, such as falling. Then, it can act accordingly to the perceived information, in order to achieve the best possible response.

Human ordinary communication is also another example. Not only humans communicate verbally, but also with gestures: sometimes a person is saying one thing with their words, but another (possibly completely different) with gestures. In this sense, for a robot to deeply understand a human conversation, it must accurately identify, not only those gestures and actions, but also their intentions. This example has also some related applications, such as understanding human gesture language or detect when a person is lying.

Another example is in the field of autonomous vehicles, where the unpredictability of pedestrians' actions and intentions could cause devastating consequences. Being able to identify, act and predict human/pedestrians actions is vital for a successful autonomous car navigation system.

With these daily examples in mind, not only a human activity recognition system is necessary, but also it has to be very fast, capable of adapting rapidly to an uncountable number of unpredictable actions performed by users. Because of these reasons, in recent years, this interesting field of research has received the attention of researchers from all around the world, specially with the introduction of Red, Green, Blue, Depth (RGB-D) sensors [2]. These sensors provide depth images and Three Dimensional (3D) point clouds with important attributes, such as robustness to illumination's variation, scaling and rotation. Also, a 3D human skeleton is possible to be acquired in real-time [3], with affordable equipment, such as Microsoft Kinect RGB-D cameras.

Based on a recent survey [4], the number of published papers about 3D skeleton-based human representations has grown significantly, from about 10 papers in 2007 to about 60 in 2014. These numbers represent about half the total number of publications related to human representation and detection, since these can be grouped into two large categories: representations based on local features and skeleton-based representations.

3D skeleton-based representation has the potential to describe a human body and motion, with a relatively small amount of information, such as joint positions, as Johansson demonstrated in [5]. Also, it is possible to extract additional meaningful information from them, such as joint velocities and accelerations (i.e. skeleton-based features). In addition, not only 3D skeleton-based representations are robust to illumination's variability and camera's perspective view, but also they are not significantly affected by skeleton's rotation or motion speed. Such human representation method is particularly suited for real-time applications, since it provides a compact representation of the human body, requiring less computational power to process it.

However, human representation is just a part of the work in human activity recognition. There are several different human activities, some very distinctive from each other, others not so much. Also, each activity may be performed by any capable human (i.e. in this stage of research, only complete human body skeletons are considered). Since each human is physically different, a finite set of rules describing a given activity is very difficult (if not impossible) to find. Adding the fact that the same activity may be performed differently by the same person (e.g. *arm waving* more slowly or more quickly; *walking* in different manners, like *lumbering* or *parading*), a machine learning approach must be implemented, so that a robot can learn by observing several examples of the same activity, performed by different subjects and in different manners.

From a variety of machine learning approaches, the Random Forest (RF) algorithm was adopted. There were several reasons for its choice, which are discussed subsequently. RF are an ensemble of Decision Trees (DT) classifiers, which means each classifier votes for one class, being the most voted class the selected one. Therefore, RF are also relatively robust to noise, since many badly trained DT [1], based on certain input variables, may vote for a wrong class, but the average of votes is expected to be in the right one. Also, since RF are built based on DT, the resulting procedure is simple, elegant and easily parallelized. Considering some properties of DT, such as interpretability, nonparametric method, no limitations of type or value for input variables, low bias if grown sufficiently deep and fast classification approach, RF prove to be also a very accurate and fast machine learning method, comparing to others. Another motivation is related to the fact that this is not a very well studied classifier, in the human activity recognition domain and related ones, when comparing to other classifiers, such as Neural Networks, Support Vector Machines, Naive Bayes, Hidden Markov Models or Deep Learning.

The focus of this work was to develop and implement a "Keyframe Based Human Activity Recognition Using Random Forests". This topic provided an excellent opportunity to study machine learning algorithms (i.e. Classification and Regression Trees (CART) and RF), skeleton-based features extraction methods and meta-heuristic optimization algorithms (i.e. differential evolution). The main objective of this work was to assemble all of this knowledge and implement a real-time application, capable of recognizing a set human activities, requiring few training examples and small training time, maintaining an acceptable overall performance.

1.1 Motivation

The pursuit of humankind to create an entity in his likeness goes back many years, to at least the ancient Greeks ages. But just in recent times, with the invention of mechanical machines and computers, this dream has become a possible reality. In 1965, an Artificial Intelligence pioneer, Herbert Simon, stated: «*Within twenty years, machines will be able to perform any task a human can*» [2]. Yet today, many progresses must be made for that goal to be achieved, despite all human endeavors and accomplishments.

Therefore, this work proposes to contribute towards the development of autonomous robots, capable, not only of recognizing human activities, but also of learning fast and

[1]DT are notoriously noisy [6]. This means they benefit from the voting averaging in a RF ensemble.

[2]Extracted from Michio Kaku's book, entitled: *Physics of the Future*.

adapting accordingly to the human world reality. These robots should provide support to humans, understanding their intentions, acts and behaviors, as well as learning from them. Capability of reaction to potential risk situations, but also being aware of human needs is a must, for a successful autonomous robots implementation. In a sentence, the main motivation behind this work was to learn how to make use of state of the art knowledge, so that machines can interact autonomously with humans and help them in every possible scenario, in the future.

1.2 Objectives

The core of this work is to develop a fast training, requiring few training examples, human activity recognition system, so that it can be implemented in real world applications. To this end, several structural stages must be built, such as features extraction methods and machine learning classifiers. Also, their assessment must be made, in order to evaluate the system performance.

In short, the main objectives are as follows:

- develop and implement a fast training, requiring few training examples, human activity recognition framework, with relevant overall performance, that is real-time oriented;
- develop and implement a features selection approach based on segmenting each activity into a set of atomic actions;
- learn, develop and implement a RF classifier in C++;
- test, compare and analyze the system's performance;
- implement the built framework in a real robotic platform, using the Robotic Operating System (ROS) environment.

1.3 Implementation and Main Contributions

A novel real-time oriented human activity recognition framework was proposed and implemented. The system is composed by two core components: **features extraction** and **RF classifier**. The features extraction component consists of four stages: preprocessing, features extraction, features normalization and features selection. One of the main contributions of this work relies in the developed features selection approach, which divides a human activity into several actions, by means of windows' examples. In this sense, no frame-by-frame learning/classification is done, which improves the overall system in terms of training speed and required number of training examples.

The machine learning classifier component was implemented from scratch in C++ and integrated into the ROS environment, along with all other considered components. The developed RF has some interesting properties. First, there are no *thresholds* to tune, in the sense that, if there is a change in the proposed default parameters, no significantly alterations are observable in the out-coming results. Second, each node split is made, based on the best solution provided by the Differential Evolution (DE) heuristic algorithm. And third, the stop splitting node condition is based on a desired confidence level (i.e. statistical interpretation).

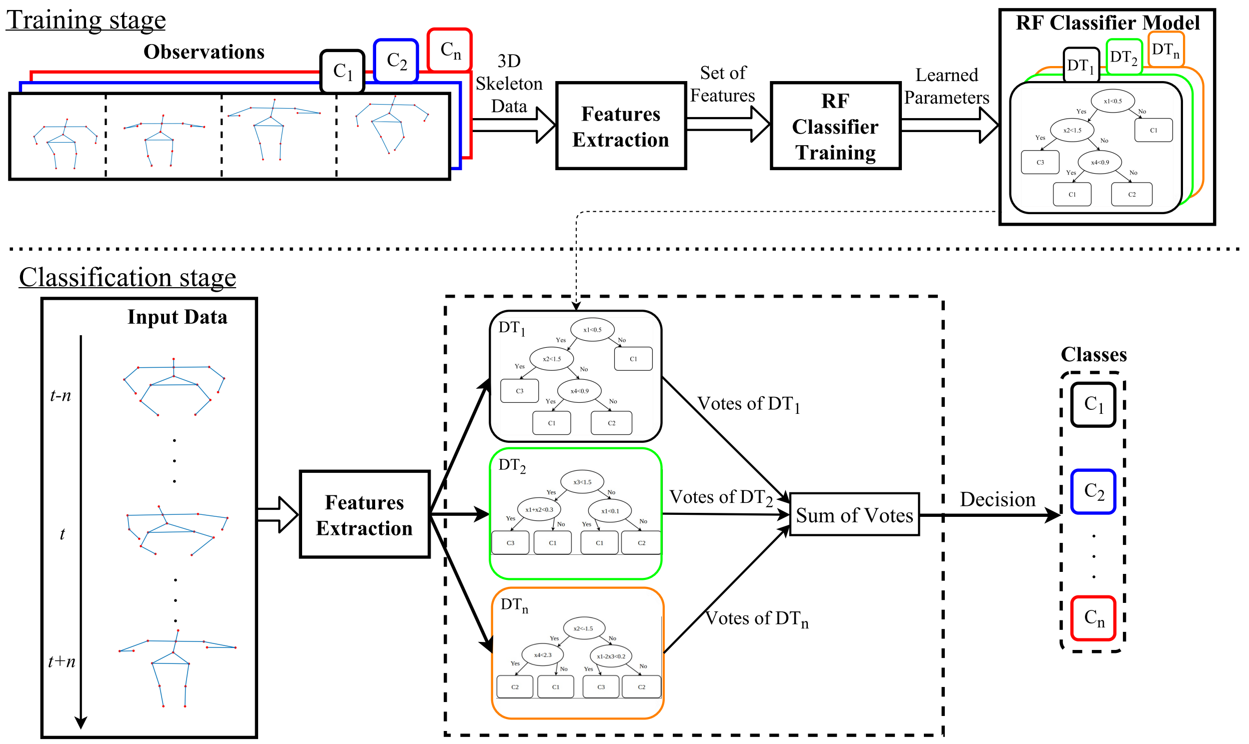


Figure 1.1: Overview of the global architecture of the developed framework. **Training stage**: the RF classifier is trained for each class (i.e. activity), making observations of humans performing activities; from each observation, 3D skeleton data is extracted, as well as discriminative information (i.e. features extraction); a RF model is built. **Classification stage**: given a set of data of a human performing an activity, features are extracted and selected; the trained RF classifier, based on the selected features, makes a decision, classifying the performed human activity.

In Fig. 1.1, a schematic of the global developed framework is shown, which serves as a base to a comprehensive explanation of all the implemented considerations.

The structural organization of this dissertation is as follows:

- **Introduction (Chapter 1):** Presentation of the focus of the dissertation, as well as its context in the real world and its possible applications. A brief discussion about motivation and proposed objectives is made. The implementation and main contributions of the work are highlighted, as well as the dissertation's organization.
- **State of the Art and Background (Chapter 2):** Related research work addressing human activities recognition, random forests and differential evolution algorithms is presented and analyzed. Also, the theoretical background to support the developed work is discussed.
- **Features Extraction (Chapter 3):** The pipeline of the developed features extraction stage is discussed. Several skeleton-based features are proposed, as well as two related features selection approaches, one based on segmenting each activity into fixed-size windows and the other based on segmenting activities into variable size windows using the concept of key poses.
- **Implemented Classifier - Random Forests (Chapter 4):** The RF classifier's implementation in C++ is presented, as well as all the underlying algorithms and structures, such as DT and DE.
- **Experimental Results (Chapter 5):** Relevant experimental results are highlighted, involving comparison studies regarding features extraction (e.g. the proposed approach with other state of the art ones) and classifiers (e.g. Developed Random Forest (DRF) with Weka's RF).
- **Conclusion and Future Work (Chapter 6):** The developed work is discussed, analyzing its strengths and limitations. Some new lines of research are envisaged, in order to further improve the proposed framework, making it more accurate in real-time, maintaining its speed and requirement of few training examples.

The implementation and main contributions are highlighted as follows:
Features Extraction (Chapter 3):

- a four-stage features extraction approach (preprocessing, features extraction, features normalization and max-min features selection);
- two distinct max-min feature selection approaches, respectively based on fixed-size windows and key poses delimited actions, are proposed;
- proposal of static and dynamic skeleton-based features.

Implemented Classifier - Random Forests (Chapter 4):

- step by step development of a custom CART algorithm, without the need to tune thresholds;
- DE algorithm as splitting node heuristic;
- stop splitting node condition as a variation of the *hypothesis testing* (i.e. with statistical interpretation);
- custom RF implementation in C++;
- integration of the DRF into ROS environment.

Experimental Results (Chapter 5):

- validation of the activity segmentation concept, using Cornell Activity Dataset (CAD-60) and leave-one-out cross validation;
- comparison of the proposed approach with other state-of-the-art methods;
- comparison of the DRF with the one provided by Weka [7];
- validation in a custom Developed Dataset (DD), with daily human activities.

2 State of the Art and Background

In this chapter, the recent most relevant research related with the topics addressed by this work is discussed. Some theoretical background is also provided, focusing in the implementation and design usefulness of the techniques used. The material presented in this chapter is essential to understand the developed work, as well as to motivate implemented ideas and future lines of research.

2.1 State of the Art

As mentioned before, human activity recognition has increasingly become a very important research area, due to its future possible real-world applications, such as surveillance [8], assistive living [9] and human-machine interaction [10]. Some relevant related work is presented in the field of human representation, as well as recent activity recognition approaches. Also, a brief overview of some Random Forest (RF) and Differential Evolution (DE) applications are discussed, since these algorithms are used in several areas of research.

2.1.1 Human Representation

For an autonomous system to be able to identify a human activity, first, there is the need to model and detect the human. With the popularization of low cost RGB-D sensors (e.g. Microsoft Kinect [11] and Asus Xtion PRO LIVE [12]), the task of human representation and detection has become much easier. These sensors provide, not only 2D visual data, but also depth information. Based on the provided data, it is possible to extract relevant features, which have important attributes, such as robustness to illumination's variations, scaling and rotation [13]. In addition, 3D human skeleton data is possible to be acquired in real-time [3]. This means that, starting from a 3D point cloud, only skeleton joints information is extracted, reducing the amount of input raw data. Also, based on the acquired skeleton joints information, skeleton-based features are possible to be extracted, such as joints velocities, angles between joints, etc. Therefore, 3D skeleton-based representation may provide information about relationships between joints, as well as body information as a whole, requiring just 3D skeleton joints data. Human representation based on depth images is also another possibility. These approaches rely on 3D clustering methods, based on 3D point clouds, removing the ground environment and obtaining a human silhouette. Then, a set of human silhouette features are extracted, using procedures like histogram of oriented gradient. Examples of related developed work are [14] and [15], where in the latter only depth information is used.

2.1.2 Activity Recognition Methods

The majority of recent human activity recognition frameworks rely in features based on the previously mention types of human representation (see Sect. 2.1.1). A method based on depth images, with sequences of temporal unique poses is proposed by Gupta *et al.* [16]. Also, Chen *et al.* [17] used a depth motion map approach for activity recognition, transforming each 3D depth frame into three 2D projected maps (front, side and top views). These are some illustrative examples of human activity recognition based on depth images. Since this work is based on 3D human skeleton representation, the following mentioned works are based on this type of information. Yang and Tian [18] proposed a formation of eigenjoints descriptors, incorporating static postures and overall dynamic motion. Features based on human pose and motion, combined with histogram of oriented gradient features from depth images, are extracted, based on the work of Sung *et al.* [19]. Shan and Akella [20] used the skeleton kinetic energy to identify key poses, which are then used as features. This is one of the works that inspired some main proposed ideas of this work. A dynamic Bayesian mixture model is developed and implemented for classification, by Faria *et al.* [21], considering also the *torso* of the skeleton as reference. Ding *et al.* [22] used an HMM approach to classify sequences of discrete symbols, based on segmented action-units. A structural support vector machine (SVM) algorithm is proposed, not only to classify human activities, but also to perceive human interactions with objects, by Koppula *et al.* [23]. Considering each activity a sequence of key poses and atomic motions, *Zhu et al.* [24] propose a multi-layer codebook activity framework recognition of such sequences, each one representing patterns of certain human activities.

2.1.3 Random Forests Applications

Random forests were first introduced by Leo Breiman [25]. His idea was to develop a low error, with low variance error classifier, inheriting some advantageous properties of Decision Trees (DT) and, at the same time, bridging some of their disadvantageous, such as the fact that they are very sensitive to noise. These classifiers have proved to be very accurate, simple and fast, comparing to other machine learning techniques [6]. A comprehensive work about the properties of random forests was developed by Gilles Louppe [26]. They are used in several fields of research, as illustrated subsequently. Sharma *et al.* [27] proposed a pixelwise object classification for real-world applications, with the intention to provide a secure human-robot interaction in industrial domains. Gan and Chen [28] proposed a human action recognition framework, using depth image sequences, based on 3D joint position and 3D joint angle features. Demirdjian and Varri [29] presented a novelty of recognizing temporal events, based on temporal random forests, where each decision tree node contains temporal information. A real-time gesture recognition scheme was employed by Miranda *et al.* [30], where the node in each DT represented a key pose and each leave represented the corresponding gesture, as one goes down a DT.

2.1.4 Differential Evolution Applications

Differential evolution is a meta-heuristic algorithm initial proposed by Storn and Price [31], with an increasingly number of users, and was considered the best genetic type algorithm in the *First International Contest on Evolutionary Computation*[1]. Its applications include neural network learning [32], radio network designs [33] and reflectivity curve simulations[2].

2.2 Background Material

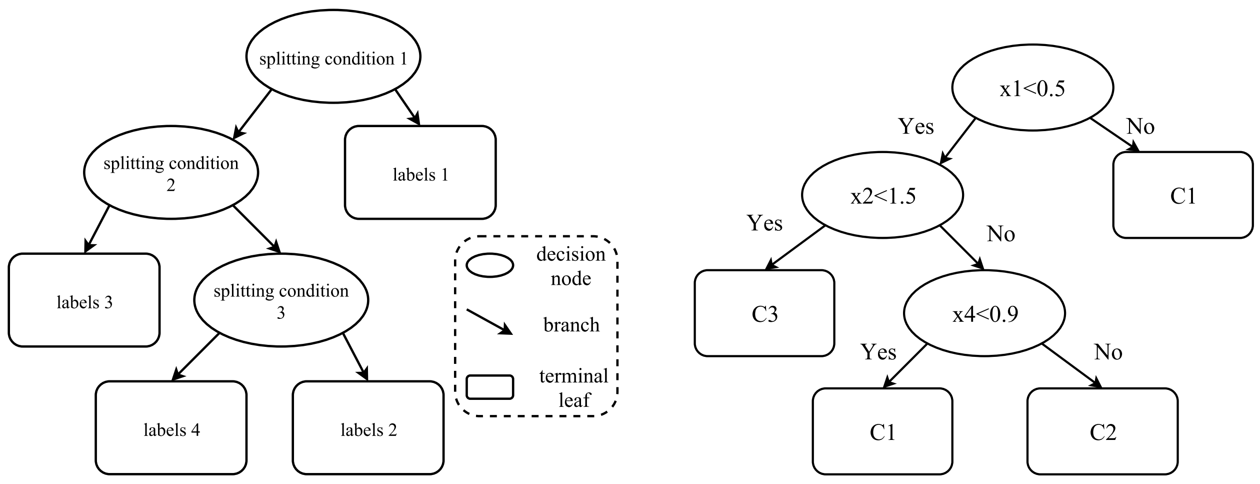
2.2.1 Decision Trees - Classification and Regression Trees

DT are efficient nonparametric methods of classification for supervised learning, implementing the divide-and-conquer strategy in a sequence of recursive splits [34]. Also, its input may consist of sets of numeric (continuous or discrete) and non-numeric variables (e.g. names,

[1] `http://www1.icsi.berkeley.edu/~storn/code.html`, at fourth September, 2016.

[2] `http://www-llb.cea.fr/prism/programs/simulreflec/simulreflec.html`, at fourth September, 2016.

colors, etc).



(a) Generic DT model. (b) Example of a DT model.

Figure 2.1: Decision tree model. Decision nodes are represented by ovals and terminal leaves by rectangles. An univariate and binary tree is shown in the DT model example.

They are composed of decision *nodes*, *branches* and terminal *leaves* (see Fig. 2.1). In each node, a test function is applied to the input, which takes a branch, depending on the test outcome. Starting at the root, this process is repeated, until reaching a leaf, containing the output.

Classification and Regression Trees (CART) is a generic decision tree-growing approach, with six underlying questions [35]:

1. How many outcomes or *splits* will there be at a node?
2. Which property should be tested at a node?
3. When should a node be declared a leaf?
4. If the tree becomes too large, how can it be made smaller and simpler?
5. If a leaf is impure, how should the category label be assigned?
6. How should missing data be handled?

These questions are the guideline for constructing the implemented DT algorithm and will all be answered in Chap. 4. For now, it is important to discuss how to split a node, defining *impurity*, which is the measure of the goodness of a node split. A split is *pure* if, after the split, all the instances reaching a node belong to the same class. In this case, the impurity of that node is 0 or 1.

There are several impurity measures:

$$\text{Entropy:} \quad \gimel_m = -\sum_{i=1}^{K} p_{m,i} \log_2 p_{m,i} \tag{2.1}$$

$$\text{Gini:}\quad \gimel_m = 1 - \sum_{i=1}^{K} p_{m,i}^2 \tag{2.2}$$

$$\text{Misclassification:}\quad \gimel_m = 1 - \max_{i=1}^{K} p_{m,i}, \tag{2.3}$$

where K is the number of classes and $p_{m,i} = \frac{N_{m,i}}{N_m}$ is the fraction of instances of class C_i that reaches node m (therefore $N_{m,i}$ is the number of instances of class C_i, reaching node m, such that $N_m = \sum\limits_{i=1}^{K} N_{m,i}$). In this sense, a good heuristic for splitting a given node m is to maximize its drop in impurity, defined by:

$$\Delta\gimel_m = \gimel_m - \sum_{j=1}^{n} p_m^j \gimel_m^j, \tag{2.4}$$

where $p_m^j = \frac{N_m^j}{N_m}$ is the fraction of instances going for branch j. It is important to notice that this approach does not guarantee an optimal DT in the sense of being compact with as few nodes as possible. The reason for that is the fact that in each node a maximum drop in impurity is searched (i.e. local maximum), instead of a search for the best drop in impurities in all the nodes of the tree (i.e. global maximum).

A condition is necessary to stop splitting nodes. Setting a minimum threshold for the drop in impurity is a possibility. Another stopping condition is to minimize a global criteria function, such as

$$\alpha \cdot \text{size} + \sum_m \gimel_m, \tag{2.5}$$

where α is a positive constant, size could represent the number of nodes of the tree and $m \in \{\text{leaf nodes}\}$. This criteria takes into account the uncertainty associated to the model represented by the tree (i.e. sum of impurities of the leaf nodes) and the complexity of it (i.e. size of the classifier).

It is also possible to use a condition based on statistical significance of the reduction of impurity. In this approach, a candidate split is tested applying a variation of the *hypothesis testing*, which is used to determine if it differs from a random split, accordingly to some *confidence level* associated to some statistics. This deviation may be quantified using *chi-squared statistics* (considering the split between only two branches) as

$$\chi^2 = \sum_{i=1}^{K} \frac{\left(N_{m,i}^L - N_{m,i} p_m^L\right)^2}{N_{m,i} p_m^L}, \tag{2.6}$$

where $N_{m,i}^L$ is the number of instances of class C_i sent to the left branch and $N_{m,i} p_m^L$ is the expected number of instances of class C_i sent to the left branch by the random rule.

When χ^2 is greater than some value given by the desired confidence level associated to the chi-squared statistics, it means that the candidate split differs from the random one and the null hypothesis is rejected, continuing the node splitting. On the other hand, if χ^2 is smaller, the splitting is stopped, since the candidate split does not differ from a random one.

The pseudo-code of the generic CART algorithm is as follows:

Algorithm 1 Generic CART Algorithm

```
find best split rule;
if split is advantageous then
    for all branches do
        find instances falling in branch;
        generate tree using CART algorithm 1;
    end for
else
    create leaf, with reaching labels;
    return ;
end if
return tree model $T$;
```

2.2.2 Random Forests

RF were introduced by Breiman [25] and consists on an ensemble of DT, such that the training input of each DT is a bootstrap sample of the original input. Each sample is independent identically distributed (i.i.d.), to keep residual correlation between trained DT, and trees are grown to the largest extent possible, without pruning, keeping each individual tree error low. One interesting property of RF is that they do not overfit as the number of trees is increased [25]. Although, Segal in [36] demonstrates small gains can be made by controlling the maximum size of trees for large datasets, this will not be a concern.

DT can store complex structures in the input data, without a prior knowledge of it and can have low bias, when grown sufficiently deep. However, they can be very noisy and overfit, resulting in high variance; hence they benefit from the averaging process, as in RF.

The idea of RF is to inherit the low error from enough deeply grown DT and at the same time reduce their variance. Since each DT is generated from i.i.d. data, the expectation of the average of the error of the set of DT is the same as the expectation of any one of them. In other words, the error of an ensemble of i.i.d. DT is expected to be as low as any one of the set's DT. This means, the only possible improvement is to reduce the variance of the average's error. Defining ρ as the Pearson's correlation coefficient between pairwise DT models and σ^2 their corresponding error's variance, it follows that the variance of the average's error of an ensemble of DT (see [26] for a comprehensive demonstration) is given by

$$\rho\sigma^2 + \frac{1-\rho}{N}\sigma^2, \tag{2.7}$$

where N is the number of trees on the ensemble. As N arbitrary increases (i.e. $N \to \infty$), the variance reduces to $\rho\sigma^2$. As one can see, if each DT model is built from a set of randomized input variables, $\rho < 1$, which means the variance of the ensemble is lower than the one of each of its individuals. In this sense, if the effects of the randomization procedure are strong enough, the RF classifier provides expected low error and variance rates.

The pseudo-code of the generic RF algorithm is as follows:

Algorithm 2 Generic Random Forests Algorithm

```
for $i = 1$ to $N$ do
    draw a bootstrap sample $F_i^*$, from the training input;
    grow a tree $T_i$ recursively, using CART algorithm 1 and selecting $m$ variables at random to split each node, from $F_i^*$;
end for
return set of trees $\{T_i\}_1^N$;
```

2.2.3 Differential Evolution

DE is a meta-heuristic algorithm [31], which aims to optimize non-linear, non-differentiable and non-continuous problems with continuous variables. Such problems are very difficult (and some times impossible) to solve analytically. This algorithm does not guarantee the optimal solution for the problem, but may be used to find an approximated optimal solution (i.e. sub-optimal solution) in a desired amount of time.

The general problem formulation is to find $x^* \in \mathbb{X} \subseteq \mathbb{R}^n$ that minimizes an objective function $f(x)$.

The basis of the algorithm is that, given a set of possible solutions (also denominated *individuals*), the best one is chosen, accordingly to the objective function. Therefore, a set of random possible solutions is generated, forming a *population*:

$$X_t = \{x_{t,1}, x_{t,2}, ..., x_{t,N}\}, \tag{2.8}$$

where $x_{t,i}$ is a possible solution to the problem

$$x_{t,i} = \begin{pmatrix} x_{t,i,1} & x_{t,i,2} & \dots & x_{t,i,n} \end{pmatrix}, \tag{2.9}$$

with $x_{t,i,j}$ being the variable j of the problem, of the individual i, at generation t. This is the *initialization* step. The next step is the *mutation* step, where each individual is mutated accordingly to

$$v_{t,i} = x_{t,r_1} + F\left(x_{t,r_2} - x_{t,r_3}\right), \tag{2.10}$$

where $r_1, r_2, r_3 \in \{1, 2, ..., N\}$ are random indexes of individuals distinct from i and each other and $F \in [0, 2]$ is the mutation factor. *Recombination* is the next step:

$$u_{t,i,j} = \begin{cases} v_{t,i,j} & \text{if rand} < c \vee j = \delta \\ x_{t,i,j} & \text{otherwise} \end{cases}, \tag{2.11}$$

where rand $\in [0, 1]$ is a random variable, $c \in [0, 1]$ is a recombination factor and $\delta \in \{1, ..., n\}$ is a random index for recombination to ensure that $u_{t,i} \neq x_{t,i}$. The last step is to *select* the best individuals, forming a new population, such that

$$x_{t+1,i} = \begin{cases} u_{t,i} & \text{if } f(u_{t,i}) < f(x_{t,i}) \\ x_{t,i} & \text{otherwise} \end{cases}, \tag{2.12}$$

where $f(\cdot)$ is the objective function.

All the steps, with the exception of the initialization, are repeated a maximum number of times (or until some other condition is met). The pseudo-code of the generic DE algorithm is as follows:

Algorithm 3 Generic Differential Evolution Algorithm

```
$X_0 \leftarrow$ generate initial population randomly;                          (*initialization*)
**while** stopping condition is not met **do**
    $V_t \leftarrow$ mutate each individual, accordingly to Eq. (2.10);            (*mutation*)
    $U_t \leftarrow$ recombine each individual, accordingly to Eq. (2.11);    (*recombination*)
    $X_{t+1} \leftarrow$ select best individuals, accordingly to Eq. (2.12);        (*selection*)
**end while**
**return** best individual $x^*$;
```

2.2.4 Activity Classification Measures

There are two indicators adopted to evaluate the performance of the proposed classification framework: precision (Prec) and recall (Rec).

Precision is the ratio between the number of elements correctly labeled (i.e. true positives - TP) and the total number of elements labeled belonging to the same class (i.e. true positives plus false positives - FP):

$$\text{Prec} = \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{2.13}$$

Recall is the ratio between the number of elements correctly labeled and the total number of elements belonging to the same class (i.e. true positives plus false negatives - FN):

$$\text{Rec} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{2.14}$$

In this sense, precision is a measure of *exactness*, whereas recall is a measure of *completeness*.

3 Features Extraction

In this chapter, both proposed features extraction methods are described, as well as all the necessary steps to implement them. In Fig. 3.1, an overview of the features extraction method is illustrated. Considering the coordinates system defined as x, y, z, corresponding to the width, height and depth, respectively, relatively to the camera, a dataset containing 3D coordinates of skeleton's joints is assumed to be provided. Each 3D joint's position is given by $P_j^t = \left(p_{jx}^t, p_{jy}^t, p_{jz}^t\right)$, where p_{dj}^t is the value of the coordinate $d \in \{x, y, z\}$ of the joint $j \in \{1, ..., m\}$, at frame t and m is the number of body's joints.

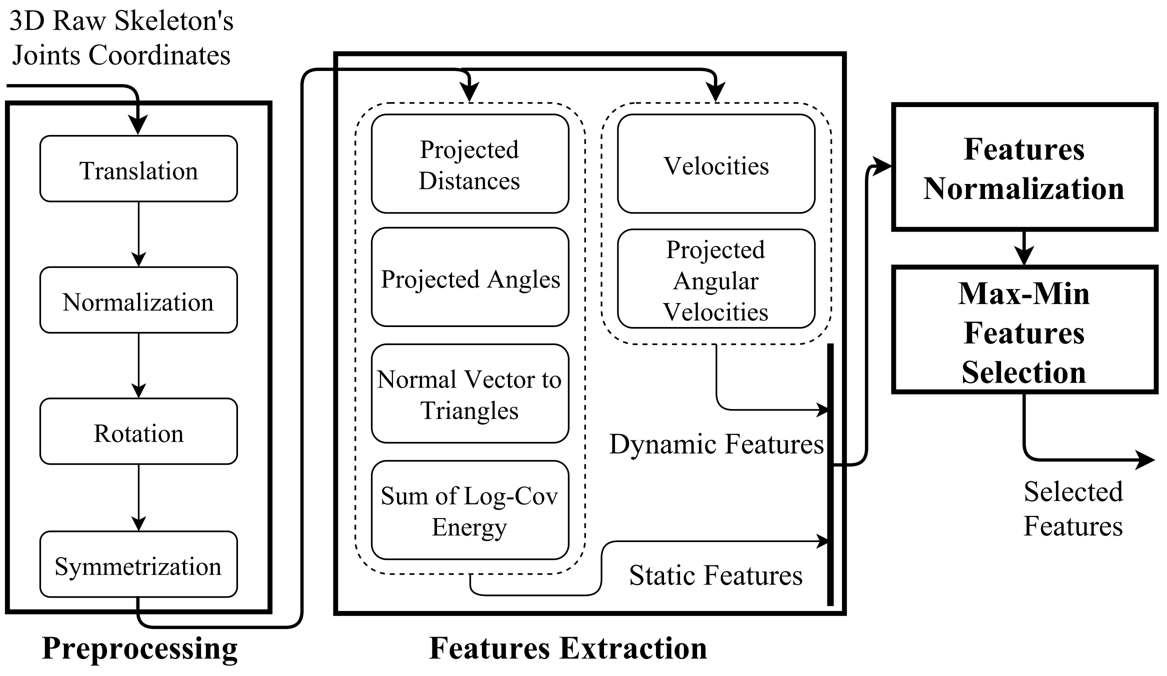


Figure 3.1: Overview of the developed features extraction method. The raw skeleton's joints coordinates are first preprocessed, suffering a set of transformations like translation, normalization, rotation and symmetrization. Then, relevant information is extracted, such as velocities of joints, distances between joints, etc. These features are normalized and finally, max-min skeleton-based features are selected, accordingly to the proposed approaches (see Sects. 3.4 and 3.5).

3.1 Preprocessing of 3D Skeleton Data

The preprocessing stage has the objective to attenuate noise introduced by the camera and to accommodate the raw data for different user's height, limb length, orientation and position, in order to have a set of features that best describe a certain human activity, independently of the user's attributes and location.

First, the skeleton is centered in relation to a fixed point (e.g. each joint being centered in relation to the torso, at each frame), accordingly to

$$C_j^t = P_j^t - P^*, \tag{3.1}$$

where $C_j^t = \{c_{jx}^t, c_{jy}^t, c_{jz}^t\}$ is the respective centered 3D skeleton's joint j, at frame t, and $P^* = \{p_x^*, p_y^*, p_z^*\}$ is the considered fixed 3D point. The main reason behind the translation is to disambiguate between different coordinates systems, since the same activity may be seen differently, accordingly to the camera's perspective. Fixing a point as the origin of a new coordinates system ceases this concern.

The normalization step was introduced to attenuate the fact that every human is physically different, with different heights and limb lengths. The normalization is performed accordingly to

$$N_j^t = \frac{C_j^t}{h}, \tag{3.2}$$

where N_j^t is the respective normalized 3D coordinates of joint j, at frame t and h is the height of the subject.

The next step is the rotation of the skeleton. This is a very important step, so that every activity may be successfully recognized, when seen from different points of view. The applied rotation is based on the *orientation normalization* proposed in [37]. Given two normalized normal vectors, one in relation to the desired plane $e_\perp$ and the other to the skeleton plane $\pi_\perp$, a rotation matrix $\mathbf{R}$ may be found accordingly to

$$\mathbf{R} = \mathbf{I}\cos\theta + \mathbf{A}\sin\theta + (1-\cos\theta)xx^T, \tag{3.3}$$

where $\mathbf{I}$ is a 3×3 identity matrix, θ is the rotation angle as

$$\theta = \cos^{-1}\left(\pi_\perp \cdot e_\perp\right), \tag{3.4}$$

x is the rotation axis as

$$x = \begin{bmatrix} x_1 & x_2 & x_3 \end{bmatrix}^T = \frac{\pi_\perp \times e_\perp}{\|\pi_\perp \times e_\perp\|} \tag{3.5}$$

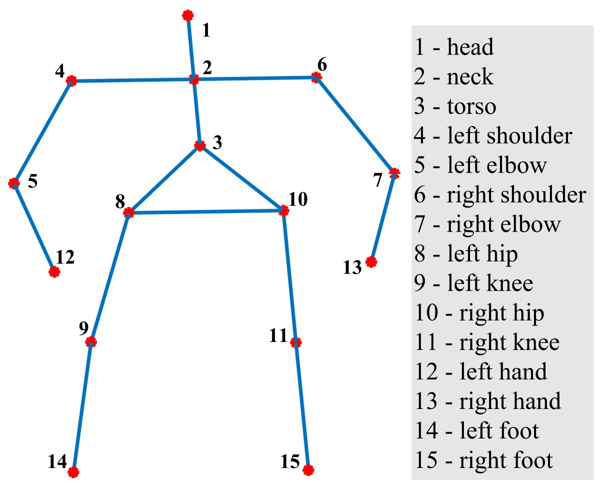
and **A** is a skew-symmetric matrix, given by

$$\mathbf{A} = \begin{bmatrix} 0 & -x_3 & x_2 \\ x_3 & 0 & -x_1 \\ -x_2 & x_1 & 0 \end{bmatrix}. \tag{3.6}$$

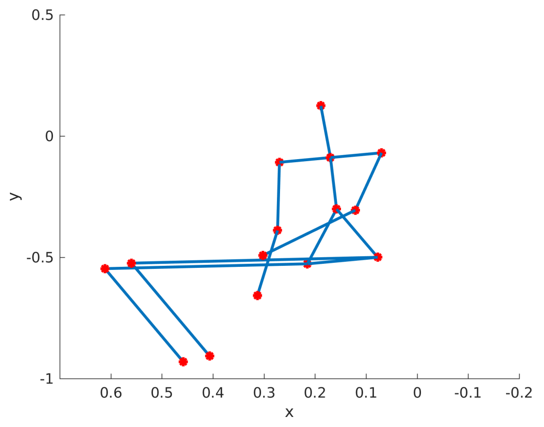
After several experimental results, the skeleton plane is defined by the following joints: *torso*, *right* and *left hips*. The rotation matrix is found for every frame and applied to every skeleton joints.

The last preprocessing step is the symmetrization. This stage is important, mainly to disambiguate between right and left-handed people (e.g. it does not matter if a person *waves* with the right or left arm; the activity is still *wave*). Following the previous steps, in this stage the skeleton joints just need to be the mirror with their symmetric ones (e.g. *right hand* is the symmetric joint of the *left hand*), in relation to a desired plane, since the skeleton is already centered and oriented. The mirrored skeleton that is formed serves as a duplicate sample for each performed activity and subject (i.e. simulation of each subject performing exactly the same activity, but with mirrored gestures). The plane that is orthogonal to the camera's plane and passes through the center of the skeleton was chosen as the mirror plane.

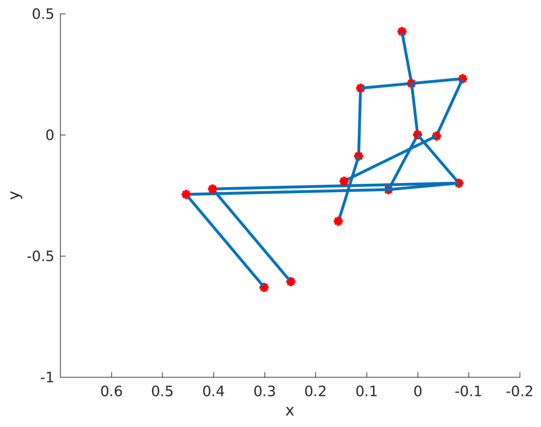
The mentioned steps are applied sequentially and are illustrated in Fig. 3.2. From this point on, the skeleton joints are assumed to be preprocessed and their position will be abusively defined as $P_j^t = (p_{jx}^t, p_{jy}^t, p_{jz}^t)$.



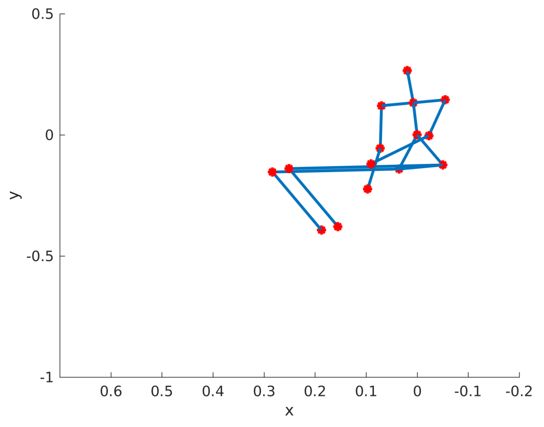
(a) Base skeleton and respective joints.



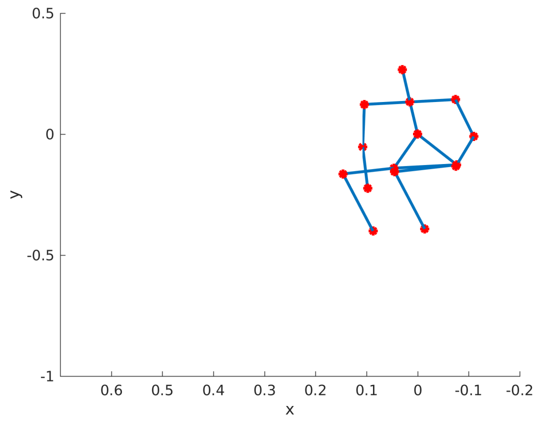
(b) Example of a sit down raw skeleton.



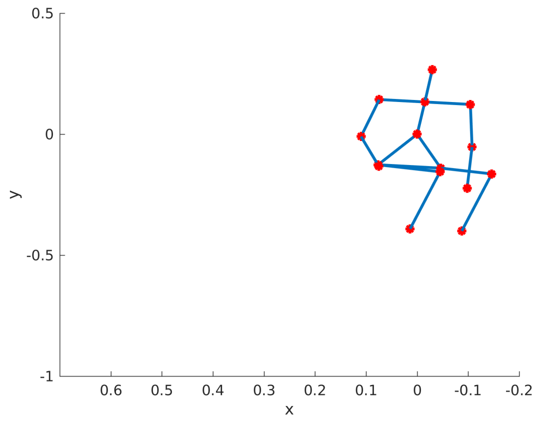
(c) Centered skeleton in relation to *torso*.



(d) Normalized skeleton.



(e) Rotated skeleton, so that its plane is parallel to the camera's plane.



(f) Symmetric skeleton in relation to a mirror plane orthogonal to the camera's plane.

Figure 3.2: Features preprocessing example of a sit down human skeleton. Each step of the preprocessing stage is applied sequentially: the 3D coordinates of the skeleton joints are acquired; the skeleton is centered in relation to a fixed point (in this example, the *torso* joint); the skeleton is normalized; the skeleton is rotated, so that its plane is parallel to the camera's plane; the skeleton is mirrored in relation to an orthogonal plane to the camera's plane.

3.2 Spatio-Temporal Features - Features Extraction

After preprocessing the raw data, extracting relevant and discriminative information (i.e. *features*) is the next step, towards achieving a successful activity recognition framework. Before enumerating them, these may be divided into two distinct categories: *static* (e.g. geometrical) and *dynamic* (e.g. temporal) features. Static features are relevant to represent extremal positions of the skeleton joints, defined as *key poses*, where the skeleton pose has zero kinetic energy [20] (i.e. at a given frame, each joint has zero kinetic energy; therefore the body has no movement). On the other hand, dynamic features are intended to give information about the skeleton's motion, from each joint's movement.

Following this criteria, the considered static features are as follows:

1. **Projected distances** between two joints (a and b) as

$$\delta_{ab}^{t} = \sqrt{\sum_{d'} (p_{ad'}^{t} - p_{bd'}^{t})^{2}}, \tag{3.7}$$

where d' belongs to one of the following sets, for each projection considered: $\{x, y\}$, $\{y, z\}$, $\{z, x\}$;

2. **Projected angles** based on three joints as

$$\theta_{id_p}^{t} = \arccos\left(\frac{(\delta_{ab}^{t})^{2} + (\delta_{bc}^{t})^{2} - (\delta_{ac}^{t})^{2}}{2 \cdot \delta_{ab}^{t} \cdot \delta_{bc}^{t}}\right), \tag{3.8}$$

where δ is the distance between two joints, given by (3.7) and $d_p \in \{xy, yz, zx\}$, for each projection considered;

3. **Normal vector to triangles** formed by three joints as

$$\Delta_{k}^{t} = \frac{(P_{a}^{t} - P_{b}^{t}) \times (P_{a}^{t} - P_{c}^{t})}{\|(P_{a}^{t} - P_{b}^{t}) \times (P_{a}^{t} - P_{c}^{t})\|}; \tag{3.9}$$

4. **Sum of log-cov energy entropy** based on the global skeleton joints position as

$$(l_{\mathrm{cov}p})^{a} = \sum_{i} \mathrm{U}_{i} \left\{\log\left(\mathrm{cov}\mathbf{A}_{p}^{a}\right)\right\}^{2} \tag{3.10}$$

and based on the considered angles as

$$(l_{\mathrm{cov}\theta})^{a} = \sum_{i} \mathrm{U}_{i} \left\{\log\left(\mathrm{cov}\mathbf{A}_{\theta}^{a}\right)\right\}^{2}, \tag{3.11}$$

where $\mathbf{A}_{p}^{a}$ and $\mathbf{A}_{\theta}^{a}$ are matrices containing the values of skeleton joints coordinates and angles, respectively, associated to activity a; cov represents the covariance matrix; log is the matrix logarithm and $\mathrm{U}(\cdot)$ returns elements of the upper triangular matrix. The idea of using the log-covariance is based on the work of Guo [38] and on its application to human activity recognition, similarly to the approach followed by Faria *et al.* [21].

The considered dynamic features are as follows:

1. **Velocities of joints coordinates** as
$$v_{jd}^{t} = \left(p_{jd}^{t} - p_{jd}^{t-1}\right) \cdot f_r, \tag{3.12}$$
where f_r is the frame rate;
2. **Projected angular velocities** as
$$\omega_{id_p}^{t} = \left(\theta_{id_p}^{t} - \theta_{id_p}^{t-1}\right) \cdot f_r. \tag{3.13}$$

These features are combined along time to form a set of features in a matrix form $\mathbf{F}'$, where each row contains the computed features at a given frame and each column corresponds to the variation of each feature along time. Particularly, depending on the situation, a set of training and testing features matrices ($\mathbf{F}'_{\text{tr}}$ and $\mathbf{F}'_{\text{te}}$, respectively) can be formed.

3.3 Features Normalization

In the scope of this work, features normalization are an optionally step, accordingly to the experimental results, computational time and overall increase in algorithmic complexity. Although it is always a good practice, Random Forest (RF) do not require its training/testing data to be normalized. The data normalization is done accordingly to:
$$f_{ij} = \frac{f'_{ij} - \min(\mathbf{F}'_{\text{tr}\cdot j})}{\max(\mathbf{F}'_{\text{tr}\cdot j}) - \min(\mathbf{F}'_{\text{tr}\cdot j})}, \tag{3.14}$$
where f'_{ij} is the current value being normalized, f_{ij} is its respective value normalized and $\mathbf{F}'_{\text{tr}\cdot j}$ refers to the column j of the matrix $\mathbf{F}'_{\text{tr}}$. This process is done for both training and testing sets, resulting in $\mathbf{F}_{\text{tr}}$ and $\mathbf{F}_{\text{te}}$ matrices, respectively. From this point on, these sets are generically referred as $\mathbf{F}$, since in both subsequently approaches, no distinction between them is necessary to be made. $\mathbf{F}$ is a matrix containing all examples of all performed activities of the form:
$$\mathbf{F} = \begin{bmatrix} \mathbf{F}^1 & \mathbf{F}^2 & \dots & \mathbf{F}^a & \dots \end{bmatrix}^T, \tag{3.15}$$
where $\mathbf{F}^a$ is a sub-matrix describing activity a.

3.4 Selection Approach I - Max-Min Skeleton-based Features with Fixed-Size Window

Many activities consist of repetitive action sequences. In this sense, it is possible to assume that each action may be discriminated just by considering extreme movements (given by dynamic features) and extreme poses (given by static features). In other words, given a fixed-size window used to observe a certain repetitive action of an activity, the main idea of this approach is to extract maximum and minimum local values of considered features. This method has low computational cost and is relatively fast to compute, since just maximum/minimum local values are needed to be computed.

Each sub-matrix $\mathbf{F}^a$ describing an activity a is divided into fixed-size sample windows of size n_{f_r} number of frames as

$$\mathbf{F}^a = \begin{bmatrix} \mathbf{F}^a_1 & \mathbf{F}^a_2 & \dots & \mathbf{F}^a_w & \dots \end{bmatrix}^T \tag{3.16}$$

Based on a set of preliminary experiments, a subset of all available features was considered in this approach. They are:

$$\mathbf{F}^a_w = \begin{bmatrix} v^t_{jd} & \theta^t_{id_p} & \delta^t_{ab} \end{bmatrix}^T , \tag{3.17}$$

plus the log-cov energies associated with the respective activities instances, which helped to increase the overall performance of the approach. The size of each sub-matrix is $n_{f_r} \times 3\left((m - m_{\text{extd}}) + n_\theta + n_\delta\right)$, where m_{extd} is the number of joints not considered for velocities computation (e.g. *torso*) and n_θ and n_δ are the number of considered projected angles and distances between joints, respectively.

From each activity example sub-matrix $\mathbf{F}^a_w$, a feature vector is constructed, by computing the $n_{\max}$ maximum and $n_{\min}$ minimum values for each considered feature as

$$f^a_w = \begin{bmatrix} f^{\max}_w & f^{\min}_w & (l_{\text{cov}p})^a_w & (l_{\text{cov}\theta})^a_w \end{bmatrix} , \tag{3.18}$$

where $f^{\max}_w = \begin{bmatrix} (v^{\max}_{jd})_w & (\theta^{\max}_{id_p})_w & (\delta^{\max}_{ab})_w \end{bmatrix}$ and $f^{\min}_w = \begin{bmatrix} (v^{\min}_{jd})_w & (\theta^{\min}_{id_p})_w & (\delta^{\min}_{ab})_w \end{bmatrix}$ correspond to vectors with the $n_{\max}$ maximum and $n_{\min}$ minimum values of each considered feature. The length of each vector is $[(n_{\max} + n_{\min}) \cdot (3((m - m_{\text{extd}}) + n_\theta + n_\delta)) + 2]$.

This is a very naive and limiting approach, but it is a first step to a more general and robust one, as it will be seen. An optimal fixed-size window may vary depending on the activity, which means each activity may have an individual optimal one. This may be very difficult to tune in real-time applications. Additionally, a certain activity may take different times to be executed (e.g. a person may *wave* at different rates), which means a fixed-size window may not be possible to tune. The following approach aims to solve these issues, considering variable-size windows.

3.5 Selection Approach II - Max-Min Skeleton-based Features and Key Poses

In this approach, for each activity, variable-size windows are extracted. These windows are delimited by frames where the skeleton has zero kinetic energy. These poses are defined to be *key poses* [20]. Key poses represent extreme points in the motion path of each joint, where most of the discriminative properties of each action are encoded. In this sense, an activity may be represented by a sequence of distinct body actions, and key poses may be used to determine/delimit their respective window size. In other words, each action is determined by considering two consecutive key poses, which delimit a variable-size window.

From [20], a key poses is defined to be a skeleton pose with zero kinetic energy:

$$E^t = \frac{1}{2}\sum_{j=1}^{m}\sum_{d}\left(v_{jd}^t\right)^2, \tag{3.19}$$

where E^t is the skeleton kinetic energy and must satisfy

$$E^t < E_{\min}, \tag{3.20}$$

where $E_{\min}$ is a configurable threshold, closer to zero to accommodate noise in the feature space, for which key poses are considered.

Although this methodology is sufficient from the conceptually perspective, an upper bound E_u is necessary to exist, so that only one key pose is identify in a neighborhood. This fact is due to possible noisy input data, even though it was preprocessed. The behavior of the effect of considering this upper bound is intended to be similar to what happens in the hysteresis loop. Therefore, a new key pose is only determined if the skeleton kinetic energy, at a given frame, is higher than the upper bound. This also guarantees that there is, at least, one frame between two identified key poses, which is fundamental to the correct behavior of this approach.

Once more, based on experimental tests, the following features are considered in this approach:

$$\mathbf{F}_w^a = \left[v_{jd}^t \quad \omega_{id_p}^t \quad \theta_{id_p}^t \quad \Delta_k^t\right], \tag{3.21}$$

with size of $n_a \times [3\left((m - m_{\text{extd}}) + n_\theta + n_\Delta\right) + n_\omega]$, where n_a is the size of the activity example window, n_Δ is the considered number of normals to triangles formed by three joints and n_ω is the considered number of projected angular velocities.

The notion of static and dynamic features is crucial from this point on to motivate this feature selection approach. Key poses are determined by frames where the skeleton joints have zero movement or, in other words, zero velocity. In this sense, for these poses it makes little sense to consider dynamic features (e.g. joint velocities), since they are zero (or close to zero). However, static features are very discriminative in these poses and must be considered as such. On the other hand, dynamic features are very important in between key poses, since the body's motion occurs in such frames, whereas information provided by static features loses significance.

In the latter case, max-min dynamic features (i.e. v and ω) are selected, as explained in the previous section, where the window size is determined by two consecutive key poses. From the identified key poses (i.e. the first w_{i_f} and the last w_{i_l}), the considered static features (i.e. θ and Δ) are extracted. This means, not only a set of interpretable features may be extracted, but also the set itself has its own meaning and intuition appealing, in the context of discriminating an activity action. Therefore, the **main contribution** of this approach is not centered in the selected features themselves, but in the way they are combined and used to discriminate each activity. It may be used with other (possibly more discriminative) features, with the only requirement of them following the static/dynamic definition and organization.

In the present work, the following example vector is constructed:

$$f^a_{w_i} = \begin{bmatrix} f^{\text{static}}_{w_{i_f}} & f^{\text{dynamic}}_{w_i} & f^{\text{static}}_{w_{i_l}} \end{bmatrix}, \tag{3.22}$$

where $f^{\text{static}}_{w_{i_f}}$ and $f^{\text{static}}_{w_{i_l}}$ represent the selected static features of the first and last key poses identified of the window, given generically by

$$f^{\text{static}}_t = \begin{bmatrix} \theta^t_{id_p} & \Delta^t_k \end{bmatrix}, \tag{3.23}$$

and $f^{\text{dynamic}}_{w_i}$ represents the selected max-min dynamic features as

$$f^{\text{dynamic}}_{w_i} = \begin{bmatrix} (v^{\max}_{jd})_{w_i} & (\omega^{\max}_{id_p})_{w_i} & (v^{\min}_{jd})_{w_i} & (\omega^{\min}_{id_p})_{w_i} \end{bmatrix}. \tag{3.24}$$

The length of each vector is $[2 \cdot 3(n_\theta + n_\Delta) + (n_{\max} + n_{\min}) \cdot (3(m - m_{\text{extd}}) + n_\omega)]$. In Fig. 3.3, an example of a sequence of skeleton poses of a human waving with an arm is illustrated, as well as the plot of their respective kinetic energies.

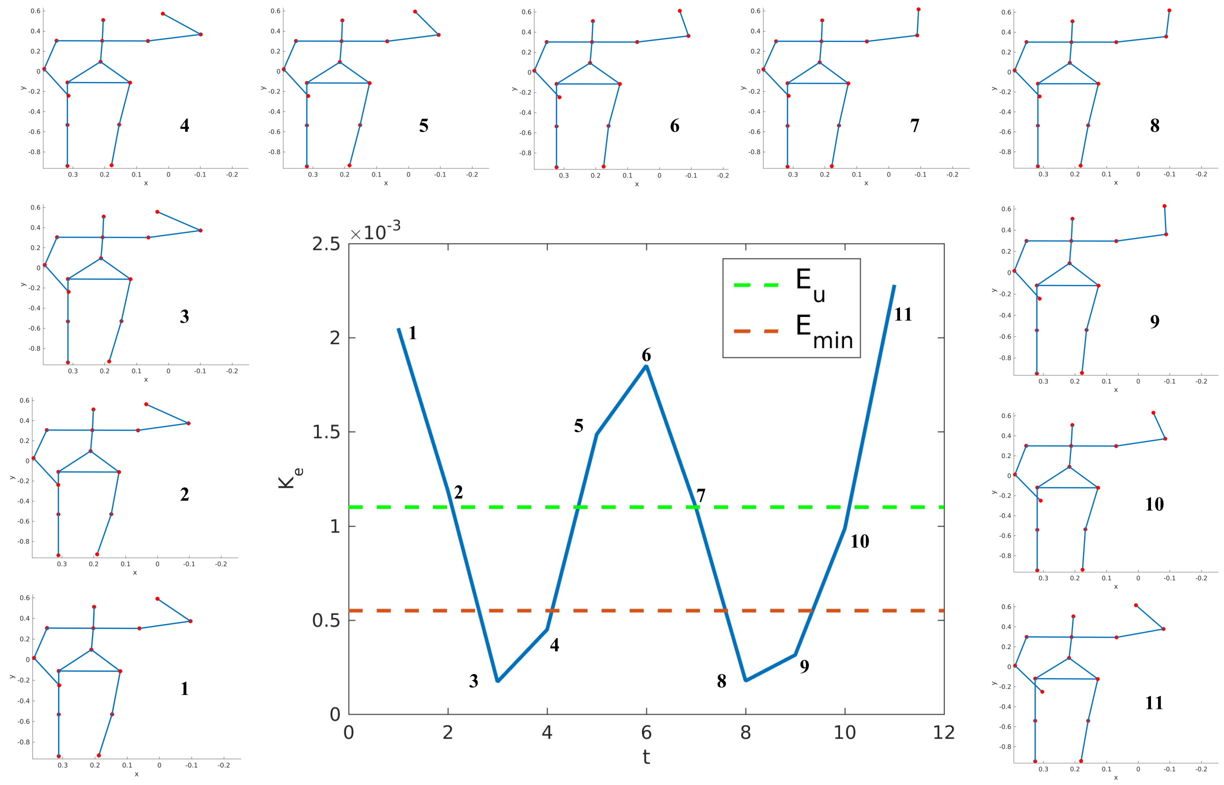


Figure 3.3: Example of a sequence of human poses and the plot of their respective kinetic energies. In this example, based on the marked thresholds, pose 3 and 8 are considered key poses. These are the extreme poses that will delimit a variable-size window (in this case, the size of the window is 6 frames), from which considered static features are selected. From pose 4 to 7, max-min dynamic features are selected. Note that, after pose 3 being classified as key pose, only from pose 5 a new key pose may be identified, since the kinetic energy of this pose is higher than the upper threshold (i.e. pose 4 can not be identified as key pose, even though its kinetic energy is lower that E_{min}).

4 Implemented Classifier - Random Forests

This chapter aims to describe the developed activity recognition classifier, as well as to justify all the assumptions and implementations decisions considered, with the global objective to design a very fast algorithm, requiring few training examples, with relevant overall results.

4.1 Decision Trees

In Sect. 2.2.1, the generic Classification and Regression Trees (CART) algorithm was introduced, as well as some background material supporting it. The answers to the six raised questions are fundamental to a good and successful CART implementation and are the basis for the developed algorithm. In this sense, the questions are hereby answered:

1. There will be two outcomes at a node (i.e. *binary* split); this means a node is divided into two branches (i.e. *branching factor* of 2), which are going to be commonly designated by *left* and *right* branches;
2. The tested property at a node will be based on the Entropy impurity, as stated in Eq. (2.1);
3. A node is considered a leaf, based on a confidence level of 90%, following the chi-squared statistics, as expressed in Eq. (2.6);
4. The tree is grown to the largest extent possible, without pruning;
5. Each leaf will contain all the label fractions reaching it;
6. Not considered.

The implemented CART algorithm was always viewed as the solid basis of a higher classifier (Random Forest (RF)): from the beginning of designing the architecture of the classifier, until the raw code implementation for its real-time application. Therefore, all these considerations are based on the assumption that they will be advantageous in the context of implementing a RF.

In Decision Trees (DT) projects, the more compact and lesser nodes and leaves necessary to describe a certain model, the better, since this means that, with less memory to store the model and consequently less iterations to reach a certain leaf, guarantying that it is still possible to predict the output with the same overall results. However, this property may be achieved, for example, by pruning the tree, which may increase the overall complexity of the training algorithm, having as a consequence an increase in the time needed for building the model. Essentially, it is a compromise between a simpler DT model and a simpler training algorithm.

In the context of the project of a RF, since it may be viewed as an ensemble of DT [25], a simpler training algorithm is preferable to a simpler model [1], since each tree is just trained with a smaller subset of variables (comparing to the original input). Another reason is that DT have relatively low bias if grown sufficiently deep, and since they are noisy, averaging them may benefit the overall result [6].

Therefore, the number of node's outcome is two, since it would require a significantly increase of complexity in the training stage to considered higher branching factors.

The test property on each node is based on the Entropy impurity function (Eq. 2.1), because of its computational simplicity, as well as its basis in information theory [35], which is useful in interpreting its outputs. In order to find the best split at a node, a Differential Evolution (DE) algorithm is used, where the objective function is the symmetrical of the drop in impurity given by (Eq. 2.4).

There are several methods for finding the best split, depending on if the test property for splitting a node is based on a single variable (i.e. *univariate trees*) or based on a function (e.g. linear combination, AND-OR propositions, etc) of all (or a subset of) the input variables (i.e. *multivariate trees*). Recall from Sect. 2.2.1, that DT allows its input variables to be numeric and/or non-numeric. In other words, its input may consist of *continuous* (e.g. real numbers) and/or *discrete* (e.g. integers, names, countries, etc) variables. It is assumed that each variable has its own known domain: for continuous ones it may be an interval between numbers; for discrete ones it may be a set of possible values. Considering first the discrete variables case, one possible method to split a node is to test, for each possible value in its domain, the corresponding instance and check for what branch it falls (i.e. if the variable of the instance is equal to the testing value, then the instance falls to the right branch;

[1]This sentence means that the time needed for training each DT is considered the most important attribute. Due to the characteristics of the RF, this is a valid assumption. Nevertheless, a simpler model is always preferable, whenever possible.

otherwise, it falls to the left branch). Then, using Eq. (2.4), find which discrete value maximizes the drop in impurity. A similar approach may be used for continuous variables, considering that its domain may be discretized. These two methods belong to the univariate trees category. For discrete values, this approach is very useful, clear and simple, providing the best splitting solution at a node.

However, the same does not apply for continuous variables, since their domain is discretized, giving origin to a compromise between splitting accuracy and time consumed in the search, which may be difficult to tune and may depend on the properties of the problem. Even so, assuming this compromise is optimized, this method does not guarantee the optimal splitting solution. This is not a problem by itself, since, as mentioned in Sect. 2.2.1, even the optimal splitting solution at each node does not guarantee the global optimal solution of the DT model. Nevertheless, a not desired characteristic is that it may produce more complex DT models, as shown in Fig. 4.1, if the training data does not match the splitting test condition.

In this case, for the same training input, a multivariate approach is preferable, since the data is better split by boundaries along the specified direction. This approach may be even seen as a generalization of the univariate one, due to the fact that it allows the division of the input space along the same directions (i.e. parallel directions along the axis). Therefore, this approach is preferable. However, its implementation may be more complicated, with greater computational costs if the methodology followed is similar to the one mentioned.

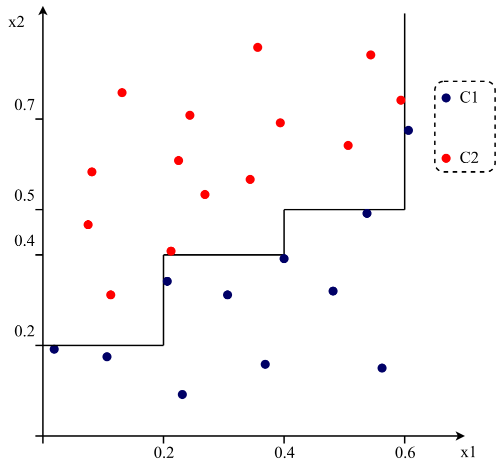
In this sense, the DE algorithm was chosen, in order to find the best splitting condition at each node. As discussed in Sect. 2.2.3, it does not guarantee the optimal solution to the problem, which is not an issue, as explained previously (see Sect. 2.2.1). It gives a sub-optimal solution in a desired amount of time, which is a very important property, since computational time is a crucial feature in the scope of this work. DE is used to find the best linear coefficients a_j ($j = \{1, \dots, n\}$) of a splitting condition of the form

$$a_1 x_1 + a_2 x_2 + \cdots + a_n x_n < a_0, \tag{4.1}$$

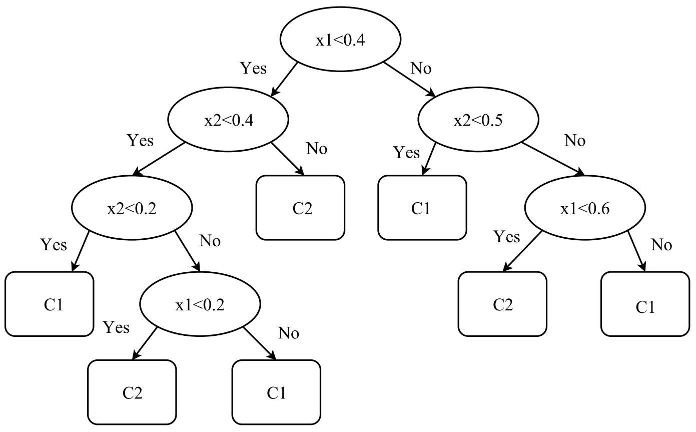
where x_j are the continuous training input variables, such that, at each node, the drop in impurity, given by Eq. (2.4), is maximized (or its symmetrical is minimized). In particular, a_0 is considered to be the constant value 1 and the population consists of N_{pop} individuals:

$$a_{t,i} = \begin{pmatrix} a_{t,i,1} & a_{t,i,2} & \dots & a_{t,i,n} \end{pmatrix}. \tag{4.2}$$

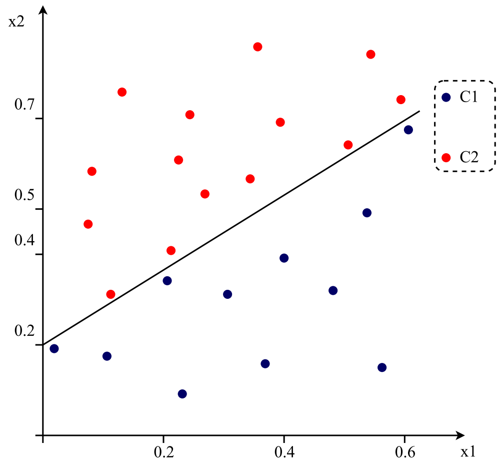
$a_{t,i,j}$ corresponds to the linear coefficient j, of the individual i at the iteration t. $F = 0.8$ and $c = 0.6$ values were chosen. DE was implemented based on the following pseudo-code:



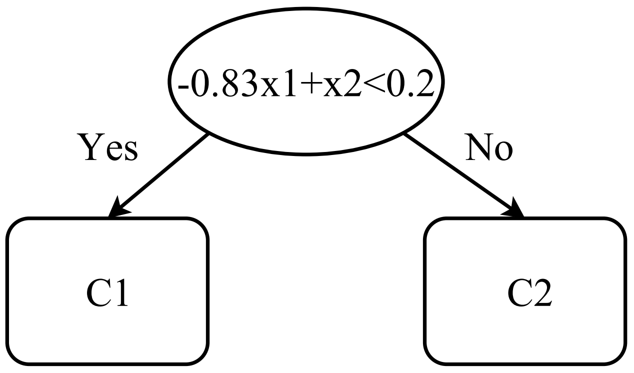
(a) Space division by a univariate DT.



(b) Univariate DT model.



(c) Space division by a multivariate DT.



(d) Multivariate DT model.

Figure 4.1: Univariate and Multivariate DT models and each influence on the division of the variables space. If the splitting test condition does not match the form of the input data, a more complicated tree model may result.

Algorithm 4 Implemented Differential Evolution Algorithm

```
generate initial population randomly;
for $t = 0$ to $N_{\text{iter}}$ do
    mutate each individual, accordingly to Eq. (2.10), with $F = 0.8$;
    recombine each individual, accordingly to Eq. (2.11), with $c = 0.6$;
    select best individuals, accordingly to Eq. (2.12), where $f(\cdot)$ is the symmetrical of the drop in impurity, given by Eq. (2.4);
end for
return best linear coefficients $a_j^*, j = \{1, \dots, n\}$;
```

The considered stopping condition for establishing a leaf is a very interesting approach, because there is no need for tunning thresholds or some other constants, which in practice may be very difficult and may depend on the training input and/or splitting conditions, for example. With this approach, the only requirement is to define the confidence level, look up the corresponding value in a chi-squared statistics table (e.g. for a confidence level of 90%, it corresponds a value of 2.71) and compare it with the computed value from Eq. 2.6. Also, there is an appealing statistical interpretation, which is explained in Sect. 2.2.1.

In each leaf, the fractions of all classes that reach them are stored. This compromise allows some averaging variants, in RF, which aim for better overall results: instead of considering that each DT has only one labeling vote, it may be considered that this one vote is divided into the equivalent fractions of labels, reaching the corresponding leaf of the tree; the fractions of all the trees are summed and the most voted label is returned. The rationale behind this approach is explained, considering the following example. Suppose the classifier consists of a RF with three DT and, for a given input, the resulting output is as follows:

$$\begin{array}{cccc} & C_1 & C_2 & C_3 \\ \mathrm{DT}_1 & 0.33 & 0.33 & 0.34 \\ \mathrm{DT}_2 & 0.33 & 0.33 & 0.34 \\ \mathrm{DT}_3 & 0.8 & 0.1 & 0.1 \\ \text{Majority vote} & 1 & 0 & 2 \\ \text{Summing vote} & 1.46 & 0.76 & 0.78 \end{array} . \tag{4.3}$$

DT_1 and DT_2 clearly vote equally for all three classes, which is the same as saying there is no class with an obvious majority. However, in the case of majority vote, they would vote in class C_3, which would be the most voted. On the other hand, DT_3 has a clear class choice, which, in the mentioned case, would make no difference. In the summing approach, all the uncertainty associated with the fractions of classes reaching a node would make no difference, since the most voted class in this case would be C_1, which is the clear choice of DT_3. This approach favors DT which have little or no uncertainty about its decision.

Another possible approach, which was thought but not implemented (see Sect. 6), was to have *specialized* DT within the set of DT of a RF classifier. This could lead to attribute weights to each DT voting, favoring the votes of specialized DT (e.g. multiplying its votes by a positive constant higher than one). The rationale of this approach was that specialized DT would not have much uncertainty in its voting results, allowing more accurate results. The CART algorithm was implemented in C++, based on the previous properties considerations, resulting in the following pseudo-code algorithm:

Algorithm 5 Implemented CART Algorithm

```
find best split rule, accordingly to DE algorithm 4;
if split is advantageous, accordingly to Eq. 2.6 then
    for branches: left and right do
        find instances falling in branch;
        generate tree using CART algorithm 5;
    end for
else
    create leaf, with reaching labels;
    return ;
end if
return tree model $T$;
```

4.2 Random Forests

The implemented CART algorithm for building DT is the backbone of the developed training algorithm. All of its characteristics are intended to provide the best possible features to the RF implementation, as well as to future considered RF implementations. Therefore, there are only two parameters to design: the number of DT and the number of picked variables at random for each node splitting in each DT, based on the training input dataset. As mentioned in Sect. 2.2.2, RF do not overfit and the number of trees considered is $N_{\text{tree}} = 100$, because preliminary experimental results showed that for a greater number of trees the overall results did not improve significantly. The number of variables was chosen to be the integer of $\log_2(n_{input}) + 1$, where n_{input} is the number of input variables in the training set. Based on the mentioned considerations, the following RF algorithm was implemented:

Algorithm 6 Generic Random Forests Algorithm

```
for $i = 1$ to $N_{\text{tree}}$ do
    draw a bootstrap sample $F_i^*$, from the training input;
    grow a tree $T_i$ recursively, using CART algorithm 5 and selecting $\text{int}(\log_2(n_{\text{input}}) + 1)$
variables at random to split each node, from $F_i^*$;
end for
return set of trees $\{T_i\}_1^{N_{\text{tree}}}$;
```

5 Experimental Results

To evaluate the proposed framework, a set of experiments were performed. Two distinct datasets were used: CAD-60 [39] and a custom dataset obtained in our lab. The implementation of the proposed approaches was done in Matlab, for validation purposes using the Random Forest (RF) training algorithm provided by Weka Version 3-6-13 software [7]. Then, in ROS environment (C++ language), both features selection approach II and RF classifier described in Chap. 4 were implemented. The experiments were performed in a 2.60 GHz Intel Core i5 CPU machine. In Fig. 3.2a, the base human skeleton is illustrated for both considered datasets, as well as the corresponding joint's indexes.

Table 5.1: Considered Angles

Angle	Joints Triplet	Angle	Joints Triplet	Angle	Joints Triplet
i	(a, b, c)	i	(a, b, c)	i	(a, b, c)
1	(6,7,13)	2	(4,5,12)	3	(6,10,11)
4	(4,8,9)	5	(10,11,15)	6	(8,9,14)
7	(6,10,13)	8	(4,8,12)	9	(1,7,13)
10	(1,5,12)	11	(3,12,13)	12	(3,14,15)

Table 5.2: Considered Normal to Triangles Formed by Three Joints

Normal	Joints Triplet	Normal	Joints Triplet
k	(a, b, c)	k	(a, b, c)
1	(4,5,12)	2	(6,7,13)
3	(8,9,14)	4	(10,11,15)
5	(1,12,13)	6	(3,12,13)
7	(5,8,12)	8	(7,10,13)

In the subsequent sections, the considered joint angles, normals to triangles formed by triplets of joints and distances between joints are the ones presented in Tables 5.1, 5.2 and 5.3, respectively. This was the set of features that provided better overall results in some preliminary experiments. Different combination of features were evaluated, but due to space limitations they are not presented here. In the particular case of the considered angles,

Table 5.3: Considered Distances Between Joints

Joints	Joints	Joints	Joints	Joints	Joints	Joints
(a,b)	(a,b)	(a,b)	(a,b)	(a,b)	(a,b)	(a,b)
(1,4)	(1,5)	(1,6)	(1,7)	(1,8)	(1,9)	(1,10)
(1,11)	(1,12)	(1,13)	(1,14)	(1,15)	(2,5)	(2,7)
(2,8)	(2,9)	(2,10)	(2,11)	(2,12)	(2,13)	(2,14)
(2,15)	(3,5)	(3,7)	(3,9)	(3,11)	(3,12)	(3,13)
(3,14)	(3,15)	(4,7)	(4,9)	(4,11)	(4,12)	(4,13)
(4,14)	(4,15)	(5,6)	(5,8)	(5,9)	(5,10)	(5,11)
(5,13)	(5,14)	(5,15)	(6,9)	(6,11)	(6,12)	(6,13)
(6,14)	(6,15)	(7,8)	(7,9)	(7,10)	(7,11)	(7,12)
(7,14)	(7,15)	(8,12)	(8,13)	(9,12)	(9,13)	(10,12)
(10,13)	(11,12)	(11,13)	(12,14)	(12,15)	(13,14)	(13,15)

the work of Faria *et al.* [21] was very influential, due to the supporting rationale and the obtained results. In the case of the considered distances, those between adjacent joints are not considered, since they are the same for every activity (i.e. not discriminative features).

The performance indicators in terms of precision (Prec) and recall (Rec) are presented for each scenario, adopting the same strategy described in [19]. A leave-one-out cross validation procedure was employed (e.g. the classifier is trained by all the subjects, except one, and tested using the remaining one). This procedure is important to check the classifier's generalization capability.

5.1 Cornell Activity Dataset

The CAD-60 consists of 3D skeleton's coordinates joints, which were acquired by a RGB-D sensor, at a frame rate of 30 Hz. The dataset contains 12 distinct human activities: *talking on phone*, *writing on board*, *drinking water*, *rinsing mouth with water*, *brushing teeth*, *wearing lens*, *talking on couch*, *relaxing on couch*, *chopping*, *stirring*, *opening container* and *working on computer*. In order to simulate real situations, a *random* action and a *still* posture were also added. These activities are categorized accordingly to 5 environments (*bathroom*, *bedroom*, *kitchen*, *living room* and *office*), depending on their corresponding context, and are performed by 4 different subjects.

Table 5.4: Performance of the Approach I on the CAD-60

Location	Activity	Weka's RF	
		Prec (%)	Rec (%)
Bathroom	random+still	86.55	92.12
	rinsing water	91.67	80.05
	brushing teeth	98.92	95.25
	wearing lens	93.75	83.02
	average	92.72	87.61
Bedroom	random+still	93.20	99.00
	talking on phone	88.00	77.12
	drinking water	77.50	84.47
	opening container	71.67	60.95
	average	82.59	80.38
Kitchen	random+still	91.47	96.75
	drinking water	99.07	100
	chopping	91.97	96.65
	stirring	99.07	92.22
	opening container	72.05	65.65
	average	90.73	90.25
Living room	random+still	95.75	98.87
	talking on phone	73.40	55.75
	drinking water	75.80	75.65
	talking on couch	89.00	94.22
	relaxing on couch	70.00	71.87
	average	80.79	79.27
Office	random+still	93.25	97.75
	talking on phone	79.40	66.95
	writing on board	96.50	94.90
	drinking water	87.27	73.40
	working on computer	100	100
	average	91.28	86.6
	Overall Average	**87.62**	**84.82**

5.1.1 Results and Analysis - Selection Approach I

The experimental results obtained for approach I are presented in Table 5.4. The overall average for precision was 87.62% and for recall was 84.82%, using the following parameters:

- $n_{f_r} = 120$; each activity is roughly divided into examples of size n_{f_f} frames;
- $m_{\text{extd}} = 3$; the corresponding velocities of *head*, *neck* and *torso* are not considered, since, after all the preprocessing steps, they are not significantly discriminative (i.e. their values is approximately zero);
- $n_\theta = 12$; the considered angles are presented in Table 5.1;

- $n_\delta = 70$; the considered distances between joints are the ones shown in Table 5.3;
- $n_{\max} = n_{\min} = 1$; only the most extreme values for each feature on the respective analysis window are considered.

Based on these parameters, the classifier was trained with an average number of 386.7 examples, each one with 566 features, requiring an average training time of 611 ms. This approach is not feasible for real-time applications (as explained in Sect. 3.4) and therefore only Weka's RF classifier was used. These results show that with few training examples and with only max-min skeleton-based features, it is possible to obtain relevant discrimination between distinct activities with very short training times. Therefore, the proposed approach of segmenting each activity into a set of actions is valid.

5.1.2 Results and Analysis - Selection Approach II

Based on the previous results for selection approach I, a more robust method was implemented (i.e. selection approach II, as described in Sect. 3.5). The experimental results obtained using this selection approach are presented in Table 5.5. Both Weka's RF classifier, with overall average for precision and recall of 81.73% and 79.01%, respectively, and the developed one (DRF), with overall average for precision and recall of 82.58% and 80.79%, respectively, were used. The following features extraction parameters were implemented for both tests:

- $m_{\text{extd}} = 3$; the corresponding velocities of *head*, *neck* and *torso* are again not considered;
- $n_\theta = 12$; the considered angles are presented in Table 5.1;
- $n_\omega = 12$; the considered angular velocities are obtained based on their respective angles;
- $n_\Delta = 8$; the considered normals to triangles formed by triplets of joints are shown in Table 5.2;
- $n_{\max} = n_{\min} = 1$; only the most extreme values for each feature on the respective analysis window are considered;
- $E_{\min} = 0.0028$; this value was tuned empirically, based on experimental tests on the training data;
- $E_u^a = 2\times\text{mean}(E^a)$; mean$(\cdot)$ is the mean function and $E^a = \{(E^1)^a, (E^2)^a, \ldots, (E^t)^a, \ldots\}$ is the set of kinetic energy values of the activity a, for all its corresponding frames; this value was also tuned empirically, based on experimental tests on the training data.

For the DRF implementation, the following parameters were used:

Table 5.5: Performance of the Approach II on the CAD-60

Location	Activity	Weka's RF		Developed Random Forest (DRF)	
		Prec (%)	Rec (%)	Prec (%)	Rec (%)
Bathroom	random+still	95.87	96.60	96.85	100
	rinsing water	81.57	68.32	100	65.47
	brushing teeth	96.05	92.97	93.47	99.25
	wearing lens	84.05	85.35	97.72	83.35
	average	89.38	85.81	97.01	87.02
Bedroom	random+still	97.87	99.62	99.20	100
	talking on phone	56.05	66.50	82.27	82.50
	drinking water	57.15	36.85	0	0
	opening container	100	94.35	100	91.27
	average	77.77	74.33	70.37	68.44
Kitchen	random+still	93.00	98.47	99.27	100
	drinking water	99.00	95.82	75.00	75.00
	chopping	83.77	92.50	100	100
	stirring	73.07	64.80	100	100
	opening container	100	86.32	100	96.67
	average	89.77	87.58	94.85	94.33
Living room	random+still	96.70	99.62	100	100
	talking on phone	59.82	75.07	82.60	78.02
	drinking water	58.15	33.42	38.10	52.95
	talking on couch	81.25	85.72	78.30	95.82
	relaxing on couch	75.00	62.50	91.67	76.42
	average	74.18	71.27	78.13	80.64
Office	random+still	94.60	96.90	98.55	99.70
	talking on phone	49.72	71.02	75.37	91.67
	writing on board	92.17	90.87	88.90	76.20
	drinking water	51.32	21.55	0	0
	working on computer	100	100	100	100
	average	77.56	76.07	72.56	73.51
	Overall Average	**81.73**	**79.01**	**82.58**	**80.79**

- multivariate trees; each node is split accordingly to Eq. 4.1, based on Differential Evolution (DE), where $N_{\text{pop}} = 12$ and $N_{\text{iter}} = 6$;
- a node is stopped splitting using Eq. 2.6, with a confidence level of 90% (which corresponds a value of 2.71);
- an activity is only classified with at least 45% of the total votes (e.g. with 100 Decision Trees (DT), there must be at least 45 votes in the same activity, so that its classification can be considered); otherwise, it is considered to be an *unknown* activity; this value may be interpreted as the confidence in the classification and is based on experimental tests.

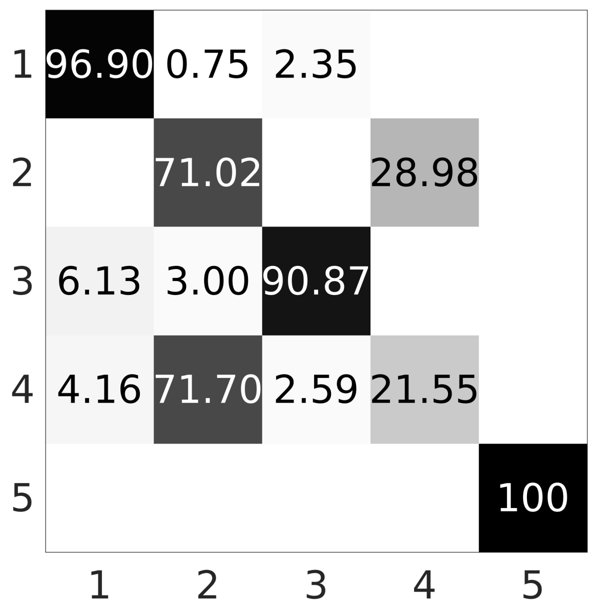
Based on these parameters, the classifiers were trained with an average number of 630.30 examples, each one with 264 features, requiring an average training time of 920 ms, for the Weka's algorithm, and 2.36 s, for the developed one. Comparing both implementations, there is a slightly increase of overall performance, using the DRF, at the cost of an increase on the required training time. Nevertheless, a required 2.36 s of training time is assumed to be also suitable for real-time applications, since the training will only occur when a new action, not included in the known actions, is detected and added to the system.

However, comparing these results with the ones obtained with method I, they are inferior. There are several possible explanations for this fact, which open some lines of future research. Although these explanations are discussed in more detail in a subsequent section, next two of the most important ones are highlighted:

- the global skeleton's kinetic energy is calculated, based on the velocities of all joints; this means that for a key pose to occur, all joints must have zero velocity; however, not all joints are used when a human performs an activity (e.g. the main focus on *drinking water* is centered in the upper joints; nonetheless, the velocities of the lower joints are considered equally important when computing the skeleton's kinetic energy); this could lead to the interference of secondary motions in the computation of the actual body's key poses (e.g. noisy data; non-important joints movements; etc); therefore, important key poses that could discriminate an activity may not be correctly identified;
- $E_{\min}$ and E_u values should not be fixed thresholds, since each activity has its own motion pattern; this means that more instances can be obtained for less dynamic activities (e.g. *working on computer*), compared to more dynamic ones (e.g. *opening container*).

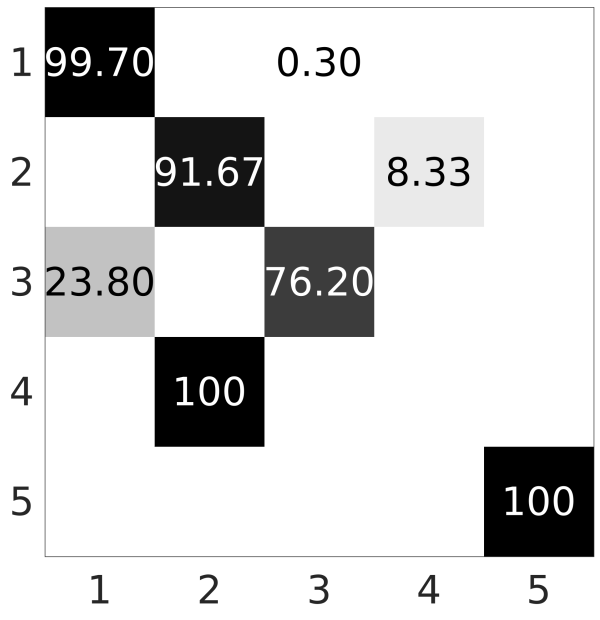
Nevertheless, this approach has shown relevant overall results, capable of making distinctions between almost considered activities. However, in this particular dataset, this approach can not distinguish between two similar[1] activities: *drinking water* and *talking on phone*. This fact is illustrated in Fig. 5.1, where two confusion matrices are presented for the office environment activities, for both considered classifiers.

[1]In this context, two activities are similar if the corresponding skeleton's joints movements do not differ significantly, from the extracted features perspective (e.g. similar max-min joints velocities, similar distances between joints in key poses, etc). In this sense, for example, the *drinking water* activity is similar to the *talking on phone* activity, since an arm is raised near the head in both of them, even though different objects are used in their execution (see Sect. 6 for a more detailed discussion about this issue).



	1	2	3	4	5
1	96.90	0.75	2.35		
2		71.02		28.98	
3	6.13	3.00	90.87		
4	4.16	71.70	2.59	21.55	
5					100

(a) Confusion matrix obtained with Weka's RF.



	1	2	3	4	5
1	99.70		0.30		
2		91.67		8.33	
3	23.80		76.20		
4		100			
5					100

(b) Confusion matrix obtained with DRF.

Figure 5.1: Confusion matrices obtained for office environment activities (1-*random+still*; 2-*talking on phone*; 3-*writing on board*; 4-*drinking water*; 5-*working on computer*). As shown, *drinking water* and *talking on phone* activities are not very well discriminated between themselves: using both classifiers, the *drinking water* activity is mainly classified as *talking on phone*. The remaining activities are considered to be relatively well classified.

In Table 5.6, a comparison of the overall results of the proposed selection approach II, in relation to other state-of-the-art methods, is presented. The comparison that was made is only based on precision and recall indicators, since no performance indicators such as training examples, features or training time are provided by the other considered methods. In this sense, the proposed approach shows relevant overall performance, comparing to other state-of-the-art methods, particularly considering that the number of training examples and required training time are very small.

From this point on, all the results are obtained using the selection approach II, since it was specifically developed for real-time applications. All the implemented parameters are as mentioned previously, with the exception of some referred ones, that occasionally needed to be tuned.

Table 5.6: Comparison of the Approach II with Other State-of-the-art Methods

Method	Prec (%)	Rec (%)
Faria *et al.* [21]	94.8	94.7
Shan and Akella [20]	93.8	94.5
Zhu, Chen and Guo [40]	93.2	84.6
Zhang and Tian [41]	86.0	84.0
Selection II - DRF	**82.6**	**80.8**
Koppula, Gupta and Saxena [23]	80.8	71.4
Gupta, Chia and Rajan [16]	78.1	75.4
Ni *et al.* [42]	75.9	69.5
Yang and Tian [18]	71.9	66.6
Piyathilaka and Kodagoda [43]	70.0	78.0
Sung *et al.* [39]	67.9	55.5

5.2 Developed Dataset

The Developed Dataset (DD) consists of 3D skeleton's coordinates joints, containing the performance of 8 distinct human activities: *falling*, *waving with arm*, *follow me*, *walking*, *jumping*, *standing up*, *sitting down* and *still*. These were performed by 7 different subjects in the same environment. The 3D joint coordinates were acquired by a RGB-D sensor (Microsoft Kinect), at a frame rate of 30 Hz, using the OpenNi Tracker algorithm. Figure 5.2 illustrates an overview of the implemented structure in ROS environment, which was developed to acquire real-time 3D skeleton data, in order to train and validate the proposed framework.

These activities are distinctive between themselves (with the possible exception between *waving with arm* and *follow me* activities), with the objective of considering real-time situations, for which there is the need of fast activity classification and consequent decision making. The focus of the work is related to the fastest and more accurate classification possible, as the following results corroborate.

5.2.1 Results and Analysis

The experimental results obtained for the DD are presented in Table 5.7, where an overall average for precision of 85.71% and 89.01% and for recall of 83.28% and 79.01%, were obtained, using Weka's RF and DRF, respectively. The following parameters were altered:

- E_{min} is determined based on an averaging of the previous skeleton kinetic energies, instead of a predefined fixed threshold;

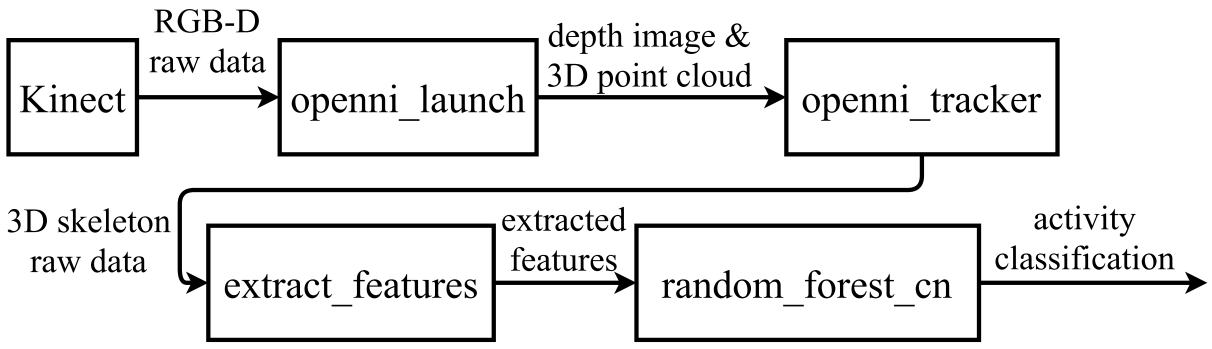


Figure 5.2: Overview of ROS implementation. RGB-D raw data is acquired by Microsoft Kinect camera [11]. The *openni_launch* node is used to convert the acquired raw data to depth images and point clouds, which are used by *openni_tracker* node to detect and track human skeletons. From the 3D skeleton raw data, the *extract_features* node computes and selects all considered features (see Sect. 3). Then, a RF algorithm classifies the set of extracted features, based on a trained model (*random_forest_cn* node).

Table 5.7: Performance of the Approach II on the DD

Activity	Weka's RF		DRF	
	Prec (%)	Rec (%)	Prec (%)	Rec (%)
falling	88.37	91.16	100	65.65
waving with arm	75.74	79.29	78.61	67.91
follow me	77.56	79.83	87.13	75.97
walking	84.34	95.49	87.34	95.41
jumping	100	70.66	100	68.27
standing up	74.60	89.51	83.64	98.11
sitting down	93.83	72.73	95.24	63.57
still	91.21	87.60	80.10	97.20
Overall Average	**85.71**	**83.28**	**89.01**	**79.01**

- each activity is classified with at least 25% of the total votes (since there are more activities to classify at the same time).

The classifiers were trained with an average number of 1272 examples, each one with 264 features, requiring an average training time of 2 s, for the Weka's algorithm, and 4.5 s, for the developed one. Weka's RF registered a higher recall rate, whereas DRF registered a higher precision rate. This is a very interesting observation and may be mainly explained due to the fact that DRF only classifies a certain activity if it has 25% of the total votes. This means, DRF possibly trades exactness for completeness, in the sense that it has more certainty, when classifying. Although not enough tests were done to corroborate this observation, there may be some applications, where it is better to not classify, when the classification certainty is lesser than a desired confidence level, rather to classify wrongly.

Although the activities of the DD are almost all distinctive between themselves, which could lead to an increase of the overall performance, there are more activities to classify and to be considered. Therefore, there is an increase of possible misclassification (i.e. the random correct classification decreases), just based on probabilistic terms.

Table 5.8: Performance of the Approach II on the DD Without Normalization

Activity	Weka's RF		DRF	
	Prec (%)	Rec (%)	Prec (%)	Rec (%)
falling	86.13	87.59	97.14	71.60
waving with arm	73.19	78.51	100	79.31
follow me	74.33	79.36	84.29	69.81
walking	82.80	93.96	87.03	96.01
jumping	85.71	67.97	100	85.24
standing up	89.87	88.76	84.57	95.54
sitting down	91.73	78.11	97.14	71.24
still	91.44	87.33	87.37	91.54
Overall Average	**84.42**	**82.34**	**87.61**	**80.83**

In Table 5.8, the experimental results for the exact same conditions as mentioned previously are shown, with the exception that the extracted features were not normalized accordingly to Eq. (3.14). An overall average for precision of 84.42% and 87.61% and for recall of 82.34% and 80.83%, using Weka's RF and DRF, respectively, were obtained. These results show that the normalization stage is optional, when using the RF classifier, even though it is a good practice. This means, saving program memory is possible, since, in this case, the maximum and minimum values of the training set for each feature do not need to be stored. Also, the normalization stage ceases to exist, cutting computational time. Depending on the concrete application and its requirements, this is a possibility.

6 Conclusion and Future Work

The presented work may be separated into two core components: the proposed skeleton-based **features extraction approach** and the **Developed Random Forest (DRF)** algorithm classifier, built and implemented from scratch. The main objective of this work was to develop a very fast training framework, requiring few training examples, with relevant performance, comparing to other state-of-the-art methods. The development of the two mentioned components allowed a full control of their behavior, maximizing their cooperation and engagement, in order to reach the main assumed goal.

The proposed features extraction approach has some contributions. First, a preprocessing stage is employed, where the raw joints coordinates of a human skeleton are converted into an invariable-oriented, normalized skeleton, as well as on its corresponding symmetrical. This stage aims to attenuate the natural variability introduced by morphological differences between different users, by forming a single universal 3D skeleton, where only the execution of an activity causes significantly variability in the skeleton joints position, which is the starting point to begin human activity recognition. Second, a set of skeleton-based features is proposed, such as joints velocities, projected angles and normal to planes formed by three joints. These features, although simple and low-level, evidence good discriminative properties. Third, two features selection approaches are proposed, with the same basic idea: fragmenting each activity into a sequence of windows, which can be used to classify the related activity. The selection approach II divides each activity into variable-size window actions, where their delimiters are frames where the skeleton has zero kinetic energy. These frames correspond to extreme joints position in the motion path and may be used to identify transitions.

Moreover, the developed classifier has some interesting properties. First, there are no thresholds to tune. The only definable parameters are: the confidence level that determines each node to be split, which has an appealing statistical interpretation; the number of individuals of the Differential Evolution (DE) population and the respective number of computational iterations (there must be a compromise between computational time and the best solution found); the minimum certainty from which an activity is classified. In this context, certainty is based on how many Decision Trees (DT) vote for the same activity, from the set that comprises the Random Forest (RF). Second, each node splitting condition is based on the set of input variables of each DT, without a significantly increase in computational algorithmic complexity or time. This means that the splitting condition may separate the input data in any direction, allowing a simpler and more compact DT grow.

These methodologies were implemented using Matlab, Weka Sofware and C++ ROS environment, obtaining relevant overall results, in terms of precision and recall, comparing to other state-of-art methods. Considering also the number of required training examples and training time, the proposed approach showed good indicators. Additionally, the work developed opened several new research directions, aiming to bridge its drawbacks and improve its characteristics. A summary of the main contributions is highlighted:

- no frame-by-frame training/classification: activity segmentation based on key poses detection;
- extraction of just extremal skeleton information, based on max-min features;
- development of a RF classifier in C++, with all the discussed properties.

The main issue with the proposed features selection method, highlighted by the attained results, is the fact that it does not provide sufficient information to discriminate between similar activities (as seen in Sect. 5.1.2). Particularly in CAD-60, the activities *drinking water* and *talking on phone* were misclassified between themselves. From the human perspective, these are very discriminative activities; why does the method confuse them?

Recall that only skeleton-based information is acquired. In this sense, there are no context information regarding the activity execution: a *glass of water* is needed to perform the *drinking water* activity; a *phone* is needed to perform the *talking on phone* activity. Figure 6.1 shows a comparison between the mentioned activities of the provided input skeleton data, from the camera's perspective view (i.e. raw 3D skeleton's joints coordinates). As illustrated, without context, both activities are very similar, since both are executed with an arm raised near the head. Therefore, for example, a parallel object classifier system would improve considerably the performance of the human activity recognition framework. Once an object is identified, it can be provided as one additional input feature, which, in the considered case, would clearly disambiguate between the two activities. Nevertheless, based on this example, a general context perception system would prove its usefulness, since the majority

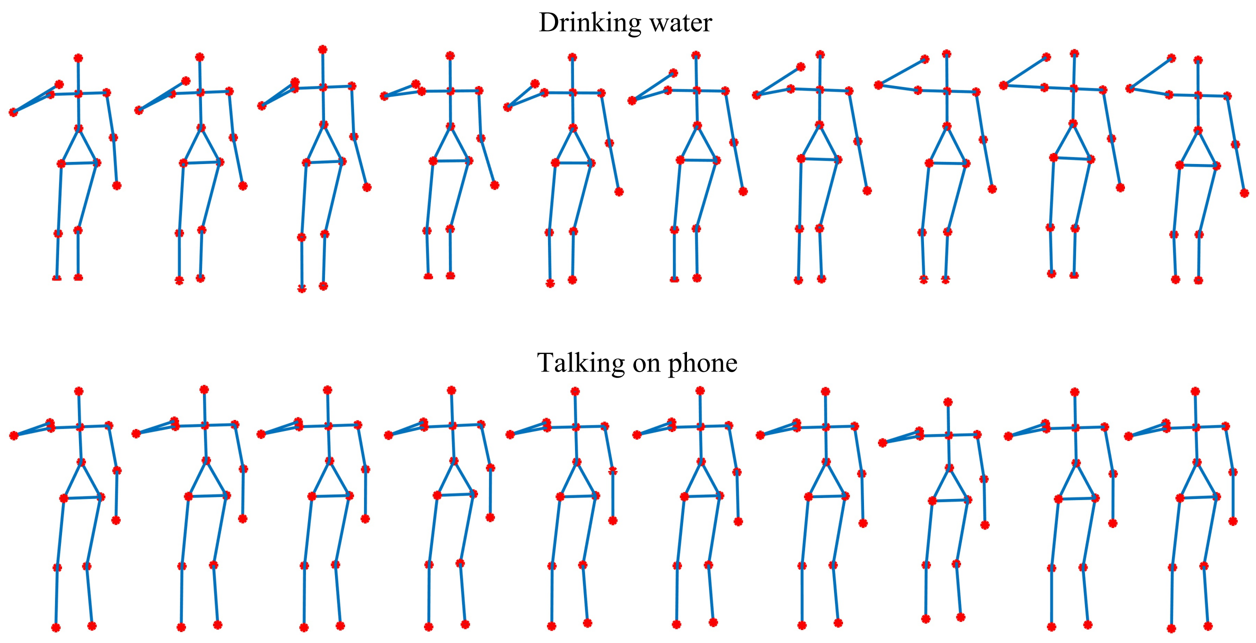


Figure 6.1: *Drinking water* and *talking on phone* activities comparison. Each illustrated human skeleton corresponds to a single acquired frame. The sequence of frames, from left to right, corresponds to the acquired data that represents the correspondent activity.

of activities are performed within a certain context, with a meaningful purpose/objective (e.g. *drinking water* quenches thirst, whereas *talking on phone* establishes a communication channel between two or more subjects; not the other way around). Therefore, context must represent a crucial part when recognizing human activities.

Another problem with the method consists on the two thresholds needed to identify a key pose: $E_{\min}$ and E_u. First, E_u was used to solve a problem, that appeared in some activities, consisting in the identification of many consecutive (or close) key poses, introducing some hysteresis behavior. However, another problem arise: what value should it be? fixed or variable? Also, in the experimental tests, the optimal value would change, depending on the activity (this is the reason why it was chosen $E_u^a = 2 \times \text{mean}(E^a)$). Although this practical approach solved the problem, finding the parameter value became a tedious process and it does not guarantee optimality. The same argument is valid for $E_{\min}$, which was also a tuned value. Second, since the camera's sensor is not perfect (i.e. the 3D data acquired is noisy), in certain activities, there were secondary movements, which do not contribute with relevant information for their recognition (e.g. leg movements performing *brushing teeth* activity). Therefore, since the skeleton's kinetic energy is calculated based on all the joints, there could be misidentified key poses (or not identified at all).

One possible research direction is to consider the division of the human skeleton into several parts (e.g. arms, legs, upper body, lower body, etc). This idea allows some simplifications of the proposed approach, since the tuning of the $E_{\min}$ value would be just applicable for parts of the skeleton. Consequently, the E_u threshold might not be necessary, and only the tunning of one would be required. Also, not all joints contribute with the same relevant information: it may depend on the performed activity. With the skeleton division, each part could contribute with weighted information, depending on their level of relevant contribution to discriminate between activities. Therefore, specialized classifiers could be trained based on each part, having a higher-level classifier combining all these informations, in order to improve the method's performance.

Concluding this work, a summary of some opened new research directions is highlighted:

- incorporate a parallel context perception system;
- divide the human skeleton into several parts, differentiating parts with more/less relevant information;
- train specialized classifiers (e.g. RF) for each considered part, having a higher-level classifier discriminating between activities;
- consider activities transitions as key poses.

7 Bibliography

[1] United Nations. World Population Ageing, 2013. *Economic & Social Affairs*, 2014.

[2] Antonio W Vieira, Erickson R Nascimento, Gabriel L Oliveira, Zicheng Liu, and Mario FM Campos. Stop: Space-Time Occupancy Patterns for 3D Action Recognition from Depth Map Sequences. In *Iberoamerican Congress on Pattern Recognition*, pages 252–259. Springer, 2012.

[3] Jamie Shotton, Toby Sharp, Alex Kipman, Andrew Fitzgibbon, Mark Finocchio, Andrew Blake, Mat Cook, and Richard Moore. Real-time Human Pose Recognition in Parts from Single Depth Images. *Communications of the ACM*, 56(1):116–124, 2013.

[4] Fei Han, Brian Reily, William Hoff, and Hao Zhang. Space-Time Representation of People Based on 3D Skeletal Data: A Review. *arXiv preprint arXiv:1601.01006*, 2016.

[5] Gunnar Johansson. Visual Perception of Biological Motion and a Model for its Analysis. *Perception & Psychophysics*, 14(2):201–211, 1973.

[6] Jerome Friedman, Trevor Hastie, and Robert Tibshirani. *The Elements of Statistical Learning*, volume 1. Springer series in statistics Springer, Berlin, 2001.

[7] Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, and Ian H Witten. The WEKA Data Mining Software: an Update. *ACM SIGKDD Explorations Newsletter*, 11(1):10–18, 2009.

[8] Bongjin Jun, Inho Choi, and Daijin Kim. Local Transform Features and Hybridization for Accurate Face and Human Detection. *IEEE transactions on pattern analysis and machine intelligence*, 35(6):1423–1436, 2013.

[9] Kei Okada, Takashi Ogura, Atsushi Haneda, Junya Fujimoto, Fabien Gravot, and Masayuki Inaba. Humanoid Motion Generation System on HRP2-JSK for Daily Life

Environment. In *IEEE International Conference Mechatronics and Automation, 2005*, volume 4, pages 1772–1777. IEEE, 2005.

[10] Yale Song, David Demirdjian, and Randall Davis. Continuous Body and Hand Gesture Recognition for Natural Human-Computer Interaction. *ACM Transactions on Interactive Intelligent Systems (TiiS)*, 2(1):5, 2012.

[11] Microsoft. Microsoft Kinect. http://dev.windows.com/en-us/kinect.

[12] Asus. Asus Xtion PRO LIVE. http://www.asus.com/Multimedia/XtionPro/.

[13] Jake K Aggarwal and Lu Xia. Human Activity Recognition from 3D Data: A Review. *Pattern Recognition Letters*, 48:70–80, 2014.

[14] Matteo Munaro and Emanuele Menegatti. Fast RGB-D People Tracking for Service Robots. *Autonomous Robots*, 37(3):227–242, 2014.

[15] Lu Xia, Chia-Chih Chen, and Jake K Aggarwal. Human Detection Using Depth Information by Kinect. In *CVPR 2011 WORKSHOPS*, pages 15–22. IEEE, 2011.

[16] Raj Gupta, Alex Yong-Sang Chia, and Deepu Rajan. Human Activities Recognition using Depth Images. In *Proceedings of the 21st ACM international conference on Multimedia*, pages 283–292. ACM, 2013.

[17] Chen Chen, Kui Liu, and Nasser Kehtarnavaz. Real-time Human Action Recognition Based on Depth Motion Maps. *Journal of real-time image processing*, pages 1–9, 2013.

[18] Xiaodong Yang and YingLi Tian. Effective 3D Action Recognition using Eigenjoints. *Journal of Visual Communication and Image Representation*, 25(1):2–11, 2014.

[19] Jaeyong Sung, Colin Ponce, Bart Selman, and Ashutosh Saxena. Unstructured Human Activity Detection from RGBD Images. In *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, pages 842–849. IEEE, 2012.

[20] Junjie Shan and Srinivas Akella. 3D Human Action Segmentation and Recognition using Pose Kinetic Energy. In *Advanced Robotics and its Social Impacts (ARSO), 2014 IEEE Workshop on*, pages 69–75. IEEE, 2014.

[21] Diego R Faria, Mario Vieira, Cristiano Premebida, and Urbano Nunes. Probabilistic Human Daily Activity Recognition Towards Robot-Assisted Living. In *Robot and Hu-*

man Interactive Communication (RO-MAN), 2015 24th IEEE International Symposium on, pages 582–587. IEEE, 2015.

[22] Wenwen Ding, Kai Liu, Xujia Fu, and Fei Cheng. Profile hmms for skeleton-based human action recognition. *Signal Processing: Image Communication*, 42:109–119, 2016.

[23] Hema Swetha Koppula, Rudhir Gupta, and Ashutosh Saxena. Learning Human Activities and Object Affordances from RGB-D Videos. *The International Journal of Robotics Research*, 32(8):951–970, 2013.

[24] Guangming Zhu, Liang Zhang, Peiyi Shen, and Juan Song. Human Action Recognition using Multi-layer Codebooks of Key Poses and Atomic Motions. *Signal Processing: Image Communication*, 2016.

[25] Leo Breiman. Random Forests. *Machine Learning*, 45(1):5–32, 2001.

[26] Gilles Louppe. Understanding Random Forests: From Theory to Practice. *arXiv preprint arXiv:1407.7502*, 2014.

[27] BIWI CVL. Efficient Real-time Pixelwise Object Class Labeling for Safe Human-Robot Collaboration in Industrial Domain. 2015.

[28] Ling Gan and Fu Chen. Human Action Recognition using APJ3D and Random Forests. *Journal of Software*, 8(9):2238–2245, 2013.

[29] David Demirdjian and Chenna Varri. Recognizing Events with Temporal Random Forests. In *Proceedings of the 2009 international conference on Multimodal interfaces*, pages 293–296. ACM, 2009.

[30] Leandro Miranda, Thales Vieira, Dimas Martinez, Thomas Lewiner, Antonio W Vieira, and Mario FM Campos. Real-time Gesture Recognition from Depth Data Through Key Poses Learning and Decision Forests. In *2012 25th SIBGRAPI Conference on Graphics, Patterns and Images*, pages 268–275. IEEE, 2012.

[31] Rainer Storn and Kenneth Price. Differential Evolution–A Simple and Efficient Heuristic for Global Optimization Over Continuous Spaces. *Journal of Global Optimization*, 11(4):341–359, 1997.

[32] Timothy Masters and Walker Land. A New Training Algorithm for the General Regression Neural Network. In *Systems, Man, and Cybernetics, 1997. Computational*

Cybernetics and Simulation., 1997 IEEE International Conference on, volume 3, pages 1990–1994. IEEE, 1997.

[33] Silvio Priem Mendes, Juan A Gomez Pulido, Miguel A Vega Rodriguez, Maria D Jaraiz Simon, and Juan M Sanchez Perez. A Differential Evolution Based Algorithm to Optimize the Radio Network Design Problem. In *2006 Second IEEE International Conference on e-Science and Grid Computing (e-Science'06)*, pages 119–119. IEEE, 2006.

[34] Ethem Alpaydin. *Introduction to Machine Learning*. MIT press, 2014.

[35] Richard O Duda, Peter E Hart, and David G Stork. *Pattern Classification*. John Wiley & Sons, 2012.

[36] Mark R Segal. Machine Learning Benchmarks and Random Forest Regression. *Center for Bioinformatics & Molecular Biostatistics*, 2004.

[37] Jiang Wang, Zicheng Liu, Ying Wu, and Junsong Yuan. Learning Actionlet Ensemble for 3D Human Action Recognition. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 36(5):914–927, 2014.

[38] Kai Guo. *Action Recognition using Log-Covariance Matrices of Silhouette and Optical-Flow Features*. Boston University, 2012.

[39] Jaeyong Sung, Colin Ponce, Bart Selman, and Ashutosh Saxena. Human Activity Detection from RGBD Images. *Plan, Activity, and Intent Recognition*, 64, 2011.

[40] Yu Zhu, Wenbin Chen, and Guodong Guo. Evaluating Spatiotemporal Interest Point Features for Depth-based Action Recognition. *Image and Vision Computing*, 32(8):453–464, 2014.

[41] Chenyang Zhang and Yingli Tian. RGB-D Camera-based Daily Living Activity Recognition. *Journal of Computer Vision and Image Processing*, 2(4):12, 2012.

[42] Bingbing Ni, Yong Pei, Pierre Moulin, and Shuicheng Yan. Multilevel Depth and Image Fusion for Human Activity Detection. *IEEE Transactions on Cybernetics*, 43(5):1383–1394, 2013.

[43] Lasitha Piyathilaka and Sarath Kodagoda. Gaussian Mixture Based HMM for Human Daily Activity Recognition using 3D Skeleton Features. In *2013 IEEE 8th Conference on Industrial Electronics and Applications (ICIEA)*, pages 567–572. IEEE, 2013.

Appendix A

Paper Accepted and Presented at BAILAR Workshop, IEEE RO-MAN 2016 Conference

Human Activity Recognition using Max-Min Skeleton-based Features and Key Poses

Urbano Miguel Nunes, Diego R. Faria, and Paulo Peixoto

Abstract—**Human activity recognition is still a very challenging research area, due to the inherently complex temporal and spatial patterns that characterize most human activities. This paper proposes a human activity recognition framework based on random forests, where each activity is classified requiring few training examples (i.e. no frame-by-frame activity classification). In a first approach, a simple mechanism that divides each action sequence into a fixed-size window is employed, where max-min skeleton-based features are extracted. In the second approach, each window is delimited by a pair of automatically detected key poses, where static and max-min dynamic features are extracted, based on the determined activity example. Both approaches are evaluated using the Cornell Activity Dataset [1], obtaining relevant overall average results, considering that these approaches are fast to train and require just a few training examples. These characteristics suggest that the proposed framework can be useful for real-time applications, where the activities are typically well distinctive and little training time is required, or to be integrated in larger and sophisticated systems, for a first quick impression/learning of certain activities.**

Index Terms—**Human Daily Activity Recognition, Random Forest, Max-Min Skeleton-based Features, Key Poses, Static and Dynamic Features**

I. Introduction

ROBOT perception is still an open area of research mainly due to the complexity that characterize the dynamic environment that surrounds the robot on real-world application scenarios. This is particularly true when the robot needs to interact with humans, like in the case of assistive robots, which should be able to quickly assess and react to potential critical situations. So, human activity recognition should play an important role on any autonomous robot perception module. In this context, this paper contributes with the proposal of two approaches for human activity recognition, both thought for real-time application scenarios, where characteristics like the number of training samples and the time needed for training play an important role. Simple max-min features are extracted, within a defined activity window, to train a random forest classifier. Few training examples are used to train this classifier. The main contributions of this work are the following:

- Two simple and effective approaches to extract extremal skeleton information, based on max-min features;
- Very fast training, requiring few training examples, properties that are real-time oriented.

These contributions, as well as the relevant performance obtained, when evaluated using *state of the art* dataset, may serve as a solid human activity recognition framework for real-time applications. The remainder of the paper is organized as follows: section II briefly describes the relevant related work, highlighting the contributions that the proposed approaches provide; section III explains both developed approaches for features extraction; section IV presents the results of the experimental procedures used to evaluate the proposed methods; section V summarizes the key ideas proposed, as well as to ongoing work, introducing some new lines of research for the near future.

II. Related Work

Human activity recognition has been a very active topic of research. Typically, recent approaches for human activity recognition rely on two sensing modalities: depth data and 3D joints position data. In [2], a method based on depth images and temporal ordering of unique poses is presented. In [3], a method to predict in real-time 3D body joints position from a single depth image from a RGB-D sensor was proposed. The most representative works based on 3D skeleton joints position data are the following: interaction of a subset of human joints [4]; eigenjoints descriptor, which incorporates static postures, motion and overall dynamics [5]; temporal key poses, based on the skeleton kinetic energy [6]; a dynamic Bayesian mixture model for classification [7]; spatio-temporal evolution of 3D postures [8]; self-organizing growing when required networks to learn spatio-temporal dependencies [9]; key poses association, using clustering algorithms without the need of a learning algorithm [10]; multi-layer codebooks of key poses and atomic motions, representing patterns of a certain human activity [11]. Random forests have also been used to classify human activities [12], as well as human gestures [13].

The two approaches presented in this paper, extract a set of features from 3D skeleton joints position data and use a random forest algorithm for classification. The first approach uses a fixed-size window to extract a set of features that describe each human activity, while the second one uses a variable size window, delimited by a pair of automatically detected key poses. The rationale for using the key poses is that they depict extreme points in the motion path of each joint, where most of the discriminative properties of each action are encoded. An overview of the proposed approach is shown in

Urbano Miguel Nunes and Paulo Peixoto are with Institute of Systems and Robotics, Department of Electrical and Computer Engineering, University of Coimbra, Polo II, 3030-290 Coimbra, Portugal (emails: urbanomiguel.g.nunes@ieee.org, peixoto@isr.uc.pt).

Diego Faria is with the System Analytics Research Institute, Aston University, Birmingham, UK (email: d.faria@aston.ac.uk).

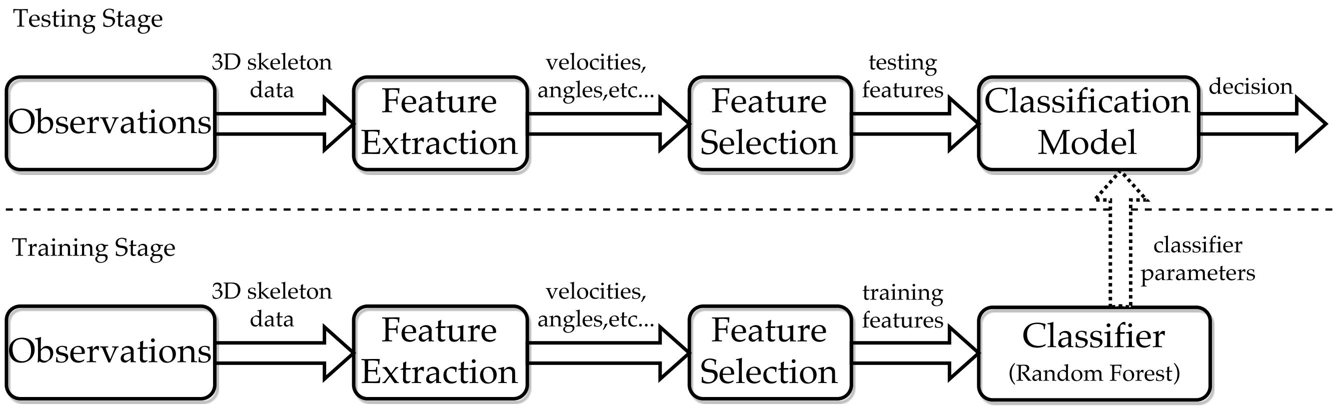


Fig. 1. Overview of the proposed approach. First, a human activity scene is observed and 3D skeleton data is collected. Then, the considered features are extracted (e.g. velocities, distances between joints). The "Feature Selection" block is generic for both developed approaches, where max-min features and static and max-min dynamic features (approach I and II, respectively) are obtained. In the training stage, a classifier model is built using a random forest as the classification algorithm, where each decision tree is constructed with the CART algorithm. The obtained classification model is then used during the testing stage.

Fig. 1. Both approaches aim to be fast to train, requiring few training examples and low computational cost, with relevant accuracy and precision, compared to other *state of the art* methods.

III. Proposed Approach

Let p_{jd}^t be the value of the coordinate $d \in \{x, y, z\}$ of the joint $j \in \{1, ..., m\}$ (m is the number of skeleton joints), at frame t. Therefore, $P_j^t = \left(p_{jx}^t, p_{jy}^t, p_{jz}^t\right)$ is the 3D vector that contains the information about the position of the joint j at frame t. The coordinates system (x, y, z) is defined as follows, relatively to the camera (see Fig. 2): x corresponds to the width; y corresponds to the height; z corresponds to the depth.

A. Preprocessing of 3D Skeleton Data

Before the process of feature extraction is performed, a preprocessing step is applied to the raw 3D skeleton data in order, not only to attenuate noise introduced by the sensor, but also to normalize the data to accommodate for different user's height, limb length, orientation and position. This preprocessing stage consists on the following steps:

1) **Translation**, to define the same origin of coordinates system for all frames; the selected one corresponds to the *torso* of the human skeleton;
2) **Normalization**, to reduce the influence of different user's height and limb length; first, the height of the subject is calculated; then all skeleton 3D coordinates are normalized according to the calculated value;
3) **Rotation**, to guarantee that the activity is always observed from the same point of view, independently of the initial pose of the subject with regard to the sensor; the rotation of the skeleton is performed in the y axis, considering the plane formed by the *torso*, *right* and *left hip* in relation to a fronto-parallel plane to the sensor;
4) **Symmetrization**, to disambiguate between right and left-handed people; since the skeleton is already in the same fronto-parallel pose in relation to the camera, it is just necessary to consider a new sample based on a mirrored version of the original 3D skeleton data.

B. Spatio-Temporal Features

The considered features may be divided into two categories: static (e.g. geometrical) and dynamic (e.g. temporal) features. The static features are intended to give information about *key poses*, which are obtained in frames where the pose has a kinetic energy equal to zero [6]. These poses represent extremal positions of a skeleton, which may be used to segment and recognize activities. This information is explored in subsequent sections. Additionally, dynamic features are intended to describe skeleton movements between key poses.

1) Static Features:

- **Projected distances** between two joints

$$\delta_{ab}^t = \sqrt{\sum_d \left(p_{ad}^t - p_{bd}^t\right)^2}, \tag{1}$$

where d belongs to one of the following sets $\{x, y\}$, $\{y, z\}$ and $\{z, x\}$ for each projection considered;

- **Projected angles** based on three joints

$$\theta_{id_p}^t = \arccos\left(\frac{(\delta_{ab}^t)^2 + (\delta_{bc}^t)^2 - (\delta_{ac}^t)^2}{2 \cdot \delta_{ab}^t \cdot \delta_{bc}^t}\right), \tag{2}$$

where δ corresponds to the Euclidean distance between two joints, given by (1), where $d_p \in \{xy, yz, zx\}$ for each projection considered;

- **Normal vector to triangles** formed by three joints

$$\Delta_k^t = \frac{(P_a^t - P_b^t) \times (P_a^t - P_c^t)}{||(P_a^t - P_b^t) \times (P_a^t - P_c^t)||}; \tag{3}$$

- **Sum of log-cov energy entropy** based on the global skeleton joints positions

$$(l_{\text{cov}p})^a = \sum_i \text{U}_i \left\{\log\left(\text{cov} A_p^a\right)\right\}^2 \tag{4}$$

and based on the considered angles

$$(l_{\text{cov}\theta})^a = \sum_i \text{U}_i \left\{\log\left(\text{cov}A_\theta^a\right)\right\}^2, \tag{5}$$

where A_p^a and A_θ^a are matrices containing the values of skeleton joints positions and angles, respectively, associated to an activity a; cov represents the covariance matrix; log is the matrix logarithm and $\text{U}(\cdot)$ returns the upper triangular matrix elements. The idea of using the log-covariance is based on the work of Guo [14] and on its application to human activity recognition, in a way similar to the approach followed by Faria *et al.* [7].

2) Dynamic Features:

- **Velocities of joints coordinates**

$$v_{jd}^t = \left(p_{jd}^t - p_{jd}^{t-1}\right) \cdot f_r, \tag{6}$$

where f_r is the frame rate;

- **Projected angular velocities**

$$\omega_{id_p}^t = \left(\theta_{id_p}^t - \theta_{id_p}^{t-1}\right) \cdot f_r. \tag{7}$$

C. Feature Normalization

The features described above are combined along time to form the set of features matrix F', where each row corresponds to a feature vector and each column represents the variation of each feature along time. Next, the set of training and testing features matrices (F'_{tr} and F'_{te}, respectively) are normalized accordingly to:

$$f_{ij} = \frac{f'_{ij} - \min(F'_{\text{tr}\cdot j})}{\max(F'_{\text{tr}\cdot j}) - \min(F'_{\text{tr}\cdot j})}, \tag{8}$$

where f'_{ij} is the current value being normalized, f_{ij} is its respective value normalized and $F'_{\text{tr}\cdot j}$ refers to the column j of the matrix F'_{tr}. Two sets of normalized features are obtained: F_{tr} and F_{te} (training and testing sets, respectively). From this point on, these sets are generically referred as F.

D. Approach I - Max-Min Skeleton-based Features with Fixed-Size Window

Given a fixed-size window, used to observe a certain activity, the objective of this approach is to extract the maximum and minimum local values of considered features. The main reasons behind this approach are its low computational cost, since just maximum and minimum local values are needed to be computed, as well as the assumption that an activity can be discriminated just by considering the extreme movements (given by dynamic features) or poses (given by static features), since many activities are composed by repetitive sequences. The loss of temporal information is assumed as a possible limitation of this method.

Assuming that the examples contained in matrix F correspond to several activities, this matrix can be rewritten as:

$$F = \begin{bmatrix} F^1 & F^2 & \dots & F^a & \dots \end{bmatrix}^T, \tag{9}$$

where F^a is a sub-matrix that describes each activity a. Each of these sub-matrices are then sub-sampled into activity examples of n_{f_r} fixed-size number of frames (i.e. the window size)

$$F^a = \begin{bmatrix} F_1^a & F_2^a & \dots & F_n^a \end{bmatrix}^T. \tag{10}$$

In this first approach, only the following features are considered, based on experimental tests:

$$F_n^a = \begin{bmatrix} v_{jd}^t & \theta_{id_p}^t & \delta_{ab}^t \end{bmatrix}. \tag{11}$$

The size of each activity example matrix is $n_{f_r} \times 3((m - m_{\text{extd}}) + n_\theta + n_\delta)$, where m_{extd} is the number of joints not considered on the feature vector (e.g. *torso*) and n_θ and n_δ are the number of considered angles and distances between joints, respectively. From each activity example matrix F_n^a, a feature vector is constructed by computing the $n_{\max}$ maximum and $n_{\min}$ minimum values for each considered feature:

$$f_n^a = \begin{bmatrix} f_n^{\max} & f_n^{\min} & (l_{\text{cov}p})_n^a & (l_{\text{cov}\theta})_n^a \end{bmatrix}, \tag{12}$$

where $f_n^{\max} = \begin{bmatrix} v_{jd}^{\max} & \theta_{id_p}^{\max} & \delta_{ab}^{\max} \end{bmatrix}$ and $f_n^{\min} = \begin{bmatrix} v_{jd}^{\min} & \theta_{id_p}^{\min} & \delta_{ab}^{\min} \end{bmatrix}$. The length of this vector, with the additional features $l_{\text{cov}p}$ and $l_{\text{cov}\theta}$ associated to the example activity, which improved the overall results, is $[(n_{\max} + n_{\min}) \cdot (3((m - m_{\text{extd}}) + n_\theta + n_\delta)) + 2]$.

E. Approach II - Max-Min Skeleton-based Features and Key Poses

The first approach may be limited by the fact that the possible optimal fixed-size window may vary, depending on the activity. On the other hand, a certain activity may take different times to be executed in real-time. Therefore, it may not be possible to fix a window size to implement the first approach. The second approach aims to solve this issue, considering variable-size windows. The concept of *key poses*, based on the pose kinetic energy, was introduced in [6]. The key poses represent extreme points in the motion path of each joint, where most of the discriminative properties of each action are encoded, so they can be used to determine the size of the analysis window. In other words, instead of a fixed-size window, a window is determined by considering two consecutive key poses. A key pose is characterized by having a kinetic energy equal to zero. From [6], the pose kinetic energy is defined as

$$E^t = \frac{1}{2} \sum_{j=1}^{m} \sum_{d} \left(v_{jd}^t\right)^2, \tag{13}$$

where $d \in \{x, y, z\}$ and the key poses must satisfy

$$E^t < E_{\min}, \tag{14}$$

where $E_{\min}$ is a tuned threshold. Although the noise of the skeleton data was attenuated in the preprocessing stage, it is important to set an upper threshold E_{u} (i.e. hysteresis behavior) after a key pose is identified, so that another key pose may be determined, disregarding the possible noisy kinetic energy values in the neighborhood of the first. This guarantee that only a key pose is identified in a neighborhood, depending on the fact of the mentioned threshold is passed. As in the previous approach, based on experimental tests, matrices F_n^a describing activity examples are obtained, but this time

$$F_n^a = \begin{bmatrix} v_{jd}^t & \omega_{id_p}^t & \theta_{id_p}^t & \Delta_k^t \end{bmatrix}, \tag{15}$$

with size of $n_a \times [3((m - m_{extd}) + n_\theta + n_\Delta) + n_\omega]$, where n_a is the size of the respective window, n_Δ is the considered number of normals to triangles formed by three joints and n_ω is the considered number of projected angular velocities.

At this point, it is important to notice the distinction made previously about the calculated features: static and dynamic. Since key poses are determined on frames where the velocities of the corresponding joints are close to zero, it does not make sense to consider dynamic features there. Therefore, for the key poses only static features are extracted (i.e. θ and Δ). The max-min dynamic features (i.e. v and ω) are extracted, in the same way as explained in the first approach, but this time in the dynamic window defined by the key poses. In other words, from a given activity, information about static postures and dynamic movements is extracted, in between key poses, which delimit a variable sized window of an activity. Thus, an example vector of the form

$$f_n^a = \begin{bmatrix} f_{\text{static}}^1 & f_{\text{dynamic}} & f_{\text{static}}^{n_a} \end{bmatrix} \tag{16}$$

is obtained, where $f_{\text{static}}^t = \begin{bmatrix} \theta_{id_p}^t & \Delta_k^t \end{bmatrix}$ and $f_{\text{dynamic}} = \begin{bmatrix} v_{jd}^{\max} & \omega_{id_p}^{\max} & v_{jd}^{\min} & \omega_{id_p}^{\min} \end{bmatrix}$. The length of this vector is $[2 \cdot 3(n_\theta + n_\Delta) + (n_{\max} + n_{\min}) \cdot (3(m - m_{extd}) + n_\omega)]$.

F. Training Model - Random Forest

Breiman [15] first introduced the random decision forest (RF), which may be viewed as an ensemble of decision-tree classifiers. It consists of two phases, where each tree is grown to the largest extent possible, without pruning, or until some defined maximum depth is reached:

1) **Bootstrap Phase**: randomly select a subset of features, from the training set, which will be used for growing a tree; the remaining features form the out-of-bag (OOB) set, which is used to estimate the OOB-error of the training set;
2) **Growing Phase**: using classification and regression tree (CART) [16], for each node to be divided, select one feature, from the randomly selected subset; the parameters of each node of every tree are optimized.

This process is done until a defined number of maximum trees. In the experimental results, a maximum of 100 trees was used. The number of selected features used to form the random subset is given by $\text{int}(\log_2(n_{\text{features}}) + 1)$.

IV. Experimental Results

In order to assess the proposed method, the Cornell Activity Dataset (CAD-60) [1] was used. The implementation of both developed approaches was done in Matlab and the Weka Version 3-6-13 software [17] provided the RF training algorithm, in a 2.60 GHz Intel Core i5 CPU machine.

A. Cornell Activity Dataset

The CAD-60 consists of 3D skeleton's coordinates joints, acquired by a RGB-D sensor at a frame rate of 30 Hz. Figure 2 exemplifies the skeleton's data provided, as well as the assumed coordinates system. Table I shows the index considered for each joint. The dataset contains 12 human distinct activities plus 1 random action and 1 still posture, categorized into 5 environments (bathroom, bedroom, kitchen, living room and office), performed by 4 different subjects. The considered joint angles are defined in Table II and the normal to the triangles formed by groups of three joints are described in Table III. The considered distances between joints are presented in table IV. These considered features aim to provide a good discrimination between activities. In this sense, for example, distances between adjacent joints are not considered (e.g. δ_{12}), since they are the same, independently of the activity.

The performance indicators in terms of Precision (Prec) and

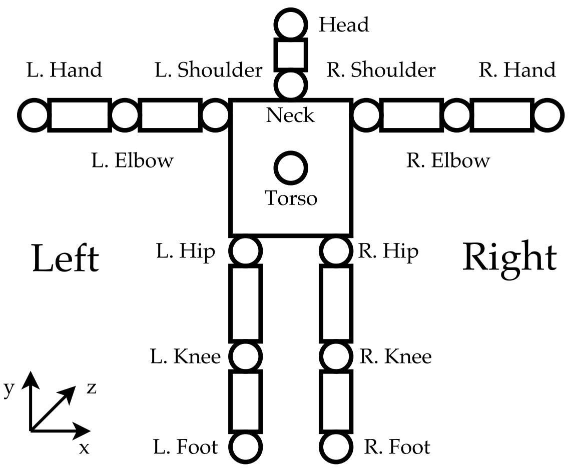


Fig. 2. Provided dataset: 3D skeleton data and respective coordinates system (after OpenNi [18]).

TABLE I
CAD-60 Dataset Joints

j	Joint	j	Joint	j	Joint
1	Head	2	Neck	3	Torso
4	L. Shoulder	5	L. Elbow	6	R. Shoulder
7	R. Elbow	8	L. Hip	9	L. Knee
10	R. Hip	11	R. Knee	12	L. Hand
13	R. Hand	14	L. Foot	15	R. Foot

TABLE II
Considered Angles

Angle	Joints Triplet	Angle	Joints Triplet
i	(a,b,c)	i	(a,b,c)
1	(6,7,13)	2	(4,5,12)
3	(6,10,11)	4	(4,8,9)
5	(10,11,15)	6	(8,9,14)
7	(6,10,13)	8	(4,8,12)
9	(1,7,13)	10	(1,5,12)
11	(3,12,13)	12	(3,14,15)

TABLE III
Considered Normal to Triangles Formed by Three Joints

Normal	Joints Triplet	Normal	Joints Triplet
k	(a,b,c)	k	(a,b,c)
1	(4,5,12)	2	(6,7,13)
3	(8,9,14)	4	(10,11,15)
5	(1,12,13)	6	(3,12,13)
7	(5,8,12)	8	(7,10,13)

TABLE IV
CONSIDERED DISTANCES BETWEEN JOINTS

Joints (a, b)	Joints (a, b)	Joints (a, b)	Joints (a, b)	Joints (a, b)
(1,4)	(1,5)	(1,6)	(1,7)	(1,8)
(1,9)	(1,10)	(1,11)	(1,12)	(1,13)
(1,14)	(1,15)	(2,5)	(2,7)	(2,8)
(2,9)	(2,10)	(2,11)	(2,12)	(2,13)
(2,14)	(2,15)	(3,5)	(3,7)	(3,9)
(3,11)	(3,12)	(3,13)	(3,14)	(3,15)
(4,7)	(4,9)	(4,11)	(4,12)	(4,13)
(4,14)	(4,15)	(5,6)	(5,8)	(5,9)
(5,10)	(5,11)	(5,13)	(5,14)	(5,15)
(6,9)	(6,11)	(6,12)	(6,13)	(6,14)
(6,15)	(7,8)	(7,9)	(7,10)	(7,11)
(7,12)	(7,14)	(7,15)	(8,12)	(8,13)
(9,12)	(9,13)	(10,12)	(10,13)	(11,12)
(11,13)	(12,14)	(12,15)	(13,14)	(13,15)

Recall (Rec) are presented, for each scenario [19]. A leave-one-out cross validation procedure is employed: the model is trained by three of the four subjects and tested using the remaining one. This strategy enables the conclusion about the generalization capability of the classifier using the proposed set of features.

B. Results and Analysis - Approach I

The results obtained for the first approach are presented in Table V. The overall average for precision was 87.62% and for recall was 84.82%. The following parameters were used:

- $n_{f_r} = 120$; this means that, if an activity has more frames, it is divided roughly into examples of size n_{f_r};
- $m_{extd} = 3$; the velocities of the *head*, *neck* and *torso* were not used;
- $n_\theta = 12$; the considered angles are shown in Table II;
- $n_\delta = 70$; the considered distances between joints are presented in Table IV;
- $n_{\max} = n_{\min} = 1$; only the most extreme values for each feature on the analysis window per activity were considered.

These results show that with few examples to train the classifier (the average number of examples to train the RF classifier is 386,7), allied to just max-min skeleton-based features (each example has 566 features), it is possible to discriminate between distinct human activities, with a good confidence and very fast training (the average training time was 611 ms). Based on these characteristics, this approach could be suitable for real-time applications.

C. Results and Analysis - Approach II

The results obtained for the second approach are presented in Table VI. The overall average for precision was 81.73% and for recall was 79.01%. The following parameters were used:

- $m_{extd} = 3$; the velocities of the *head*, *neck* and *torso* were not used;
- $n_\theta = 12$; the considered angles are summarized in table II;
- $n_\omega = 12$; the considered angular velocities are obtained based on their respective angles;
- $n_\Delta = 8$; the considered normal to triangles formed by three joints are presented in table III;
- $n_{\max} = n_{\min} = 1$; only the most extreme values for each feature on the dynamic window per activity were considered;
- $E_{\min} = 0.0028$; this value was tuned empirically, based on experimental tests on the training data;
- $E^a_u = 2 \times \text{mean}\left(E^a\right)$, where mean$(\cdot)$ is the mean function and $E^a = \left\{\left(E^1\right)^a, \left(E^2\right)^a, ..., \left(E^t\right)^a, ...\right\}$ is the set of kinetic energy values of the activity a; this value was also tuned empirically, based on experimental tests on the training data.

It is important to notice some practical constrains, which may contribute to reduce the overall performance of the approach:

- It calculates the global skeleton's kinetic energy, based on the velocities of **all** joints; this means that for a key pose to occur, every joints must have zero velocity; lets consider the *drinking water* activity (which revealed the worsts results): only the upper joints are of interest in this activity; however secondary motions can interfere in the computation of the actual key poses (e.g. noisy data; leg movements);
- The $E_{\min}$ value, which was obtained based on tests for all activities, should not be a **fixed threshold**, since there are activities with different motion patterns; this means that more examples were obtained for less dynamic activities (e.g. *relaxing on couch*), compared to more dynamic ones (e.g. *brushing teeth*).

TABLE V
PERFORMANCE OF THE APPROACH I ON THE CAD-60

Location	Activity	Prec (%)	Rec (%)
Bathroom	random+still	86.55	92.12
	rinsing water	91.67	80.05
	brushing teeth	98.92	95.25
	wearing lens	93.75	83.02
	average	92.72	87.61
Bedroom	random+still	93.20	99.00
	talking on phone	88.00	77.12
	drinking water	77.50	84.47
	opening container	71.67	60.95
	average	82.59	80.38
Kitchen	random+still	91.47	96.75
	drinking water	99.07	100
	chopping	91.97	96.65
	stirring	99.07	92.22
	opening container	72.05	65.65
	average	90.73	90.25
Living room	random+still	95.75	98.87
	talking on phone	73.40	55.75
	drinking water	75.80	75.65
	talking on couch	89.00	94.22
	relaxing on couch	70.00	71.87
	average	80.79	79.27
Office	random+still	93.25	97.75
	talking on phone	79.40	66.95
	writing on board	96.50	94.90
	drinking water	87.27	73.40
	working on computer	100	100
	average	91.28	86.6
	Overall Average	**87.62**	**84.82**

TABLE VI
PERFORMANCE OF THE APPROACH II ON THE CAD-60

Location	Activity	Prec (%)	Rec (%)
Bathroom	random+still	95.87	96.60
	rinsing water	81.57	68.32
	brushing teeth	96.05	92.97
	wearing lens	84.05	85.35
	average	89.38	85.81
Bedroom	random+still	97.87	99.62
	talking on phone	56.05	66.50
	drinking water	57.15	36.85
	opening container	100	94.35
	average	77.77	74.33
Kitchen	random+still	93.00	98.47
	drinking water	99.00	95.82
	chopping	83.77	92.50
	stirring	73.07	64.80
	opening container	100	86.32
	average	89.77	87.58
Living room	random+still	96.70	99.62
	talking on phone	59.82	75.07
	drinking water	58.15	33.42
	talking on couch	81.25	85.72
	relaxing on couch	75.00	62.50
	average	74.18	71.27
Office	random+still	94.60	96.90
	talking on phone	49.72	71.02
	writing on board	92.17	90.87
	drinking water	51.32	21.55
	working on computer	100	100
	average	77.56	76.07
	Overall Average	**81.73**	**79.01**

Nevertheless, these problems inspire some new ideas for future research, which are discussed in the next section. The training of the classifier using this approach is also fast (the average training time is 0.92 s), requiring an average of 630.30 training examples, each with 264 features. Given these characteristics, this approach is also suitable for being used on real-time applications.

V. CONCLUSION AND FUTURE WORK

The proposed approaches are based in max-min skeleton-based features. While in the first approach each example consists of a fixed-size window, in the second one, the considered window is delimited by key poses (no fixed-size window), determined by frames where the skeleton has zero kinetic energy. The second approach may suggest several new research directions:

- Instead of considering the global skeleton's kinetic energy, it may be possible to divide the human skeleton into several parts, calculate the kinetic energy for each part and then apply a similar method as approach II to those parts; this could be useful to differentiate parts that are more/less important to some activity;
- Train a specialized RF for each part, according to their correspondent features, having a higher-level classifier to discriminate between activities; it would also be interesting to assign different weights to each part, which could enable the detection of multiple activities at the same time (i.e. activities being performed by different body parts);
- Distinguish transitions between different activities, considering the computed key poses;
- Develop a parallel system to perceive the context of the activity (e.g. classification of used objects), allowing the extraction of context features.

Another interesting characteristic of the proposed method is its fast training, requiring few training examples. This could lead to a line of research, where the training algorithm could learn and incorporate new activities in real-time, conciliating all the previously mentioned directions of research.

REFERENCES

[1] J. Sung, C. Ponce, B. Selman, and A. Saxena, "Human Activity Detection from RGBD Images," *Plan, Activity, and Intent Recognition*, vol. 64, 2011.

[2] R. Gupta, A. Y.-S. Chia, and D. Rajan, "Human Activities Recognition using Depth Images," in *Proceedings of the 21st ACM international conference on Multimedia*. ACM, 2013, pp. 283–292.

[3] J. Shotton, T. Sharp, A. Kipman, A. Fitzgibbon, M. Finocchio, A. Blake, M. Cook, and R. Moore, "Real-time Human Pose Recognition in Parts from Single Depth Images," *Communications of the ACM*, vol. 56, no. 1, pp. 116–124, 2013.

[4] J. Wang, Z. Liu, Y. Wu, and J. Yuan, "Learning Actionlet Ensemble for 3D Human Action Recognition," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 36, no. 5, pp. 914–927, 2014.

[5] X. Yang and Y. Tian, "Effective 3D Action Recognition using Eigen-joints," *Journal of Visual Communication and Image Representation*, vol. 25, no. 1, pp. 2–11, 2014.

[6] J. Shan and S. Akella, "3D Human Action Segmentation and Recognition using Pose Kinetic Energy," in *Advanced Robotics and its Social Impacts (ARSO), 2014 IEEE Workshop on*. IEEE, 2014, pp. 69–75.

[7] D. R. Faria, M. Vieira, C. Premebida, and U. Nunes, "Probabilistic Human Daily Activity Recognition Towards Robot-Assisted Living," in *Robot and Human Interactive Communication (RO-MAN), 2015 24th IEEE International Symposium on*. IEEE, 2015, pp. 582–587.

[8] S. Gaglio, G. L. Re, and M. Morana, "Human Activity Recognition Process using 3-D Posture Data," *Human-Machine Systems, IEEE Transactions on*, vol. 45, no. 5, pp. 586–597, 2015.

[9] G. I. Parisi, C. Weber, and S. Wermter, "Self-organizing Neural Integration of Pose-motion Features for Human Action Recognition," *Frontiers in Neurorobotics*, vol. 9, 2015.

[10] E. Cippitelli, S. Gasparrini, E. Gambi, and S. Spinsante, "A Human Activity Recognition System using Skeleton Data from RGBD Sensors," *Computational Intelligence and Neuroscience*, vol. 2016, 2016.

[11] G. Zhu, L. Zhang, P. Shen, and J. Song, "Human Action Recognition using Multi-layer Codebooks of Key Poses and Atomic Motions," *Signal Processing: Image Communication*, 2016.

[12] L. Gan and F. Chen, "Human Action Recognition using APJ3D and Random Forests," *Journal of Software*, vol. 8, no. 9, pp. 2238–2245, 2013.

[13] W. Liu, Y. Fan, T. Lei, and Z. Zhang, "Human Gesture Recognition using Orientation Segmentation Feature on Random Forest," in *Signal and Information Processing (ChinaSIP), 2014 IEEE China Summit & International Conference on*. IEEE, 2014, pp. 480–484.

[14] K. Guo, *Action Recognition using Log-Covariance Matrices of Silhouette and Optical-Flow Features*. Boston University, 2012.

[15] L. Breiman, "Random Forests," *Machine Learning*, vol. 45, no. 1, pp. 5–32, 2001.

[16] L. Breiman, J. Friedman, C. J. Stone, and R. A. Olshen, *Classification and Regression Trees*. CRC press, 1984.

[17] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten, "The WEKA Data Mining Software: an Update," *ACM SIGKDD Explorations Newsletter*, vol. 11, no. 1, pp. 10–18, 2009.

[18] "OpenNi SDK," Accessed: 2016-06-06. [Online]. Available: http://www.openni.org/

[19] J. Sung, C. Ponce, B. Selman, and A. Saxena, "Unstructured Human Activity Detection from RGBD Images," in *Robotics and Automation (ICRA), 2012 IEEE International Conference on*. IEEE, 2012, pp. 842–849.