Gesture Recognition Toolkit Using a Kinect Sensor

Marius-Cristian Giuroiu
Department of Computer Science
Technical University of Cluj-Napoca
Romania
E-mail: mariuscris9019@gmail.com

Tiberiu Marita
Department of Computer Science
Technical University of Cluj-Napoca
Romania
E-mail: Tiberiu.Marita@cs.utcluj.ro

***Abstract* - Computational modeling of human behavior has become a very important field of computer vision. Gesture recognition allows people to interact with machines in a natural way without the use of dedicated I/O devices. This paper presents a simple system that can recognize dynamic and static gestures using the depth map and the higher level output (skeleton and facial features) provided by a Kinect sensor. Two approaches are chosen for the recognition task: the Dynamic Time Warping Algorithm is used to recognize dynamic gestures, while a Bayesian classifier is used for the static gestures/postures. In contrast with some specialized methods presented in the literature, the current approach is very generic and can be used with minimal modification for recognizing a large variety of gestures. As a result, it can be deployed in a multitude of fields from security (monitoring rooms and sending alarm signals), medicine (helping people with physical disabilities) to education and so on. The tests results show that the system is accurate, easy to use and highly customizable.**

Keywords—human computer interaction; Kinect; depth map; gesture recognition; dynamic time warping; Naïve Bayes classifier.

I. INTRODUCTION

Gestures recognition systems have various applications in the field of human computer interaction especially in developing assistive technologies for sensory/motor impaired people, communication, cognition, education, home automation, smart cars etc. Some examples are mentioned bellow:

- Monitoring the drivers' attention when they are in traffic. The system can detect if a driver is tired or unfit to drive a car by analyzing his facial expressions and body language/gestures. Such a system can offer advices and trigger alarms [1].
- Creating perceptual human-computer interfaces. People with motoric disabilities can interact with a computer using their eyes movements or hands gestures [2].
- Sign language. A gestures recognition system can help people, who do not know sign language, to communicate more easily with their peers who are hearing impaired [3].
- Virtual control of certain electrical devices [4].

Regarding the type of the sensorial input used, several approaches were proposed for gesture recognition. Each solution deals with specific trade-offs regarding the equipment costs, computational complexity and results quality:

- By using low cost 2D cameras [5], additional algorithms that require intensive computing resources (segmentation, objects detection) must be used.
- The depth information provided by a stereo camera system eases the scene segmentation by reducing the search space for the object detection process but the system setup and the map generation are expensive, both financially and computationally [6].
- Wired gloves can be used to determine the position and rotation of the hands but they have only been available at a huge cost, with the finger bend sensors and the tracking device having to be bought separately [7].

Depth or range cameras [8] can generate directly a map of depth points of what is seen in a short-range. These points can then be further used to build a 3D representation of the scene.

The current paper will describe a system that is able to recognize body and facial static and dynamic gestures using an affordable depth camera: the Kinect sensor [9]. The system uses the depth information provided by Kinect [8] in conjunction with the functions provided by its SDK and toolkits [10], [11].

The Kinect device is described in detail in [12]. A list of advantages and disadvantages of the Kinect camera is also presented. The Kinect sensor is a human motion tracking peripheral which was first released by Microsoft in November, 2010 for the Xbox 360 console and in February, 2012 for the PC (version 1). The v.1 Kinect used in our experiments has a 640 x 480 32-bit RGB color camera and a 320 x 240 11-bit depth camera, both running at 30 frames per second. The field of view (FOV) is 58º(h) x 47º(v) and the depth range is between 0.4 … 4.5 m. Additionally it has a microphone array which can be used for detecting voice commands. The version 2 of the Kinect sensor, launched in 2014, came with some hardware and software improvements: increased resolution of the color (1920x1080) and depth (512x424) images with Time-Of-Flight (ToF) measuring principle, increased FOV of 70º(h) x 60º(v).

The Kinect v1.x SDK [10] provides the raw data for developing advanced gesture recognition, facial recognition and voice recognition applications. For example the Kinect's v.1 skeletal tracking tool is able to track up to 20 joints per active player but lacks in intrinsic gesture recognition implementations. Therefore gesture recognition application should be built by users over the SDK.

A tool that performs gesture recognition on the skeletal data using the Dynamic Time Warping (DTW) algorithm is described in [13]. This tool applies DTW in 2D (on the front-view projection of 3D body joints). A similar method that uses

3D coordinates and an improved version of the DTW algorithm by feature weighting is presented in [14]. In [15] a method for static gestures recognition using the Kinect sensor is presented. A set of classification techniques (back propagation neural network, support vector machine, decision tree, and Naïve Bayes) were tested and compared for 3 very simple body postures: stand, sit down, and lie down. In [16] a more complex dynamic gesture recognition method able to recognize five basic human emotional states using classifiers is presented. Unfortunately their approach has some shortcomings: metric features are not normalized (lacking in scale invariance) and their implementation does not provide real-time capabilities (probably due to the off-line data exchange method between the Kinect sensor and the Matlab processing module which is not explained in the paper).

Hands and fingers tracking represent important body parts used in vision based human computer interaction. In [17] the author proposes a unique approach for hands and fingers recognition using a Kinect sensor for search space reduction and further analysis of the 2D projection of the hand. In [18] the authors are performing the hand segmentation by combining both skin color and depth information and then fitting a 3D hand model to the observed data recovering the hand articulations. In [19] the authors propose a hand gesture recognition method based on a novel distance metric, Finger-Earth Mover's Distance (FEMD), to measure the dissimilarity between hand shapes

For now, Kinect is mostly used in games and in commercial applications that use animated avatars [20]. But in [21] it is proposed the usage of Kinect in ADAS (Advances driver assistance systems) for monitoring the driver's drowsiness or distraction while in [22] a driver awareness testing system is presented using a Kinect device as a driver interaction tool.

In the current work we propose to combine two different recognition techniques (DTW and Naïve Bayes Classifier) in order to create an original and generic system that can be configured to learn and recognize every type of gesture a human can make, even if it is a dynamic gesture or a static posture. In contrast most of the current gesture recognition systems based on the Kinect sensor are very specialized for some specific body parts or gesture set.

The same input is used for both techniques. We use 3D points for body and fingers movements, Animation Units (AU) for face expressions and head rotation angles for head movements. Every value used is normalized so the system works regardless of the body size and shape. The system can be used as a prototype when creating human-computer interfaces. By firing events when executing gestures we could easily control machines or we can use it in a large variety of applications as those mentioned in the introduction.

The next section will present in more detail the output of the Kinect sensor and its SDK. Section 3 will describe the DTW algorithm and Naïve Bayes classifier. Section 4 will describe how the DTW and the classification algorithms were tailored to recognize gestures, postures and expressions. Section 5 presents some experimental results of the proposed system. Last section will conclude the current work.

II. Output Of The Kinect Sensor And SDK

A. Skeleton tracking

The Kinect SDK offers free tools that can be used to develop simple applications for detecting and tracking the body/skeleton [10] and the head [11] of a person.

Skeletal Tracking allows Kinect to recognize people and follow their actions. Using the depth map, the sensor tracks up to 2 users in details and recognize up to 6 users in the field of view of the sensor. The tracked features are the joints of the body skeleton, each skeleton having a tracking ID.

The skeleton is modeled as a list of 3D points (x, y, z) that corresponds to 20 types of joints [23] (*HandLeft*, *HandRight*, *WristLeft*, *WristRight*, *ElbowLeft, ElbowRight, Head, Shoulder_Center, Spine, Hip_Center* etc.). This list of points can be used as input for the DTW algorithm or for fun applications like finding a user's height, moving the mouse by using one of the joints, or changing the slides of a presentation by waving the arm. Every joint can be in one of the following three states: *tracked* for a clearly visible joint, *inferred* when a joint is not completely visible and Kinect infers its position based on the position of neighboring points, or *non-tracked* if the point is not tracked at the moment.

B. The Candide face model

A framework for face tracking was included in Kinect SDK since version 1.5. This framework uses two libraries: *Microsoft.Kinect.Toolkit* and a derived library called *Face Tracking* [11]. This library can return up to 87 2D points of the human face (Fig. 1.a) and 13 additional points that are deduced from the existing ones (example : *center of the eye*, *center of the nose* etc.). These points can be used as inputs for different kinds of *Human Machine Interface* applications.

Having the position of the top of the head (joint of the head), a parameterized face model consisting in 113 vertices and 168 surfaces (Fig. 1.b) based on the CANDIDE-3 [24] model is provided by the SDK.

The *Face Tracking* module offers also the objects *frame.Rotation* containing the 3 rotations angles (*pitch*, *yaw* and *roll*) of the head around the 3 axes of the Kinect (Fig. 1 c), and *frame.Translation* object which contains the components of the translation vector $[T_x, T_y, T_z]$. We can use these values to find the position of the human head.

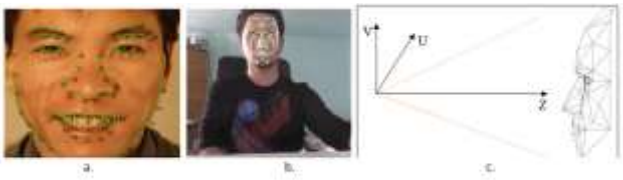

Fig. 1. Face points detection using FaceTracking (Kinect Toolkit) [11]: a. 2D points; b. 3D virtual surface extrapolated from 2D points using the CANDIDE-3 model [24]; c. Representation of rotations angles and translation vector

Facial gestures can be inferred based on the variability of lips, eyebrows and jaw position, lips stretcher, etc. These characteristics can be measured using 6 Animation Units (AU) and 11 Shape Units (SU), which are a subset of the CANDIDE-3 model.

The Shape Units (SU) are specifying the affected vertices and the displacement (x, y, z) per affected vertex and can be used to estimate the particular shape of the human face: the neutral position of the mouth, eyebrows, eyes, etc. The shape units are coded as follows: *SU0 - Head height, SU1 - Eyebrows vertical position, SU2 - Eyes vertical position, SU3 - Eyes width, SU4 - Eyes height, SU5 - Eye separation distance, SU8 - Nose vertical position, SU10 - Mouth vertical position, SU11 - Mouth width*.

The Animation Units (AU) represent deviations of the SU from the neutral shape of the face and are most commonly used to morph targets on animated avatar models so that the avatar acts as the tracked user does. Each AU is expressed as a numeric weight varying between -1 and +1 and are labeled as follows: *AU0 - Upper Lip Raiser, AU1 - Jaw Lowerer, AU2 - Lip Stretcher, AU3 - Brow Lowerer, AU4 - Lip Corner Depressor, AU5 - Outer Brow Raiser*. For a Neutral Face all AUs have null values.

III. Related Algorithms

A. Dynamic Time Warping (DTW) Algorithm

Given two time series $\mathbf{X} = (x_1, x_2, ..., x_N)$; $N \in N$ and $\mathbf{Y} = (y_1, y_2, ..., y_M)$; $M \in N$, DTW[25] finds the optimal solution in O(MN). This complexity can be improved using techniques like *multi-scaling* [26].

The data sequences must be sampled at equidistant points in time (problem that can be solved by re-sampling).

If we need to compare two sequences $\mathbf{X}, \mathbf{Y} \in \Phi$ which are taking values from another space of properties then we need to use a local distance measure:

$$d : \Phi \times \Phi \rightarrow R \geq 0 \quad (1)$$

Intuitively, d takes smaller values when the sequences are similar and bigger values when they are different. DTW is a *Dynamic Programming* algorithm so we can say that the distance function is a cost function. The task of finding the optimal alignment of the sequences becomes the task of arranging all sequence points by minimizing the cost function (or distance).

The algorithm starts by building the distance matrix $\mathbf{C} \in R^{NxM}$ which represents every pair of distance between **X** and **Y**. This distance matrix is called the local cost matrix of two sequences **X** and **Y**:

$$\mathbf{C}_t \in R^{NxM} : \mathbf{c}_{i,j} = \| x_i - y_j \|, i \in [1 : N], j \in [1 : M] \quad (2)$$

Once the cost matrix is built, the algorithm finds an alignment path between the two sequences. This path (also called warping path) defines the correspondence of $x_i \in X$ to $y_i \in Y$. The first and last elements of **X** and **Y** must be assigned to each other. This alignment path must have a minimal cost.

The alignment path build by DTW is a sequence of points $\mathbf{p} = (\mathbf{p}_1, \mathbf{p}_2, ..., \mathbf{p}_K)$ with $\mathbf{p}_k = (p_i, p_j) \in [1:N] \times [1:M])$, $k \in [1:K]$ which must satisfy the following conditions :

1. *The Boundary Criteria:* $\mathbf{p}_1 = (1, 1)$ and $\mathbf{p}_K = (N, M)$. Given 2 sequences **X** and **Y**. The first and the final point from the **X** sequence must be aligned with the first and last point from the **Y** sequence.
2. *Monotonicity condition:* $n_1 \leq n_2 \leq ... \leq n_K$ and $m_1 \leq m_2 \leq ... \leq m_K$. This condition keeps the order in time for the points.
3. *Step Size Condition:* this criteria limits big jumps (time displacements) in the alignment of the sequences. For starters, we will use the function:
$$\mathbf{p}_{k+1} - \mathbf{p}_k \in \{(1, 1), (1, 0), (0, 1)\}.$$

The cost function that corresponds with the warping path (minimum cost path) will be:

$$\mathbf{c}_p(X,Y) = \sum_{k=1}^{K} \mathbf{c}(x_{n_k}, y_{m_k}) \quad (3)$$

The minimum cost alignment path is called least cost path and is denoted by $\boldsymbol{P}^*$. To find a minimum cost alignment path we must find every alignment paths between **X** and **Y**.

This solution is very costly in terms of computation as the number of possible alignment paths increases exponentially when **X** and **Y** increase linearly. To prevent this from happening we must use *Dynamic Programming* [27] to implement DTW. The complexity will be reduced to O(MN).

The Dynamic Programming part of the DTW algorithm uses the following distance function:

$$\mathbf{D}(X, Y) = \mathbf{C}_p^*(\mathbf{X}, \mathbf{Y}) = \min \{ \mathbf{C}_p(X, Y), p \in \mathbf{P}^{NxM} \} \quad (4)$$

where P^{NxM} is the set of every possible alignment path and it builds the global cost matrix or accumulated cost matrix defined below:

1. First row : $\mathbf{D}(1, j) = \sum_{k=1}^{j} \mathbf{c}(x_1, y_k)$, $j \in [1, M]$
2. Fist column : $\mathbf{D}(i, 1) = \sum_{k=1}^{j} \mathbf{c}(x_k, y_1)$, $i \in [1, N]$
3. All other elements are:

$$\mathbf{D}(i, j) = \min \begin{Bmatrix} \mathbf{D}(i-1, j-1), \\ \mathbf{D}(i-1, j), \\ \mathbf{D}(i, j-1) \end{Bmatrix} + \mathbf{c}(x_i, y_j), i \in [1, N], j \in [1, M]).$$

The necessary time to build the matrix is O(NM). The algoritm that builds the matrix has **X**, **Y** and **C** as inputs. **C** is the matrix of local cost and represent all possible distances between **X** and **Y**. The algorithm to create the accumulated cost matrix is described in [9].

Once the accumulated cost matrix is build we can find the shortest path by going backwards from the point $\mathbf{p}_{end} = (M, N)$ to the point $\mathbf{p}_{start} = (1, 1)$ using the *Greedy method* described by the *OptimalWarpingPath* algorithm [25].

The following improvements can be added to the DTW algorithm:

1. *Step function (slope constraint):* if DTW makes *k* consecutive steps in one direction, it must make at least l steps in a diagonal direction.
2. *Weighting:* by adding weights to each of the distances based on the step direction we could penalize or favor certain types of point-to-point correspondences.

3. *Global path constraints:* the *Sakoe-Chiba band* [28] and *Itakura parallelogram* [29] which are constraining the warping range:

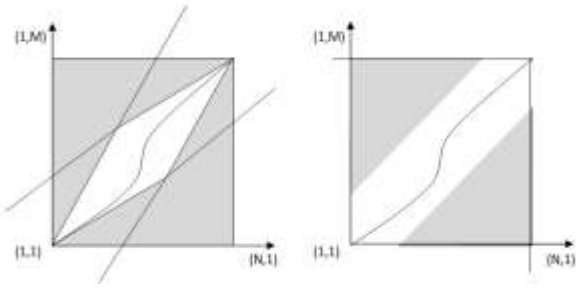

Fig. 2. a) Left - Itakura parallelogram b) right - Sakoe-Chiba band.

4. *Scaling:* by decreasing the length of the time series (size of N and M) using coarsening. Scaling can be obtained using a low-pass filter or linear approximation [30].

B. Hands and fingers detection

In the past, the majority of the algorithms for hands and fingers recognition were based on the pigmentation of the hands. The problem is that these algorithms can only be used on one kind of pigmentation, light or dark. So, depending on the person, the algorithm must be changed. However we can still use some parts of these algorithms in conjunction with depth information.

Two approaches can be used to recognize the hands and the fingertips:

- By using a large database of hands images in order to train the system to find the correspondence between them. This method is computationally expensive.
- Using some geometrical properties to approximate the location of the fingertips and the center of the palm.

The steps necessary to recognize the fingertips and the center of the palm according to the second approach could be the following [17]:

1. Generate a 2D region of interest (ROI) of the hand from the depth map
2. Decrease the noise
3. Classify the contour or inside pixels
4. Differentiate hands and calculate their contour
5. Allocate inside points
6. Find the center of the palm
7. Find the fingertips
8. Allocate points in a 3D space

C. Naïve Bayes Classifier

The Naïve Bayes Classifier technique is based on the so-called Bayesian theorem and is particularly suited when the dimensionality of the inputs is high. Despite its simplicity, Naïve Bayes can often outperform more sophisticated classification methods. The Naïve Bayes classifier is used in text classification, spam filter, online applications and so on.

1) Bayes' theorem

Known alternatively as Bayes' law or Bayes' rule, it relates current probability to prior probability. To predict future events we must use knowledge about past events [31].

$$P(h/D) = \frac{P(D/h)P(h)}{P(D)} \quad (5)$$

where:

P(h) : posterior probability of the hypothesis
P(D) : posterior probability of the training data set
P(h/D) : the probability of h given D
P(D/h) : the probability of D given h

2) MAP (Maximum A Posteriori) hypothesis generation

Generally we want to find the most probable hypothesis given the training data. We define:

hMAP = arg max P(h/D) (where h is part of H and H is the hypothesis space) [32].

$$hMAP = \arg\max \frac{P(D/h)P(h)}{P(D)}, \quad hMAP = \arg\max P(D/h)\,P(h) \quad (6)$$

3) ML (Maximum Likelihood) hypothesis generation

We assume that $p(h_i) = p(h_j)$ for all pairs i, j (uniform priors, i.e. $P_H \sim$ Uniform) [32]. We can further simplify and choose the maximum likelihood hypothesis as:

$$hML = \arg\max P(D/h_i) \text{ (where } h_i \text{ is part of H)} \quad (7)$$

4) Naïve Bayes Classification

It is based on the Bayes' theorem and is used especially when the dimensionality of the input is high. Parameter estimation for Naïve Bayes models uses the ML method. Despite the simplified assumptions, in many complex real situations Naïve Bayes performs very well. The big advantage of this type of classification is that it can use very little training data to estimate the parameters [32].

Considering D a set of tuples, where each tuple is a n-dimensional vector: $\mathbf{X}(x_1, x_2, x_3, \ldots, x_n)$ and having the classes $C_1, C_2, C_3, \ldots, C_m$ the Naïve Bayes classifier can predict that **X** belongs to the C_i class if:

$$P(C_i/\mathbf{X}) > P(C_j/\mathbf{X}) \text{ for } i <= j <= m,\ j <> i \quad (8)$$

where $P(C_i/\mathbf{X})$ is the Maximum Posteriori hypothesis :

$$P(C_i/\mathbf{X}) = \frac{P(\mathbf{X}/Ci)P(Ci)}{P(\mathbf{X})} \quad (9)$$

The solution is to maximize $P(\mathbf{X}/C_i)P(C_i)$ because $P(\mathbf{X})$ is a constant. Because it is very expensive computationally to evaluate $P(\mathbf{X}/C_i)$ the Naïve assumptions of *class conditional independence* is used:

$$P(\mathbf{X}/C_i) = \prod_{k=1}^{N} P(x_k/Ci) \quad (10)$$

$$P(\mathbf{X}/C_i) = P(x_1/C_i) * P(x_2/C_i) * \ldots * P(x_n/C_i)$$

IV. Gesture Recognition Mechanism

A. Dynamic gesture recognition using the DTW algorithm

A feature vector is created using values retrieved from the Kinect sensor. This vector may need to be centered using a reference point and then normalized using a specific anthropomorphic feature of the person. Normalization insures the scale invariance of the processed data (for example the height of a person). Training sequences $\boldsymbol{Y}_q$ are created for every gesture needed to be recognized. During a gesture, *M* temporal instances of the feature vector are registered (*M* depends on the duration of the gesture). In order to cope with the inter/intra person gesture execution variability up to six

time series are registered per gesture (labeled with the same gesture class). Finally, DTW can be applied on a test sequence **X**. This procedure tries to find the best match (warping path) between a subset of *M* temporal instances of **X** and every set $\boldsymbol{Y}_q$.

1) Body and fingers gestures recognition

1. A feature vector is created using the coordinates of the *upper body joints* / *fingertips*. These are 3D points. The Kinect SDK can return up to 20 body joint coordinates (3D points) for a person for every frame. The algorithm described in [17] can retrieve up to 6 points (5 fingers and the center of the palm) for both hands of a person.
2. A reference point is chosen - *middle point of the segment between the two shoulders joints* / *the center of the palm* $\Rightarrow (U_r, V_r, Z_r)$
3. The *joints* / *fingertips* coordinates (U_i, V_i, Z_i) (i = 1..J / J - the number of control points) are centered against the reference point:

$$U_i^c = U_i - U_r$$
$$V_i^c = V_i - V_r$$
$$Z_i^c = Z_i - Z_r$$

4. The centered coordinates are normalized using the Euclidian distance between *the joints of the shoulders* / *the center of the palm and the elbow*. The set (U_i^n, V_i^n, Z_i^n) is obtained. The feature vector has the following form:

$$\mathbf{v}_{3D} = \{(U_i^n, V_i^n, Z_i^n), i = 1..J\}$$

5. A training sequence is created for every gesture that needs to be recognized. During a gesture, *M* temporal positions of the *body members* / *fingers* are registered:

$$\mathbf{Y}_q = \{ \cup_{k=1}^{M} (U_i^n, V_i^n, Z_i^n), i = 1 .. j\}, q = 1..N$$

where: N – is the number of recognizable gestures.

6. The DTW algorithm is applied on a test sequence **X** in order to recognize a specific body/hand gesture.

2) Facial gestures recognition

1. A feature vector/set is created using the animation units (AU) for a temporal instance :

$$V_{2D\text{-}face\text{-}AU} = (AU_0, AU_1, AU_2, AU_3, AU_4, AU_5)$$

2. A training sequence is created for every gesture that needs to be recognized. During a gesture *M* temporal instances of the feature vector's AUs are registered:

$$\mathbf{Y}_q = \cup_{K=1}^{M} (AU_0, AU_1, AU_2, AU_3, AU_4, AU_5), \ q = 1..N,$$

where: N – is the number of recognizable gestures.

3. The DTW algorithm is applied on a test sequence **X** in order to recognize a specific facial gesture.

3) Head gestures recognition

To recognize head gestures we follow the same steps as for *facial gestures recognition*. The only difference is that the feature vector is created using the rotation angles of the head: $V_{Head\_Rotation} = (X, Y, Z)$.

B. Static gesture recognition using the Naïve Bayes classifier

The WEKA tool [33] is used to create classification models for programmatic use. These models can be manipulated and used to evaluate data. The result of the evaluation indicates if a static gesture is recognized or not. The same data structure that is used for dynamic gestures can be used to create the classification models for static gestures.

The following process is applied for every category of static gestures (body, fingers or face static gestures):

1. Training data is created using feature vectors captured over multiple frames for every static gesture. The feature vectors are normalized.
2. An initial model is created using the training data, the Naïve Bayes classifier and 10-fold cross-validation.
3. The model is loaded into the system.
4. For every frame the system can recognize static gestures by running normalized test feature vectors against the trained model.

V. TESTING AND VALIDATION

We tested the recognition of dynamic gestures and also postures (static gestures). Due to the limited space in the paper's presentation, only the results for the quantitative evaluation of the body and facial gestures recognition are presented here. A qualitative assessment of the results for all types of static and dynamic gestures recognition can be watched in some test movies at the links provided at the end of the paper (Media Files section).

In order to assess the performance of the system the confusion matrix was used. The following gestures were tested during the evaluation:

- 8 types of body dynamic gestures :
  a = Left_Hand_Goes_Up
  b = Left_Hand_Goes_To_The_Left
  c = Left_Hand_Goes_To_The_Right
  d = Right_Hand_Goes_Up
  e = Right_Hand_Goes_To_The_Left
  f = Right_Hand_Goes_To_The_Right
  g = Both_Hands_Go_Up
  h = Both_Hands_Go_Down
- 8 type of body static gestures (postures):
  a = Left_Hand_Up
  b= Left_Hand_To_The_Left
  c = Left_Hand_To_The_Right
  d = Right_Hand_Up
  e = Right_Hand_To_The_Left
  f = Right_Hand_To_The_Right
  g = Both_Hands_Up,
  h = Both_Hands_Down
- 5 types of face dynamic gestures/static expressions:
  a = Neutral
  b = Yawn
  c = Make_Frowny_Face
  d = Be_Happy
  e = Raise_Eyebrows.

The attributes used for body gestures / postures are: *HandLeft_X*, *HandLeft_Y*, *HandLeft_Z*, *WristLeft_X*, *WristLeft_Y*, *WristLeft_Z*, *ElbowLeft_X*, *ElbowLeft_Y*, *ElbowLeft_Z*, *ElbowRight_X*, *ElbowRight_Y*, *ElbowRight_Z*, *WristRight_X*, *WristRight_Y*, *WristRight_Z*, *HandRight_X*, *HandRight_Y*, *HandRight_Z.*

The attributes used for face gestures / expressions are: *BrowLover*, *BrowRaiser*, *JawLower*, *LipCornerDepressor*, *LipRaiser*, *LipStretcher*.

A. Testing the classifiers

We captured 198 frames of data (or feature vectors that contain normalized attributes) for every type of static gesture, in order to cope with the intra/inter person gesture execution variability. We used the WEKA tool to train the models. The total number of trained instances for body gestures was 1584 (198 frames x 8 gestures) and for face gestures was 990 (198 frames x 5 gestures). After testing multiple classifiers we decided to choose the Naïve Bayes classifier (i.e. tables I and II present such a comparison between the Naïve Bayes and Lazy LWL classifiers). Tables III and VI show the Naïve Bayes classifier performance for the provided training data set.

TABLE I. COMPARISON BETWEEN CLASSIFIERS WHEN LEARNING BODY POSTURES

Classifier / Attribute	Naïve Bayes	Lazy LWL
Total number of instances	1584	1584
Attributes	18	18
Test options	10-fold cross-valid.	10-fold cross-valid.
Correct classified instances	1584 100 %	1581 99.9369 %
Wrong classified instances	0 0 %	1 0.0631 %
Kappa statistic	1	0.9993
Mean absolute error	0	0.0745
Root mean squared error	0	0.1415
Relative absolute error	0 %	34.0365 %
Root relative squared error	0 %	42.7787 %

TABLE II. COMPARISON BETWEEN CLASSIFIERS WHEN LEARNING FACE EXPRESSIONS

Classifier / Attribute	Naïve Bayes	Lazy LWL
Total number of instances	990	990
Attributes	6	6
Test options	10-fold cross-valid.	10-fold cross-valid.
Correct classified instances	982 99.1919 %	980 98.9899 %
Wrong Classified instances	8 0.8081 %	10 1.0101 %
Kappa statistic	0.9899	0.9874
Mean absolute error	0.0079	0.113
Root mean squared error	0.0554	0.2152
Relative absolute error	1.437 %	35.32.39 %
Root relative squared error	13.8426 %	53.7905

TABLE III. CLASSIFIER PERFORMANCE FOR STATIC FACE EXPRESSIONS RECOGNITION USING THE NAÏVE BAYES CLASSIFIER

Class code	TP Rate	FP RATE	Precision	Recall	F-Measure	ROC Area	Class name
a	0.985	0.006	0.975	0.985	0.98	0.999	Neutral
b	1	0	1	1	1	1	Yawn
c	1	0	1	1	1	1	Frowny_Face
d	0.99	0.004	0.985	0.99	0.987	1	Happy
e	0.99	0	1	0.985	0.992	1	Raise_Eyebrows
Avg.	0.992	0.002	0.992	0.992	0.992	1	

TABLE IV. CLASSIFIER PERFORMANCE FOR BODY POSTURES RECOGNITION USING THE NAÏVE BAYES CLASSIFIER

Class code	TP Rate	FP RATE	Prec.	Recall	F-Measure	ROC Area	Class name
a	1	0	1	1	1	1	Left_Hand_Up
b	1	0	1	1	1	1	Left_Hand_To_The_L
c	1	0	1	1	1	1	Left_Hand_To_The_R
d	0.995	0	1	0.995	0.997	0.995	Right_Hand_Up
e	1	0	1	1	1	1	Right_Hand_To_The_L
f	1	0.001	0.995	1	0.997	1	Right_Hand_To_The_Ri
g	1	0	1	1	1	1	Both_Hands_Up
h	1	0	1	1	1	1	Both_Hands_Down
Avg.	0.999	0	0.999	0.999	0.999	0.999	

B. Testing the System

The following methodology was used to test the system:

- Record a series of static and dynamic gestures for two different persons executing the gestures/postures in front of the Kinect camera.
- In order to cope with the inter/intra person gesture execution variability, each person's gesture execution was tested/repeated for about 20 times.
- Set the Kinect sensor to: near mode and seated (we are interested only in the upper body joints).

1) Testing static gestures using the Naïve Bayes classifier

For the body postures/facial expressions the test scenarios contained the following key points:

- 5 types of facial and 8 types of body postures expressions (as shown in Fig. 3 and 4) were tested.
- Every person should remain approximately in the same position / have the same expression for at least a few frames in order to consider the gesture as recognized.

Fig. 3. Examples of tested static face gestures/expressions.

Fig. 4. Examples of tested static body gestures/postures.

The system's performance for static gestures/expressions is shown in the form of the confusion matrix and precision (positive predictive value): PPV=TP/(TP+FP) [34] (Tables V and VI). The recognition of facial gestures **a**, **c** and **d** obtained a lower PPV compared with **b** and **e** because the degree of similarity between them is higher. This can be observed from the confusion matrix: **a** was recognized as **c** 6 times and as **d** 3 times, **d** was recognized as **c** 5 times and **c** was recognized as **d** 4 times.

TABLE V. SYSTEM PERFORMANCE FOR FACE EXPRESSIONS RECOGNITION USING THE NAÏVE BAYES CLASSIFIER (CONFUSION MATRIX AND PRECISION/PPV)

a	b	c	d	e		TP	FP	PPV[%]
27	2	6	3	2	**a**	27	13	67,5
0	31	3	3	3	**b**	31	9	77,5
3	2	28	4	3	**c**	28	12	70
4	2	5	27	2	**d**	27	13	67,5
4	0	2	1	33	**e**	33	7	82,5
38	**37**	**44**	**38**	**43**	**Total**	**146**	**54**	**73**

TABLE VI. SYSTEM PERFORMANCE FOR BODY POSTURES RECOGNITION USING THE NAÏVE BAYES CLASSIFIER (CONFUSION MATRIX AND PRECISION/PPV)

a	b	c	d	e	f	g	h		TP	FP	PPV[%]
31	3	4	0	0	0	2	0	**a**	31	9	77,5
3	31	3	0	1	0	0	2	**b**	31	9	77,5
5	3	27	0	1	0	0	4	**c**	27	13	67,5
0	0	1	32	3	1	3	0	**d**	32	8	80
0	0	1	3	28	4	3	1	**e**	28	12	70
0	0	0	1	2	30	3	4	**f**	30	10	75
3	1	1	2	0	0	33	0	**g**	33	7	82.5
0	1	2	1	2	1	0	33	**h**	33	7	82.5
42	**39**	**39**	**39**	**37**	**36**	**44**	**44**	**Total**	**245**	**75**	**76,5**

2) Testing dynamic gestures using the DTW algorithm

For the dynamic body and facial gestures the test scenarios contained the following key points:

- 5 types of dynamic facial gestures and 8 types of dynamic body gestures (as defined at the beginning of chapter)
- Every gesture tested consisted in a time series of 32 frames (sets of feature vectors)
- Each test sequences was compared with the registered instance using the DTW algorithm

The system's performance for dynamic gestures is shown in the form of the confusion matrix and PPV (tables VII and VIII):

TABLE VII. SYSTEM PERFORMANCE FOR DYNAMIC FACE GESTURES RECOGNITION (CONFUSION MATRIX AND PRECISION/PPV)

a	b	c	d	e		TP	FP	PPV[%]
29	1	5	3	2	**a**	29	11	72,5
1	33	2	2	2	**b**	33	7	82,5
4	1	28	5	2	**c**	28	12	70
6	1	4	27	2	**d**	27	13	67,5
2	1	2	3	32	**e**	32	8	80
42	**37**	**41**	**40**	**40**	**Total**	**149**	**51**	**74,5**

TABLE VIII. SYSTEM PERFORMANCE FOR DYNAMIC BODY GESTURES RECOGNITION (CONFUSION MATRIX AND PRECISION/PPV)

a	b	c	d	e	f	g	h		TP	FP	PPV[%]
34	3	2	0	0	0	1	0	**a**	34	6	85
3	33	3	0	1	0	0	0	**b**	33	7	82,5
3	4	27	0	4	0	1	1	**c**	27	13	67,5
0	0	1	32	3	3	1	0	**d**	32	8	80
0	0	4	4	26	4	1	1	**e**	26	14	65
0	0	0	1	4	34	1	0	**f**	34	6	85
2	0	0	2	1	0	35	0	**g**	35	5	85,5
0	0	1	0	2	0	0	37	**h**	37	3	92,5
42	**40**	**38**	**39**	**41**	**41**	**40**	**39**	**Total**	**258**	**62**	**80,6**

The results are encouraging considering the small size of the training data. From our observations during testing, the size of the training data and the degree of similarity between gestures dictates the PPV up to a point.

VI. CONCLUSIONS

The implemented system represents a typical interface for the natural human computer interaction paradigm, without the help of classical peripherals like mousses or keyboards. The system can recognize static gestures (body postures / facial expressions) and more complex dynamic gestures in real time at up to 30 frames per second.

The recognition of dynamic gestures was done using the *Dynamic Time Warping* (DTW) algorithm applied on series of 3D features provided by the Kinect SDK (skeleton joints) or inferred from the depth map (finger tips). For facial gestures, the animation units (AUs) provided by the Kinect SDK were used.

The recognition of static gestures was done by implementing a Naïve Bayes classifier model using the WEKA tool. The same data structures that were used for dynamic gestures recognition were used to create the classification models for the static gestures.

Regarding the performance of the dynamic body gestures recognition method, the obtained results are similar with the ones presented in [14]. Regarding the static body gesture recognition method, our approach outperformed with about 20% the results reported in [15] in the case of the Naïve Bayes classifier, even for far more complex static gestures/postures. The better performance of other classification methods (i.e. backpropagation, neural networks, support vector machines, decision trees - as reported in [15] and [16]) will be further investigated for our complex gestures set and our real time constraints.

For the facial expressions and gestures the recognition performance was a little bit (3 % for the static approach and 6 % for dynamic approach) lower. This can be explained due to the increased complexity and variability of the face appearance and consequently lower error margins for the face features (AUs) computation that were reflected in the classification and DTW models. No similar method based on the Kinect's facial features was found in literature for a quantitative performance comparison of the method.

After performing validation test on several scenarios the following conclusions can be drawn:

- The degree of similarity/dissimilarity between gestures has an important impact on the recognized gestures precision/PPV.
- The system proved reliable because it has a good gestures recognition precision/PPV even if the system wasn't intensively taught with multiple persons.
- To improve the precision of the recognized gestures the system must be trained with multiple persons. Every person should perform the entire set of gestures multiple times.
- The application presents a high level of usability because it has a graphical interface that is easy to learn and use.

The algorithms that were implemented can be further improved. Such an improvement would be the changing of the DTW algorithm with its faster version *FastDTW* [26]. This version has linear time and space complexity and it uses a multilevel approach that recursively projects a solution from a coarser resolution and refines the projected solution. Another improvement is to add the possibility of using multiple Kinect sensors to provide input for the system (we may run into some interference problems [35]). Another idea would be to make the recognition mechanism be accessible as a web service. This would enable us to provide gesture recognition services deployed in cluster architecture. The performance would be significantly improved and would make the Kinect sensor be interchangeable with another type of depth camera.

MEDIA FILES

Results for static gestures recognition (classification approach):

- Body static gestures: https://vimeo.com/128776557
- Face static gestures: https://vimeo.com/128776558
- Finger static gestures: https://vimeo.com/128776559

Results for dynamic gestures recognition (DTW approach). Results of the static gestures recognition performed on individual frames are also shown for some gestures categories:

- Body dynamic gestures: https://vimeo.com/128787791
- Face dynamic gestures: https://vimeo.com/128787795
- Fingers dynamic gestures: https://vimeo.com/128787793
- Head orientation dynamic: https://vimeo.com/128787794

REFERENCES

[1] Q. Ji, X. Yang, "Real-time eye, gaze, and face pose tracking for monitoring driver vigilance." *Real-Time Imaging*, vol.8, no.5, pp. 357-377, 2002.

[2] P. Jia, H. Hu, T. Lu, K. Yuan, "Head gesture recognition for hands-free control of an intelligent wheelchair," *Industrial Robot: An International Journal*, vol. 34, no.1, pp. 60-68, 2007.

[3] D. Kumarage *et al.*, "Real-time sign language gesture recognition using still-image comparison & motion recognition," in *Proc. of 6th IEEE International Conference on Industrial and Information Systems (ICIIS)*, 2011, pp. 169-174.

[4] X. Zhang, *et al.*, "Hand gesture recognition and virtual game control based on 3D accelerometer and EMG sensors," In *Proc. of the 14th International Conference on Intelligent User Interfaces*, 2009, pp. 401-406.

[5] G. Ushaw *et al.*, "An Efficient Application of Gesture Recognition from a 2D Camera for rehabilitation of Patients with Impaired Dexterity," in *HEALTHINF*, 2013, pp. 315-318.

[6] X. Li, J. An, J. Min, K. Hong, "Hand gesture recognition by stereo camera using the thinning method," in *Proc. of International Conference on In Multimedia Technology (ICMT)*, 2011, pp. 3077-3080.

[7] M. Ganzeboom, (2011), "How hand gestures are recognized using a dataglove," *Human Media Interaction (HMI)* [Online]. Available: http://hmi.ewi.utwente.nl/verslagen/capita-selecta/CS-Ganzeboom-Mario-3.pdf

[8] B. Langmann *et al.*, "Depth Camera Technology Comparison and Performance Evaluation," in *Proc. of the 1st International Conference on Pattern Recognition Applications and Methods* (ICPRAM) 2012, pp. 438-444.

[9] Microsoft (2013, June). *Kinect for Windows* [Online]. Available: http://www.microsoft.com/en-us/kinectforwindows/discover/features.aspx.

[10] Microsoft Developer Network (2015, April), *Kinect for Windows SDK* [Online]. Available: https://msdn.microsoft.com/en-us/library/hh855347.aspx)

[11] Microsoft Developer Network (2015, April). *Face Tracking* [Online]. Available: http://msdn.microsoft.com/en-us/library/jj130970.aspx.

[12] H. H. Wu, A. Bainbridge-Smith. (2015, April). *Advantages of using a Kinect Camera in various applications* [Online]. Available: http://www.academia.edu/2070005/Advantages_of_using_a_Kinect_Camera_in_various_applications

[13] (2013 June). *Kinect gesture SDK* [Online]. Available: http://kinectdtw.codeplex.com/documentation.

[14] M. Reyes *et al.*, "Feature weighting in dynamic timewarping for gesture recognition in depth data," in *Proc. of 2011 IEEE International Conference on Computer Vision Workshops*, 2011, pp. 1182-1188.

[15] O. Patsadu, „Human gesture recognition using Kinect camera," in *Proc. Of 2012 International Joint Conference on Computer Science and Software Engineering (JCSSE)*, 2012, pp. 28 – 32.

[16] S. Saha *et al.*, "A study on emotion recognition from body gestures using Kinect sensor," in *Proc. of IEEE 2014 International Conference on Communications and Signal Processing (ICCSP)*, 2014, pp. 056-060.

[17] F. Trapero Cerezo, "3D Hand and Finger Recognition using Kinect," Universidad de Granada (UGR), Spain, Tech. Rep. 2012.

[18] I. Oikonomidis *et al.*, "Efficient model-based 3D tracking of hand articulations using Kinect", In Proc. of the 22nd British Machine Vision Conference, BMVC'2011, vol. 1, no. 2, pp. 3

[19] Z. Ren *et al.*, "Robust part-based hand gesture recognition using kinect sensor," *IEEE Transactions on Multimedia*, vol.15, no.5, pp. 1110-1120, 2014.

[20] (2013, March). *Face Shift* [Online]. Available: http://www.faceshift.com/.

[21] L. Li *et al.*, "Multi-sensor soft-computing system for driver drowsiness detection," in *Soft Computing in Industrial Applications*, Springer International Publishing, 2014, pp. 129-140.

[22] A. Turpen. (2015, Apr). *Toyota using Driver Awareness Research Vehicle with Microsoft Kinect to combat distracted driving* [Online]. Avail.:http://www.carnewscafe.com/2013/11/toyota_darv_microsoft_kinect/

[23] K. Aitpayev, J. Gaber, "Creation of 3D Human Avatar using Kinect," *Asian Transactions on Fundamentals of Electronics, Communication & Multimedia*, vol.1, no.5, pp. 12-24, 2012.

[24] (2013 March). *CANDIDE-3 - A parameterized face* [Online]. Available: http://www.icg.isy.liu.se/candide/

[25] P. Senin, "Dynamic time warping algorithm review," Information and Computer Science Department University of Hawaii at Manoa Honolulu, USA, Tech. Rep. 2008, pp. 1-23.

[26] S. Salvador, P. Chan, "FastDTW: Toward accurate dynamic time warping in linear time and space," *Intelligent Data Analysis*, vol.11, no.5, pp. 561-580, 2007.

[27] (2013 June). *A Tutorial on Dynamic programming* [Online]. Available: http://www.avatar.se/lectures/molbioinfo2001/dynprog/dynamic.html

[28] H. Sakoe, S. Chiba, "Dynamic programming algorithm optimization for spoken word recognition," *IEEE Trans. Acoust., Speech, Signal Process.*, vol. 26, no.1, pp. 43-49, 1978.

[29] F. Itakura, "Minimum prediction residual principle applied to speech recognition," *IEEE Trans. Acoust., Speech, Signal Process.*, vol. 23, no.1, pp. 67-72, 1975.

[30] S. Chu *et al.*, "Iterative Deepening Dynamic Time Warping for Time Series," in *Proc 2 nd SIAM International Conference on Data Mining*, 2002, pp. 195-212.

[31] (2015, April). *Naïve Bayes Classifier Introductory Overview* [Online]. Available: http://www.statsoft.com/textbook/Naïve-bayes-classifier

[32] W. Hsu. (2015 April). *Bayes's Theorem, MAP, and Maximum Likelihood Hypotheses*, CIS 732: Machine Learning and Pattern Recognitio, Kansas State University [Online]. Available: http://www.kddresearch.org/Courses/Spring-2008/CIS732/Lectures/Lecture-06-20080204.pdf

[33] M. Hall *et al.*, "The WEKA Data Mining Software: An Update," *ACM SIGKDD explorations newsletter* , vol. 11, no 1, 2009, pp. 10-18.

[34] (2015 May). *Confusion matrix* [Online]. Available: https://en.wikipedia.org/wiki/Confusion_matrix

[35] A. Maimone, H. Fuchs, "Reducing interference between multiple structured light depth sensors using motion," in *2012 IEEE Virtual Reality Short Papers and Posters (VRW), 2012,* pp. 51-54.