# A Comparative Analysis of Classification Algorithms Applied to M5AIE-Extracted Human Poses

André Brandão, Leandro A. F. Fernandes, and Esteban Clua
MediaLab-UFF, Instituto de Computação, Universidade Federal Fluminense
CEP 24210-240 Niterói, RJ, Brazil
Email: brandao@daad-alumni.de, {laffernandes,esteban}@ic.uff.br

***Abstract*—Classification algorithms became an important subject in games, mainly because of the introduction of new interaction paradigms such as the Natural User Interfaces (NUI). It is possible to find many works that make use of classification algorithms in the literature, but they still do not provide any study that could compare different classification algorithms in the context of human pose classification. Most of the related work mention the adopted algorithm for classification issues. Nevertheless, we do not know the answer of the question: Which classification algorithm could give the best choice in pose recognition context? In this paper we propose and develop a detailed analysis, using our own pose detection and tracking method, called M5AIE method with different algorithms: C5.5 Gain Ratio Decision Tree, Naïve Bayes Classifier and K-Nearest Neighbor (KNN) Algorithm. As a consequence of our study, we provide results that can help researchers to choose among the selected algorithms to use in human pose classification in digital games context.**

## I. Introduction

In the past few years, important advances were achieved in Computer Vision research, specially in gesture recognition. These advances create the possibility of application development in Human-Computer Interaction, security, health-care and digital games [1]. According to Tang et al. [2], there is an increasing demand from game players for games enhanced functionalities and one of these functionalities is interaction.

The launch of low-cost capture devices of depth images promoted facilities in gesture recognition research. These devices provide better options for the development of interaction paradigms such as Natural User Interface (NUI). In 2010, Microsoft launched the Kinect and described how it works in [1]. Shotton et al. presented how the device works and the applicability in digital games. The authors selected the possible user's playing movements (driving, kicking, running, navigating menus, etc) that are applied in Xbox 360 games. For experiments, the authors used 31 keypoints and their classifier was developed using tree structures. In another work, Shotton et al. [3] describe new approaches to human pose estimation and also use tree structures.

Tang et al. [2] describe a study that uses an interactive dancing game with real-time recognition of continuous dance moves from 3D human motion capture. The authors describe positive feedback from the users experiences. Tang et al. also describe the development of their own motion recognition algorithm for dance moves.

Rogez et al. [4] address human pose recognition as a classification problem. In their work, the authors describe a pose detection algorithm that is based on tree structures. Shotton et al. [1], [3], Tang et al. [2], Rogez et al. [4] and many other related studies mention the used classification algorithm. However, the authors do not explain why they chose each of their options, and they do not compare their algorithms with traditional classification algorithms.

To the best of our knowledge, there is no work in the literature that performs a comparison among classification algorithms in human pose recognition in the context of games. Huang et al. [5] compared Naïve Bayes, Decision Trees, and Support Vector Machines (SVMs), to evaluate which is the best measure to use when classification algorithms are compared. In [5], the authors used datasets that had only two classes (binary datasets) and compare the use of two different measures: accuracy and Area Under the Curve (AUC). In this thesis, the accuracy measure was used in the experiments (Section V) to evaluate the selected classification algorithms. Amor et al. [6] used intrusion detection system datasets to compare Naïve Bayes and Decision Trees. Amor et al. described how a Naïve Bayes classification algorithm can provide competitive results. The authors of the two studies [5], [6] did not consider datasets in human pose classification in their experiments.

In this paper we propose and develop a detailed analysis, using our own pose detection and tracking method, called M5AIE method, with different algorithms: C5.5 Gain Ratio Decision Tree [7], Naïve Bayes Classifier [8] and K-Nearest Neighbor (KNN) Classifier [9]. We provide results that can help researchers to choose among the selected algorithms to use in human pose classification in digital games context.

The main contributions of this paper include the following:

- How different classification algorithms can be used in human pose detection in digital games context; and
- A comparative analysis of three classification algorithms in human pose classification.

This paper is organized as follows: Section II presents some of the related studies. Section III describes the M5AIE method which is the body part labeling and tracking method. Section IV describes the selected classification algorithms for this study. The experiments and results are presented in Section V. Section VI concludes the work with a discussion and future directions for the research.

## II. Related Work

Human action recognition is a related area of Computer Vision that addresses motion in videos. Mota et al. [10]

introduced a video motion indexing scheme that was based on modeling optical flow. In their work, the authors proposed a global motion tensor descriptor for video sequences, and optical flow was described with a polynomial representation. In contrast to Mota et al.'s work, this work is concerned with the detection and tracking of body parts in RGB-D image sequences and with pose identification in single frames of the sequence.

Movement recognition in games has many different approaches. Many of the movement recognition technologies apply video or frames in real time as an input of a recognition system. Raskar et al.'s work [11] presents one of the approaches which use photosensitive tags to decode optical signals can capture locations of each point, orientation, incident illumination and reflectance. The photosensitive tags are markers which are imperceptible. However, the usage of tags can complicate their application in digital game context, because they are very specific. Also, Raskar et al.'s work detect movements but it does not classify human poses. A low-cost approach is described by Wang and Popović [12]; these authors propose an easy-to-use color glove. In that case, even the color gloves are inexpensive; it was necessary to use instrumented gloves in the construction of the database. The database is used to recognize letters in sign language. Nevertheless, Wang and Popović's work performs pose virtual reconstruction, which is unnecessary in the context of this work. Wang and Popović applied their work on sign language finger spelling. Figueiredo et al. [13] also make use of gloves, and their study made use of a yellow gloves, which were applied to interaction only, without pose or movement recognition. Almeida [14], as Wang and Popović, worked with sign language and they made use of a depth sensor to capture the movements. Wang and Popović and Almeida use classifiers but they did not detail the used classifiers. Figueiredo et al.'s work do not use classifier because their focus is on interaction.

Bourdev and Malik [15] and Bourdev et al. [16] apply a two dimensional image as input and estimates the three dimension coordinates of the selected keypoints. Moreover, it is presented the definition of Poselets as a particular part of the human pose under a given viewpoint. This approach is defined with a set of examples that are close in 3D configuration space. The main contribution of [15], [16] is the notion of a part of a pose, a "poselet", and an algorithm for selecting good poselets. Each poselet provides examples for training a linear SVM classifier. However, none of these studies detailed why they chose the selected classifier.

Bleiweiss et al. [17] describe a real-time framework that blends the players actual movements, which are tracked using a depth sensor, with pre-defined animation sequences. According to the authors, their depth-based framework enables an enhanced visual feedback mechanism by understanding the player's full body motion and seamlessly blending it with pre-animated content. Bleiweiss et al. made use of a full body tracking algorithm proposed by Ganapathi et al. [18]. In [17], the authors presented a system whose skeleton mimics the player's movements.

The work of Schönauer et al. [19] describes a full body input for rehabilitation. The contribution of their work includes the implementation of serious game targeting rehabilitation of patients with chronic pain of the lower back and neck. Schönauer et al. argue that the tracker used for their motion capture system, which is an io-tracker [20], is marker based and uses an infrared optical motion tracking system. In [19], Schönauer et al. used a classification tree for posture estimation and did not compare the classifier with any other algorithm.

An interactive dancing game was presented by Tang et al. [2]. The motion recognition algorithm was developed based on a finite state machine representation and a Block Matching Cost. The game uses motion templates, and the Block Matching Cost calculates the cost for matching a frame of the player's move with a frame of a template move.

Tian et al. [21] presented a semantic feature to represent characteristics of different human motion classes. Cimen et al. [22] describe how they classify human motion with descriptors that are related to emotion classification. Sun et al. [23] show how conditional regression forests can be used in human pose estimation.

In our work, we chose the Kinect since it is a low-cost device and it provides real-time movement recognition. There are available technologies to integrate the Kinect with Unity3D engine, which was used in the development of the Jecripe game [24]. This game inspired the selected movements in our experiments. The next section presents some games and advantages of using movement recognition in digital games. The following section describes the M5AIE method for human body parts detection and tracking.

## III. The M5AIE Method

The M5AIE method aggregates different concepts; some of them were not originally developed for detecting, tracking, and pose classification. The computational flow of the M5AIE algorithm is illustrated in Fig. 1. After the alignment of the RGB and depth information of a given frame, we use the Minimum Background Subtraction algorithm [25] to address most of the unnecessary information in the frame (Section III-A). In turn, the area of the person facing the sensor is replaced by the few pixels that define its discrete medial axis transformation (Section III-B). The detection of body part candidates begins by building a graph in which each pixel of the medial axis is seen as a vertex that is connected to its neighbors by weighted edges; each weight is given by the Euclidean distance between a pair of pixels (Section III-C). Using this graph, body part candidates are detected through the AGEX points detection method (Section III-D). A labeling step is performed to relate AGEX points to their respective body parts (Section III-E). When labeling fails, the information computed in the previous frame is used in combination with the ASIFT method for tracking the body parts into the current frame (Section III-F). Pose classification is performed in the last stage of our method (Section IV).

### *A. Minimum Background Subtraction Algorithm*

The Minimum Background Subtraction algorithm is composed of training and subtraction stages. During the training stage, the approach limits the background values regarding the following assumptions: indoor environment, static background, and static position and orientation of the sensor. The algorithm is presented in [25]. We have chosen to use this background subtraction algorithm because it had the best results in previous experiments [26].

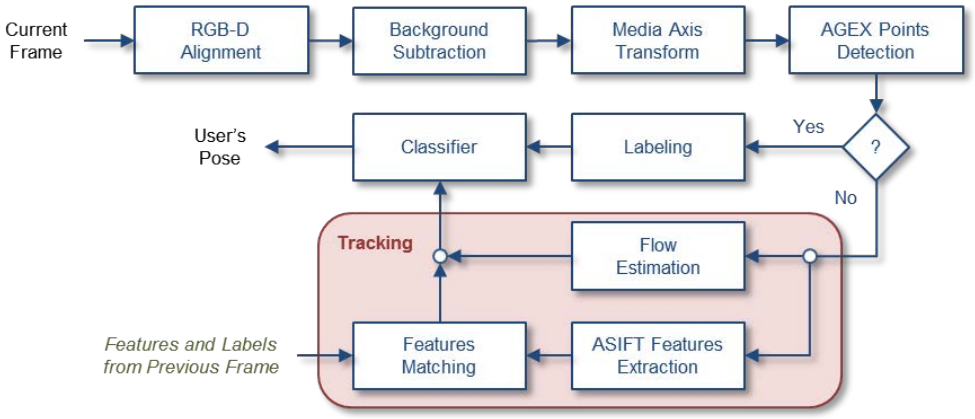



Fig. 1. Flowchart of the proposed M5AIE approach applied to RGB-D images to identify the pose of the imaged subject. See Section III for details. The question mark after the AGEX Points Detection stage verifies whether the Labeling stage of the algorithm can be performed.

### *B. Discrete Medial Axis Through Distance Transformation*

The 2D medial axis transform constitutes finding the centers of the maximum disks that can fit inside of an object [27]. We use such a structure to reduce the number of pixels that are to be considered as vertices in the graph computation (Section III-C). By doing so, we reduce the processing overload of the next stages of our method. We have performed medial axis extraction by computing the distance transform of the binary image that results from the background subtraction.

### *C. Graph Construction Based on Depth Image*

We use image pixel coordinates to build a graph in linear time, as implemented in Schwarz et al. [28]. In such a case, two vertices are considered to be neighbors if the corresponding pixels are separated by a maximum distance threshold $\delta$. We follow Plagemann et al.'s strategy [29] to connect two vertices and Schwarz et al.'s scheme to weight the edges with the Euclidean distance of the imaged surface points related to the vertices. However, we build the graph with medial axis pixels only, while previous works use the whole shape of the user. As a result, we get an optimized graph.

### *D. Accumulative Geodesic Extrema Points*

Accumulative Geodesic Extrema Points, known as AGEX points, are selected while considering the distances of the points according to the edges that connect the vertices in the graph $G_t$ [29]. This method maximizes the distances of the points using the Dijkstra [30] algorithm. A detailed description of AGEX is presented in [29].

### *E. Body Part Labeling*

The initialization step for body part labeling comprises a person facing the camera for a few seconds and taking a snapshot on a T-pose. Because the first AGEX point ($AGEX_1$) is the centroid, we can define the lower and upper parts of the body and separate the other points (from $AGEX_2$ to $AGEX_6$) according to their coordinate values.

### *F. ASIFT-Based Body Parts Tracking*

In our tracking strategy, ASIFT is used to identify the features in the frame $t$ that are related to the AGEX points identified in frame $t-1$. However, ASIFT (more details in [31]) cannot be used directly in tracking due to some practical issues: (i) the time execution increases as the input images become larger; (ii) in the case of background segmented images, ASIFT detects too many features in the border of the foreground region; (iii) there is not necessarily a matching feature for every pixel from one image to another; and (iv) ASIFT can match two features whose positions are far away from an expected conservative maximum distance. We addressed these problems using the following heuristics.

**Use of tiny images instead of complete frames:** to avoid the heavy computational load of ASIFT applied to the whole image, we apply ASIFT on five tiny images that contain the body parts in frame $t-1$ and the sub-images of the regions in which the same body parts can possibly be found in frame $t$.

**Blurring the background of sub-images:** the ASIFT method usually detects features only at the frontier between the foreground and the background regions because of the high contrast between fore- and background pixels. As a result, the blurred images improve the detection of ASIFT features inside the foreground region.

**Searching in a region instead of searching for coordinates only:** this heuristic is related to the problem that there is not necessarily a matching feature for every pixel from one image to another. We handle this problem in the following way: if there is no body part matching feature from the sub-image at $t-1$ with the sub-image at $t$, then we search for the point $P$, which is the nearest body feature in $t-1$ that has a match in $t$. Then, $P$ is considered the body part final result.

**Body-parts position estimation:** to assert the consistency of the matching of ASIFT features in the sub-images of consecutive frames, we estimate the expected location of the feature in frame $t$ using the uniform linear motion equation considering its location in frames $t-1$ and $t-2$.

In our framework, the acquisition of the color and the

depth images is performed by the same device (a Kinect), and the RGB-D image alignment is performed by Kinect SDK. However, due to the asynchronous nature of the image sensors, the final aligned RGB-D image may be ill-formed. As a result, background color pixels can be incorrectly mapped to foreground regions. To make the proposed matching procedure suitable for tracking, we found four major situations to be handled: (i) the matched ASIFT feature and the point estimated with the uniform linear motion equation correspond to well-mapped background pixels; (ii) the matched ASIFT feature resides in the well-mapped background while the estimated point is part of the users body; (iii) the matched ASIFT feature belongs to the human body, and the estimated point is part of the background; and (iv) both the matched ASIFT feature and the estimated point correspond to the actual body. Due to space restrictions, this paper does not include a discussion about how to handle each situations.

## IV. Pose Classification

Classification techniques were used in this study to identify categorical labels such as "Pose A" and "Pose B" for the current subject, according to the position of each of the body parts that are detected or tracked in a given image of the sequence.

The human pose classification was performed using three different algorithms: the C4.5 Gain Ratio Decision Tree [7], the Naïve Bayes classifier [8] and the K-Nearest Neighbor (KNN) classifier [9]. These algorithms were selected due to their low computational load and simplicity, which makes them suitable for real-time applications.

### A. C4.5 Gain Ratio Decision Tree

Decision trees follow the "divide and conquer" approach. According to Amor et al. [6], the decision tree structure is composed of the following elements: (i) decision node, which specifies a test attribute that is responsible for the comparison of an attribute value with a constant; (ii) an edge that is one of the possible attribute values (the test attribute is placed here); and (iii) leaf nodes that give the classification to which the object belongs.

Decision trees have two stages: building the tree and the classification itself. Building the tree constitutes selecting the test value for each decision node and the classification labels of each leaf. Decision trees are built based on a given training set. The classification stage is made starting from the root of the decision tree. To go down the tree, tests are made to achieve one of the leaf nodes.

Many algorithms were developed for the construction of decision trees for the classification task. In the experiments, the considered decision tree was the C4.5 Gain Ratio Decision Tree algorithm developed by Quinlan [7]. The C4.5 Gain Ratio Decision Tree selects the attribute that has the largest number of possible values to be assumed as the current node in the tree construction. This criterion is an extension to information gain. However, the Gain Ratio applies a normalization to information gain. Then, the selection of an attribute to be the current node in the tree is defined on the largest gain ratio value. The Gain Ratio value is obtained by the information gain and the normalization of the probability of each of the attribute values.

### B. Naïve Bayes Classifier

Naïve Bayes makes a strong independence relation in which the features are independent in the context of a session class [6], [8]. The Naïve Bayesian classifier works, basically, as follows: (i) the training set is composed of tuples, and these tuples are attribute values in a predefined order; (ii) for each class, a conditional probability can be calculated based on the used training set; and (iii) the likelihood of a testing tuple is defined based on the calculated conditional probabilities of each class.

The main difference between the two mentioned classifiers is that while C4.5 is a decision tree classifier, the Naïve Bayes is based on the Bayes rule of conditional probabilities. In decision trees, the attributes are tested, and the final classifications are at the leaves. In this approach, the attributes have a high level of dependency on each other. However, the Naïve Bayes classifier evaluates each attribute individually, considering them to be independent.

### C. K-Nearest Neighbor Classifier

In 1967, Cover and Hart [9] introduced the k-Nearest Neighbor as a pattern classifier. A training set is built by tuples and a tuple $X$, whose class is unknown, is then tested. The tuple $X$ is compared with each of the training tuples. The $k$ closest tuples to $X$ are considered to predict its class. "Closeness" is considered a distance metric, and it can be calculated, for example, with the Manhattan, Chebyshev or Euclidean distance. The unknown class of $X$ is assigned to the most common class among its $k$ nearest neighbors.

### D. Bounding Box and Grid

In this work, the algorithms receive as input the labels and the locations of the body parts according to an $N \times N$ grid that is defined inside the bounding box that contains the whole body of the imaged subject. Fig. 2 shows the grid squares with $N = 8$. A bounding box was used to identify the cell number of the body parts. The bounding box provides the relative positions according to the detected human body. This approach makes it possible to identify the cell number of the body parts, independently of their occupied positions in the whole segmented image.

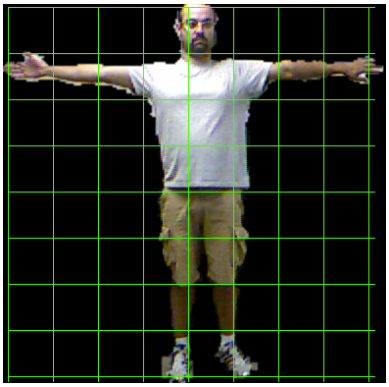


Fig. 2. A bounding box limits the human body and it is divided into $N \times N$ cells. In this example, $N = 8$.

The considered classification algorithms (C4.5, Naïve Bayes and KNN) require the execution of a training stage to build a model to be used during the classification of the poses (see Figure **??**). In our work, the dataset is used both for training and testing. The class of each tuple comprises the cell-coordinates in the grid that body parts assume at each image in a sequence. The pose classification of each training tuple was made manually in each frame. In the classification procedure, a tuple constitutes a sequence in which the cell position of every individual body part is described in the same order as the order that appears in the attributes definition. The following code illustrates an example of how the attributes are declared and a few training tuples:

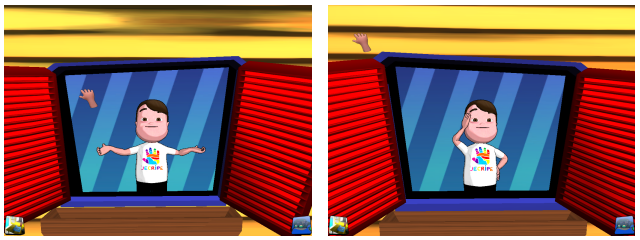
```
@relation poses

@attribute headx NUMERIC
@attribute heady NUMERIC
@attribute rightFootx NUMERIC
@attribute rightFooty NUMERIC
@attribute leftFootx NUMERIC
@attribute leftFooty NUMERIC
@attribute leftHandx NUMERIC
@attribute leftHandy NUMERIC
@attribute rightHandx NUMERIC
@attribute rightHandy NUMERIC
@attribute pose {tpose, dancing, guitar,
drum, kick, punch, kickAndPunch}

@data
0, 3, 7, 5, 7, 3, 1, 0, 1, 7, tpose
0, 4, 7, 5, 7, 3, 1, 0, 1, 7, tpose
0, 4, 7, 6, 7, 1, 0, 0, 3, 7, dancing
0, 3, 7, 5, 7, 2, 0, 0, 3, 7, dancing
0, 5, 7, 6, 7, 3, 1, 0, 3, 7, guitar
0, 3, 7, 5, 7, 3, 3, 0, 3, 7, drum
...
```

The first ten attributes are numeric. Each of these attributes is related to a coordinate of the considered body parts. The *pose* attribute is the only one which is categorical. The training data begins with the label *@data*. Each attribute is separated by a coma. The last attribute is the testing class.

## V. Experiments and Results

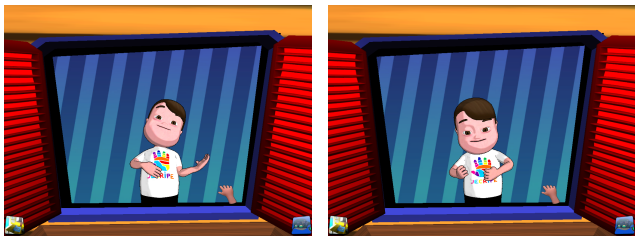
The described approach was implemented in Python and was evaluated on real image sequences. The ASIFT algorithm was implemented in C++. We used the reference implementation provided by Morel and Yu [32]. To perform the distance transformation, we used OpenCV adaptive thresholding and other basic image processing procedures. The image sequences were collected using a Kinect sensor, which provides both depth and color images with a $640 \times 480$ pixel resolution. The resolution of the tiny images was set to $80 \times 80$.

The classification algorithms were evaluated using the data mining tool WEKA 3.6.8 (Waikato Environment for Knowledge Analysis) [33]. To adopt the traditional classifiers C4.5 Gain Ratio Decision Tree, Naïve Bayes and KNN, we used the J48, Naïve Bayes and Ibk implementations that are available in the WEKA tool, respectively.



(a) *T-pose* (b) *Dancing*

Fig. 3. Illustration of *T-pose* and *dancing* human poses in a game developed by our research group that inspired our experiments.



(a) *Play guitar* (b) *Play drums*

Fig. 4. Illustration of *Play guitar* and *Play drums* human poses.

We used $k$-fold cross-validation in our test. In this approach, the dataset is randomly partitioned into $k$ subsets. Only one subset is used as validation data for testing the model. The other $k-1$ subsets are used for training the classification model. The cross-validation process is repeated $k$ times. Each of the $k$ subsets is used only once for validation. The final result is the average of the results obtained at each round. In our experiments, we used $k = 10$.

The goal of these experiments is to answer the following questions:

1) Can the classifiers make correct predictions with different poses and the same class?
2) Is there any difference in the results when using different users to build the data set?
3) Which is the best value for $N$ in the grid $N \times N$?
4) Which of the three considered classification algorithms is the best for human pose prediction in the game context?

We previously collected sequences with human poses that were inspired in the Jecripe game, developed by our research group [24]. The poses define the classes, which are: *T-pose*, *dancing*, *play guitar*, and *play drums* (see Fig. 3 and Fig. 4). Three other movements, which were not related to the game, were also included: *punch*, *kick* and *kick + punch*.

We characterize the classes as the following: The *T-pose* constitutes a person with both arms and hands at the same level as the shoulders. In the *dancing* class, one of the hands is on the head; the other hand is on the hip, and one or both feet are on the ground. As a consequence, we have six combinations of poses for the class *dancing*: (i) left hand on the head and feet on the ground; (ii) left hand on the head and moving left foot; (iii) left hand on the head and moving right foot; (iv) right hand on the head and feet on the ground; (v) right hand on the head and moving right foot; and (vi) right hand on the

TABLE I. IMAGE SEQUENCE EVALUATION FOR VOLUNTEER A.

| Sequence Number | Movement | Number of Images | Tracking Until The End |
|---|---|---|---|
| Sequence A1 | dancing (i) | 140 | yes |
| Sequence A2 | dancing (i) | 116 | yes |
| Sequence A3 | dancing (ii) | 100 | yes |
| Sequence A4 | playing guitar | 140 | yes* |
| Sequence A5 | playing drums | 190 | yes* |
| Sequence A6 | playing drums | 130 | yes* |
| Sequence A7 | playing drums | 130 | yes* |
| Sequence A8 | punch (I) | 84 | yes |
| Sequence A9 | punch (I) | 81 | yes** |
| Sequence A10 | kick (a) | 66 | yes |
| Sequence A11 | dancing (iii) | 58 | yes |
| Sequence A12 | dancing (ii) | 68 | yes |
| Sequence A13 | kick + punch (A) | 57 | yes |
| Sequence A14 | dancing (iv) | 104 | yes |
| Sequence A15 | dancing (v) | 152 | yes |
| Sequence A16 | dancing (vi) | 98 | yes |
| Sequence A17 | kick + punch (D) | 55 | yes |

*Tracked until the end of the sequence but it had a problem in the presence of self-occlusion.

**Problem caused by movement velocity.

TABLE II. IMAGE SEQUENCE EVALUATION FOR VOLUNTEER B.

| Sequence Number | Movement | Number of Images | Tracking Until The End |
|---|---|---|---|
| Sequence B1 | dancing (i) | 99 | yes |
| Sequence B2 | dancing (iv) | 84 | yes |
| Sequence B3 | dancing (iii) | 84 | yes |
| Sequence B4 | dancing (ii) | 62 | yes |
| Sequence B5 | dancing (v) | 72 | yes |
| Sequence B6 | dancing (vi) | 79 | yes |
| Sequence B7 | punch (I) | 65 | yes |
| Sequence B8 | punch (II) | 75 | yes |
| Sequence B9 | kick (b) | 70 | yes |
| Sequence B10 | kick (a) | 79 | yes |
| Sequence B11 | kick + punch (C) | 73 | yes |
| Sequence B12 | kick + punch (D) | 74 | yes |
| Sequence B13 | kick + punch (B) | 99 | yes |
| Sequence B14 | kick + punch (A) | 97 | yes |

head and moving left foot. All of the six poses have the same class, which is *dancing*.

In the *playing guitar* class, the user imitates the moves of playing an instrument, shaking the right hand while the left hand stays at the same level as his/her shoulders. The *playing drums* class is when the user shakes his/her hands up and down alternately. There are two possible poses for the *punch* class, both of which have feet on the ground: (I) right hand and (II) left hand. Similar to the *punch*, the *kick* class can be made with: (a) right foot and (b) left foot, with both hands below the centroid. The *kick + punch* class can be made in four different poses: (A) kick with left foot and punch with left hand; (B) kick with left foot and punch with right hand; (C) kick with right foot and punch with right hand; and (D) kick with right foot and punch with left hand.

We used three different volunteers in our experiments: A, B and C. For each user, we collected a different number of sequences. Volunteer A is male, 1.76 meters tall, and has dark hair. Table I shows the collected sequences with Volunteer A. We collected 17 sequences with all of the classes.

Volunteer B is male, 1.90 meters tall and has blond hair. Volunteer B made 14 different sequences in four classes, all of them without self-occlusion. All of the possible poses for each of the four classes were collected. Table II details each of the collected poses from Volunteer B.

Volunteer C is female, 1.66 meters tall and has dark hair. Similar to Volunteer B, we collected sequences of four different classes with Volunteer C. Additionally, no problem was detected during the collection of the poses, which shows that the M5AIE method works well in sequences that do not have self-occlusions. We collected 13 sequences with Volunteer C because we wanted to test fewer training tuples with the pose *kick + punch* (A).

TABLE III. IMAGE SEQUENCE EVALUATION FOR VOLUNTEER C.

| Sequence Number | Movement | Number of Images | Tracking Until The End |
|---|---|---|---|
| Sequence C1 | dancing (i) | 48 | yes |
| Sequence C2 | dancing (iv) | 69 | yes |
| Sequence C3 | dancing (iii) | 45 | yes |
| Sequence C4 | dancing (ii) | 54 | yes |
| Sequence C5 | dancing (v) | 54 | yes |
| Sequence C6 | dancing (vi) | 45 | yes |
| Sequence C7 | punch (I) | 90 | yes |
| Sequence C8 | punch (II) | 88 | yes |
| Sequence C9 | kick (b) | 49 | yes |
| Sequence C10 | kick (a) | 54 | yes |
| Sequence C11 | kick + punch (C) | 90 | yes |
| Sequence C12 | kick + punch (D) | 100 | yes |
| Sequence C13 | kick + punch (B) | 85 | yes |

We observed that the M5AIE method had problems with poses that had self-occlusions. The problems were detected in the *playing guitar* and *playing drums* poses. This problem detection was crucial for the collection of the other users sequences; as a result, we avoided collecting these poses. However, we kept the results to make the tuples and test the classification algorithms. In only one sequence, the tracking method had problems that were caused by the movement velocity, but the pose classification was not affected.

As mentioned in Section IV, the dataset that was used for both the training and testing comprises the grid-coordinates that body parts assume at each frame of a set of image sequences that were produced for this work and the manual classification of the pose in each frame. We varied the number of cells of the grid in each frame, as follows: $8 \times 8$ (Table IV), $16 \times 16$ (Table V), $32 \times 32$ (Table VI) and $64 \times 64$ (Table VII).

The set of $k$ values for the KNN algorithm is $\{1, 3, 5, 7, 9, 11\}$, and different distances were used in our experiments. We combined the set of $k$ values with the Manhattan, Chebyshev and Euclidean distances. For each of the $N$ values of the grids $N \times N$, we made a data set that had all of the tuples from the three different users that made the described poses and 2128 tuples.

Table IV, where $N = 8$, shows that the Naïve Bayes Classifier gave the highest number of incorrectly classified instances ($21.22\%$). For all of the other classifiers, the percentage of instances that were correctly classified were above $93\%$. The C4.5 Gain Ratio Decision Tree had similar results as the KNN algorithm when $k >= 3$. As the $k$ value increased, the percentage of correctly classified instances decreased. Nevertheless, the Manhattan distance had the best results for every $k$ value. The best of all of the results in Table IV were with $K = 1$, primarily from using the Manhattan distance, with a $98.24\%$ correct. Most of the errors made by the classifier were from confusing *dancing* with *punch* and *kick + punch* classes.

Considering $N = 16$ (Table V), once more, the Naïve Bayes Classifier gave the highest percentage of incorrectly classified instances, with $28.74\%$. For all the other classi-

TABLE IV. RESULTS FOR $N = 8$.

| **Classification with Grid** $8 \times 8$ | **Correct*** | **Incorrect**** |
|---|---|---|
| C4.5 Gain Ratio Decision Tree | 97.26% | 2.74% |
| Naïve Bayes | 78.78% | 21.22% |
| KNN with K=1 and Manhattan Distance | 98.24% | 1.76% |
| KNN with K=1 and Chebyshev Distance | 98.07% | 1.93% |
| KNN with K=1 and Euclidean Distance | 98.24% | 1.76% |
| KNN with K=3 and Manhattan Distance | 97.79% | 2.21% |
| KNN with K=3 and Chebyshev Distance | 97.01% | 2.99% |
| KNN with K=3 and Euclidean Distance | 97.66% | 2.34% |
| KNN with K=5 and Manhattan Distance | 96.89% | 3.11% |
| KNN with K=5 and Chebyshev Distance | 95.94% | 4.06% |
| KNN with K=5 and Euclidean Distance | 96.60% | 3.40% |
| KNN with K=7 and Manhattan Distance | 96.23% | 3.77% |
| KNN with K=7 and Chebyshev Distance | 94.22% | 5.78% |
| KNN with K=7 and Euclidean Distance | 95.99% | 4.01% |
| KNN with K=9 and Manhattan Distance | 95.94% | 4.06% |
| KNN with K=9 and Chebyshev Distance | 93.32% | 6.68% |
| KNN with K=9 and Euclidean Distance | 95.86% | 4.14% |
| KNN with K=11 and Manhattan Distance | 96.31% | 3.69% |
| KNN with K=11 and Chebyshev Distance | 93.20% | 6.80% |
| KNN with K=11 and Euclidean Distance | 96.15% | 3.85% |

*Correctly Classified Instances
**Incorrectly Classified Instances

TABLE V. RESULTS FOR $N = 16$.

| **Classification with Grid** $16 \times 16$ | **Correct*** | **Incorrect**** |
|---|---|---|
| C4.5 Gain Ratio Decision Tree | 97.39% | 2.61% |
| Naïve Bayes | 71.26% | 28.74% |
| KNN with K=1 and Manhattan Distance | 98.84% | 1.16% |
| KNN with K=1 and Chebyshev Distance | 98.31% | 1.69% |
| KNN with K=1 and Euclidean Distance | 98.79% | 1.21% |
| KNN with K=3 and Manhattan Distance | 98.36% | 1.64% |
| KNN with K=3 and Chebyshev Distance | 96.33% | 3.67% |
| KNN with K=3 and Euclidean Distance | 97.97% | 2.03% |
| KNN with K=5 and Manhattan Distance | 98.02% | 1.98% |
| KNN with K=5 and Chebyshev Distance | 95.60% | 4.40% |
| KNN with K=5 and Euclidean Distance | 97.58% | 2.42% |
| KNN with K=7 and Manhattan Distance | 97.39% | 2.61% |
| KNN with K=7 and Chebyshev Distance | 95.22% | 4.78% |
| KNN with K=7 and Euclidean Distance | 97.29% | 2.71% |
| KNN with K=9 and Manhattan Distance | 97.20% | 2.80% |
| KNN with K=9 and Chebyshev Distance | 94.30% | 5.70% |
| KNN with K=9 and Euclidean Distance | 96.86% | 3.14% |
| KNN with K=11 and Manhattan Distance | 96.47% | 3.53% |
| KNN with K=11 and Chebyshev Distance | 92.90% | 7.10% |
| KNN with K=11 and Euclidean Distance | 96.18% | 3.82% |

*Correctly Classified Instances
**Incorrectly Classified Instances

TABLE VI. RESULTS FOR $N = 32$.

| **Classification with Grid** $32 \times 32$ | **Correct*** | **Incorrect**** |
|---|---|---|
| C4.5 Gain Ratio Decision Tree | 98.59% | 1.41% |
| Naïve Bayes | 76.17% | 23.83% |
| KNN with K=1 and Manhattan Distance | 99.72% | 0.28% |
| KNN with K=1 and Chebyshev Distance | 99.58% | 0.42% |
| KNN with K=1 and Euclidean Distance | 99.77% | 0.24% |
| KNN with K=3 and Manhattan Distance | 99.39% | 0.61% |
| KNN with K=3 and Chebyshev Distance | 98.45% | 1.55% |
| KNN with K=3 and Euclidean Distance | 99.34% | 0.66% |
| KNN with K=5 and Manhattan Distance | 99.34% | 0.66% |
| KNN with K=5 and Chebyshev Distance | 97.93% | 2.07% |
| KNN with K=5 and Euclidean Distance | 98.83% | 1.17% |
| KNN with K=7 and Manhattan Distance | 99.15% | 0.85% |
| KNN with K=7 and Chebyshev Distance | 97.32% | 2.68% |
| KNN with K=7 and Euclidean Distance | 98.64% | 1.36% |
| KNN with K=9 and Manhattan Distance | 98.73% | 1.27% |
| KNN with K=9 and Chebyshev Distance | 95.82% | 4.18% |
| KNN with K=9 and Euclidean Distance | 98.03% | 1.97% |
| KNN with K=11 and Manhattan Distance | 98.26% | 1.74% |
| KNN with K=11 and Chebyshev Distance | 95.21% | 4.79% |
| KNN with K=11 and Euclidean Distance | 97.37% | 2.63% |

*Correctly Classified Instances
**Incorrectly Classified Instances

TABLE VII. RESULTS FOR $N = 64$.

| **Classification with Grid** $64 \times 64$ | **Correct*** | **Incorrect**** |
|---|---|---|
| C4.5 Gain Ratio Decision Tree | 98.54% | 1.46% |
| Naïve Bayes | 77.02% | 22.98% |
| KNN with K=1 and Manhattan Distance | 99.81% | 0.19% |
| KNN with K=1 and Chebyshev Distance | 99.62% | 0.38% |
| KNN with K=1 and Euclidean Distance | 99.81% | 0.19% |
| KNN with K=3 and Manhattan Distance | 99.62% | 0.38% |
| KNN with K=3 and Chebyshev Distance | 98.26% | 1.74% |
| KNN with K=3 and Euclidean Distance | 99.34% | 0.66% |
| KNN with K=5 and Manhattan Distance | 99.44% | 0.56% |
| KNN with K=5 and Chebyshev Distance | 97.23% | 2.77% |
| KNN with K=5 and Euclidean Distance | 99.20% | 0.80% |
| KNN with K=7 and Manhattan Distance | 99.25% | 0.75% |
| KNN with K=7 and Chebyshev Distance | 96.76% | 3.24% |
| KNN with K=7 and Euclidean Distance | 98.92% | 1.08% |
| KNN with K=9 and Manhattan Distance | 99.01% | 0.99% |
| KNN with K=9 and Chebyshev Distance | 95.39% | 4.61% |
| KNN with K=9 and Euclidean Distance | 98.50% | 1.50% |
| KNN with K=11 and Manhattan Distance | 98.50% | 1.50% |
| KNN with K=11 and Chebyshev Distance | 94.55% | 5.45% |
| KNN with K=11 and Euclidean Distance | 97.84% | 2.16% |

*Correctly Classified Instances
**Incorrectly Classified Instances

fications, the incorrectly classified instances were less than $8\%$. The C4.5 Gain Ratio Decision Tree had only similar results with $k >= 7$ considering the Manhattan and Euclidean distances. If we consider only the values with the same value $k$, the Chebyshev distance gave the worst results. On the other hand, the Manhattan distance gave the best results. We could observe that the best results were obtained again with $k = 1$ and the Manhattan distance. Again, increasing the value of $k$, the results become worse for all of the used distances. The best percentage of correctness with $N = 16$ ($98.84\%$) was slightly better than with $N = 8$ ($98.24\%$), when both used $k = 1$ and the Manhattan distance. The *dancing* class was confused with the *playing guitar*, *punch* and *kick + punch* classes.

With $N = 32$, similar to with $N = 8$ and $N = 16$, the Naïve Bayes classifier gave the smallest percentage of correctly classified instances ($76.17\%$). All of the other results had more than $95\%$ correctness on instances of classification. If we compare the C4.5 algorithm with KNN (without the Chebyshev distance), we obtain similar results to when $k >= 9$. The best results were with $K = 1$ but with the Euclidean distance ($99.77\%$), which was followed very closely by the Manhattan distance ($99.72\%$). This result is even better than the best result in Table V. Most of the incorrectly classified instances occurred with instances of *dancing*, *punch* and *kick + punch*. Table VI shows the results for $N = 32$.

Table VII shows the results with $N = 64$. As was expected, the Naïve Bayes had $22.98\%$ incorrectly classified instances, followed by KNN with $k = 11$ and the Chebyshev distance, which had $5.45\%$ incorrect. All of the others gave more than $94\%$ correctly classified instances. The C4.5 algorithm had similar results with only KNN when $k = 11$. Similar to the other best results, in Table VII, KNN with $k = 1$ gave the best results with both distances, Manhattan and Euclidean, with exactly the same value, $99.81\%$. Because the results are very close to $100\%$ using $N = 64$, we could observe a relatively high number of errors using Naïve Bayes, which gave errors in the classes *dancing*, *punch* and *kick + punch*.

Until this point, we exposed the results, showing each table in an isolated way. However, we can observe additional results by comparing the tables with one another. All of the algorithms had similar results while considering the same algorithm with different $N$ values. In all of the cases, the worst results came from the Naïve Bayes Classifier. The C4.5 had

similar results with KNN depending on the $k$ value of each Table. Although the results are very similar from one table to another, we can see that the results of the C4.5 algorithm and KNN become better when $N$ becomes higher. Considering the distances, in general, the Manhattan gave the best results if we compare the same $k$ value in every Table. The Euclidean distance gave very similar results to the Manhattan, and only once the results from the Euclidean distance were better than the Manhattan distance. In all of the KNN experiments, the Chebyshev distance gave a percentage of incorrectly classified instances that was higher than for the other two considered distances.

Returning to the main goals of the experiments, we can conclude that:

1) With the exception of the Naïve Bayes, all of the other classifiers had at least $92\%$ of the instances classified correctly. With this result, we consider that the tested classifiers can make correct predictions with different poses of the same class.
2) Even using three different users to build our data set, we had a high number of instances correctly classified. Thus, we consider that the usage of different volunteers in our experiments did not affect the results. Moreover, these results showed that the use of the M5AIE method for body part detection and tracking works properly in movements without self-occlusions. Further experiments should be performed using dozens (or maybe hundreds) of volunteers to check whether the classification models would be affected. If this results are similar with a much larger number of volunteers, then the classification models are good for additional volunteers.
3) Because the results were improved with increasing values of $N$, the best results were given with $N = 64$. However, if we continue to increase the value of $N$, the results could be improved until a certain value. However, there is a possibility that, from a certain value on, the results might not become any better or they even might start to become worse. The last assumption is justified because the number of grids can become so large that each pixel could occupy more than one cell in the grid. Again, further experiments should be performed to find the exact value of $N$ for which the results obtain the best percentage of instances that are correctly classified.
4) The classification algorithm with the best results was the KNN algorithm with $k = 1$ while using the Manhattan distance.

As mentioned in 3, we believe that if we continue to increase the value of $N$, it could improve the results even more until a certain limit value is obtained. From that limit value for $N$ onward, the results could start to become worse (as mentioned in item 3, above). Perhaps if we normalized the coordinates according to the bounding box instead of a grid divided into cells, we could obtain the best results. We consider the KNN with $k = 1$ and the Manhattan distance as the winning algorithm in our experiments.

According to the concept of each distance measure, our inference for why we obtained the worst results using the Chebyshev distance is that this distance undervalues the distance between the body parts in each frame and the classifier makes mistakes when making its predictions. The Chebyshev distance gives the longest distance considering all of the axis distances from point A to another point B. Then, the body parts can be closer than they actually are to each other. However, the Manhattan and Euclidean distances can be more realistic for human movements. This last assumption should be the reason for the best results for the Manhattan distance, and the Euclidean distance gives very similar results in comparison to the Manhattan distance.

## VI. Conclusions and Future Work

We made a comparison among classification algorithms in human pose recognition and game context. In this paper we proposed and develop a detailed analysis using our own pose detection and tracking method, called M5AIE with different algorithms: C5.5 Gain Ratio Decision Tree [7], Naïve Bayes Classifier [8] and K-Nearest Neighbor (KNN) Classifier [9]. We applied the M5AIE method for detecting and tracking five main parts of the human body (head, hands and feet) in sequences of RGB-D images. Our method generates tuples that which can be used with different classifiers.

The proposed M5AIE algorithm was implemented in proof-of-concept programs. At this moment, we did not consider the computational load of this specific implementation to be a fundamental requirement because the main goal of this work is to assert the possibility of using a technique for body part detection, tracking and pose classification. The literature provide real-time results for the medial axis transform [34] and the SIFT method [35]. We believe the M5AIE can be efficiently implemented and used as part of real-time tracking solutions that are applied to games.

We limited our experiments to an indoor environment, static background, static position and orientation of the sensor and to single-user segmentation. Experiments showed that, to be correctly tracked, sequences must not have body part occlusions. The selected pose classes were inspired by the Jecripe game, developed by our group [24], and we added three more poses in our experiments. We used three volunteers with very different biotypes to collect the pose sequences with varation of the number of images and poses. Besides the different classification algorithms, we tested three kinds of distances: Manhattan, Chebyshev and Euclidean. We consider KNN with $k = 1$ and Manhattan distance as the winner because it provided the best results in all experiments. We believe the coordinates of the five main body parts could be normalized in the bounding box because. In our experiments, as long as we increased the division of the used grid (8, 16, 32 and 64), the results got better. However, we also believe there is a limit on dividing the grid. Further experiments should be done to find the value which the division does not make sense anymore. Also, further experiments should be done to prove that normalized coordinates could be a good choice in the usage of a bounding box for cell definition.

Future work include more experiments using dozens or hundreds of volunteers to check if the classification models would be affected. Also, more experiments to find the exact value of $N$ which the results get the best percentage of

instances correctly classified. This last task would give the limit value for $N$ which the results might start to get worse or not getting better anymore. Another future work should be the test approach. Instead of using $k$-fold cross-validation, we could use more tests approaches. The aim of vary the test approach is to combine results of each volunteer. For example, if we use tuples generated from Volunteer A and B, we could use as testing tuple only the generated tuples from Volunteer C, and so on.

More, future work also include the improvement of the M5AIE method with self-occlusion and partial occlusion treatment between two users. The detection and tracking method can be implemented to have real-time results. As a consequence of real-time results, we could integrate the M5AIE method in the game which motivated the poses in our experiments. The inclusion of the KNN with $k = 1$ and Manhattan distance would be also included in future research.

## ACKNOWLEDGMENT

We would like to thank the German Research Center for Artificial Intelligence (DFKI), CAPES-Brazil, and German Academic Exchange Service (DAAD).

## REFERENCES

[1] J. Shotton, A. Fitzgibbon, M. Cook, T. Sharp, M. Finocchio, R. Moore, A. Kipman, and A. Blake, "Real-time human pose recognition in parts from single depth images," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Colorado Springs, CO, USA, 2011, pp. 1297–1304.

[2] J. K. T. Tang, J. C. P. Chan, and H. Leung, "Interactive dancing game with real-time recognition of continuous dance moves from 3d human motion capture," in *Proceedings of the 5th International Conference on Ubiquitous Information Management and Communication*, ser. ICUIMC '11. New York, NY, USA: ACM, 2011, pp. 50:1–50:9.

[3] J. Shotton, R. Girshick, A. Fitzgibbon, T. Sharp, M. Cook, M. Finocchio, R. Moore, P. Kohli, A. Criminisi, A. Kipman, and A. Blake, "Efficient human pose estimation from single depth images," in *Decision Forests for Computer Vision and Medical Image Analysis*, ser. Advances in Computer Vision and Pattern Recognition. Springer London, 2013, pp. 175–192.

[4] G. Rogez, J. Rihan, S. Ramalingam, C. Orrite, and P. H. Torr, "Randomized trees for human pose detection," *Computer Vision and Pattern Recognition, IEEE Computer Society Conference on*, vol. 0, pp. 1–8, 2008.

[5] J. Huang, J. Lu, and C. X. Ling, "Comparing naive bayes, decision trees, and svm with auc and accuracy," in *IEEE International Conference on Data Mining*, 2003.

[6] N. B. Amor, S. Benferhat, and Z. Elouedi, "Naive bayes vs decision trees in intrusion detection systems," in *Proceedings of the 2004 ACM symposium on Applied computing*, ser. SAC '04. New York, NY, USA: ACM, 2004.

[7] J. R. Quinlan, *C4.5: programs for machine learning*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1993.

[8] P. Domingos and M. Pazzani, "On the optimality of the simple bayesian classifier under zero-one loss," *Machine learning*, vol. 29, no. 2, pp. 103–130, 1997.

[9] T. Cover and P. Hart, "Nearest neighbor pattern classification," *Information Theory, IEEE Transactions on*, vol. 13, no. 1, pp. 21–27, 1967.

[10] V. Mota, E. Perez, M. Vieira, L. Maciel, F. Precioso, and P. Gosselin, "A tensor based on optical flow for global description of motion in videos," in *Graphics, Patterns and Images (SIBGRAPI), 2012 25th SIBGRAPI Conference on*, 2012, pp. 298–301.

[11] R. Raskar, H. Nii, B. deDecker, Y. Hashimoto, J. Summet, D. Moore, Y. Zhao, J. Westhues, P. Dietz, J. Barnwell, S. Nayar, M. Inami, P. Bekaert, M. Noland, V. Branzoi, and E. Bruns, "Prakash: lighting aware motion capture using photosensing markers and multiplexed illuminators," *ACM Trans. Graph.*, vol. 26, July 2007.

[12] R. Y. Wang and J. Popović, "Real-time hand-tracking with a color glove," *ACM Trans. Graph.*, vol. 28, pp. 63:1–63:8, July 2009.

[13] L. Figueiredo, J. Teixeira, A. Cavalcanti, V. Teichrieb, and J. Kelner, "An open-source framework for air guitar games," in *Games and Digital Entertainment (SBGAMES), 2009 VIII Brazilian Symposium on*, 2009, pp. 74–82.

[14] R. N. de Almeida, "Portuguese sign language recognition via computer vision and depth sensor," Master's thesis, Lisbon University Institute, Lisbon, October 2011.

[15] L. Bourdev and J. Malik, "Poselets: Body part detectors trained using 3d human pose annotations," in *International Conference on Computer Vision*, sep 2009. [Online]. Available: http://www.eecs.berkeley.edu/ lbourdev/poselets

[16] L. Bourdev, S. Maji, T. Brox, and J. Malik, "Detecting people using mutually consistent poselet activations," in *European Conference on Computer Vision*, sep 2010. [Online]. Available: http://www.eecs.berkeley.edu/ lbourdev/poselets

[17] A. Bleiweiss, D. Eshar, G. Kutliroff, A. Lerner, Y. Oshrat, and Y. Yanai, "Enhanced interactive gaming by blending full-body tracking and gesture animation," in *ACM SIGGRAPH ASIA 2010 Sketches*, ser. SA '10. New York, NY, USA: ACM, 2010, pp. 34:1–34:2.

[18] V. Ganapathi, C. Plagemann, S. Thrun, and D. Koller, "Real time motion capture using a single time-of-flight camera," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, San Francisco, CA, USA, 2010, pp. 755–762.

[19] C. Schönauer, T. Pintaric, and H. Kaufmann, "Full body interaction for serious games in motor rehabilitation," in *Proceedings of the 2nd Augmented Human International Conference*, ser. AH '11. New York, NY, USA: ACM, 2011, pp. 4:1–4:8.

[20] T. Pintaric and H. Kaufmann, "Affordable infrared-optical pose-tracking for virtual and augmented reality," in *Proceedings of Trends and Issues in Tracking for Virtual Environments Workshop*, IEEE, Ed., vol. VR. IEEE, 2007.

[21] T. Qi, Y. Feng, J. Xiao, Y. Zhuang, X. Yang, and J. Zhang, "A semantic feature for human motion retrieval," *Computer Animation and Virtual Worlds*, vol. 24, no. 3-4, pp. 399–407, 2013.

[22] G. Cimen, H. Ilhan, T. Capin, and H. Gurcay, "Classification of human motion based on affective state descriptors," *Computer Animation and Virtual Worlds*, vol. 24, no. 3-4, pp. 355–363, 2013.

[23] M. Sun, P. Kohli, and J. Shotton, "Conditional regression forests for human pose estimation," in *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, 2012, pp. 3394–3401.

[24] A. Brandão, L. Brandão, G. Nascimento, B. Moreira, C. N. Vasconcelos, and E. Clua, "Jecripe: stimulating cognitive abilities of children with down syndrome in pre-scholar age using a game approach," in *Proceedings of the 7th International Conference on Advances in Computer Entertainment Technology*, ser. ACE '10. New York, NY, USA: ACM, 2010, pp. 15–18.

[25] E. Stone and M. Skubic, "Evaluation of an inexpensive depth camera for passive in-home fall risk assessment," in *5th International Conference on Pervasive Computing Technologies for Healthcare (PervasiveHealth), 2011*, 2011, pp. 71 –77.

[26] Blind for Revision, "Blind for revision," in *Blind for Revision*, 2012.

[27] H. Blum, "A transformation for extracting new descriptors of shape," *Models for the perception of speech and visual form*, vol. 19, no. 5, pp. 362–380, 1967.

[28] L. A. Schwarz, A. Mkhitaryan, D. Mateus, and N. Navab, "Human skeleton tracking from depth data using geodesic distances and optical flow," *Image Vision Comput.*, vol. 30, no. 3, pp. 217–226, Mar. 2012.

[29] C. Plagemann, V. Ganapathi, D. Koller, and S. Thrun, "Real-time identification and localization of body parts from depth images," in *Proceedings of the IEEE International Conference on Robotics & Automation (ICRA)*, Anchorage, Alaska, USA, 2010, pp. 3108–3113.

[30] E. Dijkstra, "A note on two problems in connexion with graphs," *Numerische Mathematik*, vol. 1, no. 1, pp. 269–271, 1959.

[31] J.-M. Morel and G. Yu, "Asift: A new framework for fully affine invariant image comparison," *SIAM J. Img. Sci.*, vol. 2, no. 2, pp. 438–469, Apr. 2009.

[32] G. Yu and J.-M. Morel, "Asift: An algorithm for fully affine invariant comparison," *Image Processing On Line*, p. Online, 2011.

[33] I. H. Witten and E. Frank, *Data Mining: Practical machine learning tools and techniques*. Morgan Kaufmann, 2011.

[34] F. Pinto and C. Freitas, "Fast medial axis transform for planar domains with general boundaries," in *Computer Graphics and Image Processing (SIBGRAPI), 2009 XXII Brazilian Symposium on*, 2009, pp. 96–103.

[35] L.-C. Chiu, T.-S. Chang, J.-Y. Chen, and N.-C. Chang, "Fast sift design for real-time visual feature extraction," *Image Processing, IEEE Transactions on*, vol. 22, no. 8, pp. 3158–3167, 2013.