Machine learning for real time poses classification using Kinect skeleton data

Youness Choubik[1], Abdelhak Mahmoudi[2]
[1] Faculty of Sciences Rabat, [2] Ecole Normale Superieure Rabat
[1] youness.choubik@gmail.com, [2] abdelhak.mahmoudi@gmail.com

Abstract

Poses recognition is an important research topic because some situations require silent communication (sign language, surgeon poses to the nurse for assistance etc.). Traditionally, poses recognition requires high quality expensive cameras and complicated computer vision algorithms. This is not the case thanks to the Microsoft Kinect sensor which provides an inexpensive and easy way for real time user interaction. In this paper, we proposed a real time human poses classification technique, by using skeleton data provided by the Kinect sensor. Different users performed a set of tasks from a vocabulary of eighteen poses. From skeleton data of each pose, twenty features are extracted so that they are invariant with respect to the users size and its position in the scene. We then compared the generalization performances of four machine learning algorithms; support vectors machines (SVM), artificial neural networks (ANN), k-nearest neighbors (KNN) and Bayes classifier (BC). The method used in this work shows that SVM outperforms the other algorithms.

Keywords— **Poses classification, Kinect, SVM, Cross-validation.**

1 Introduction

Machine learning has undergone significant developments over the past few years, and multiple application domains saw their appearance. The Kinect sensor [2] was introduced to the market in 2010, and offered Natural User Interface (NUI), which allowed to use body movement, voice commands, interpret gestures and facial expression, speech recognition, and also environment recognition. The kinect sensor have been introduced in poses recognition applications, such as building a translation system that translates sign language to spoken language, like Edon and Konstantinos which used hand shape recognition to translate a limited vocabulary of Kosova Sign Language [6], and Lang who used Hidden Markov Models with a continuous observation density [3].

Poses classification was approached in different ways. Recently machine learning techniques where introduced. Among related works, Miranda et al. use a tailored angular representation of the skeleton joints, then identify gestures through Support Vector Machines and Decision Forest [5]. Liu and Lovell work with a Hidden Markov Model (HMM) based framework for hand gesture detection and recognition [4]. Bhattacharya et al. uses Support Vector Machine (SVM) and the Decision Tree (DT) on the Kinect sensors data stream [1].

In this paper we are interested in applying machine learning algorithms to classify real time poses belonging to a defined vocabulary, using the skeleton data provided by the Kinect sensor. We first extract features that represent the human skeleton from the kinect skeleton data, which are then used in the features vector to train machine learning algorithms. The parameters of the algorithms are adjusted over the cross-validation technique to maximize the classification performance.

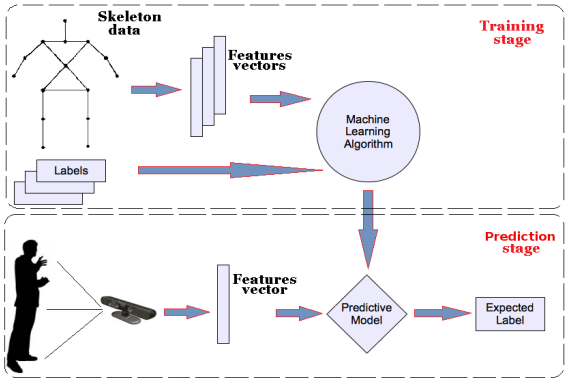


Figure 1: In the training stage the feature vectors are extracted from the skeleton data then given to the machine learning algorithm which construct the generation model used in the prediction stage to predict user's pose

2 The kinect sensor

The Microsoft Kinect is a device composed of multiple sensors (fig 2). An RGB camera, an IR emitter and IR depth sensor, a microphone array for speech recognition and a tilt motor to track the movements of the user. The most interesting are the IR emitter and IR depth sensor that distinguish the depth of objects in the field of view of the

RGB camera.

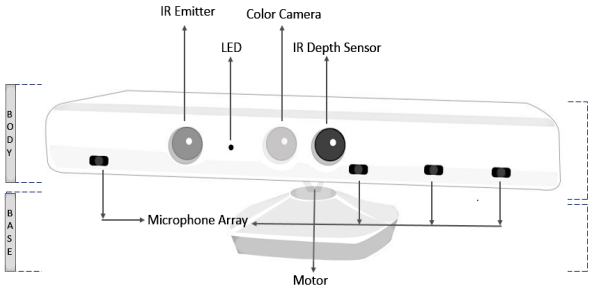


Figure 2: The kinect architecture

The OpenNI (Open Natural Interaction) framework is an abstract layer that can interfaces with different types of equipment and provides functions for developing applications using natural interactions.

OpenNI allows communication between real sensors (such as an RGB camera, camera/infrared transmitters, microphones ..) and "collectors" which are in fact the OpenNI modules that will treat the data acquired by the sensors and derive useful information for the developed application.

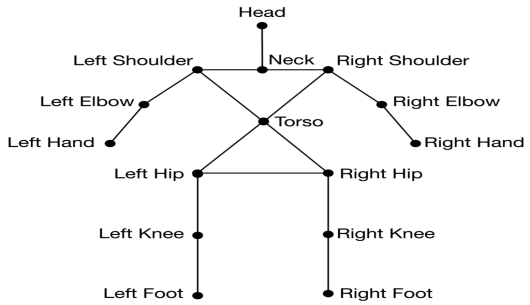


Figure 3: OpenNI's joints

The Kinect sensor sends skeletal data to the computer as a temporal sequence of coordinates $\mathbf{x_i}$, $\mathbf{y_i}$ and $\mathbf{z_i}$ (0<i<14) of all 15 tracked joints.

3 Mathematical model

3.1 Support Vector Machines

Support Vector Machines (SVM) is a supervised leaning, and its basic idea is to find an optimal hyperplane margin that properly separates data, while being the furthest away possible from all data. The hyperplane is defined by $f(x) = w^T \cdot x + b = 0$, and the margin by $min \frac{1}{2}\|w\|^2$, where w is the normal weight vector of the separating hyperplane, b is the bias term, and x is a vector of n features. The classification of a new individual $x^{(i)}$ is given by its position relative to the hyperplane. SVM is based on the use of kernel functions that allow optimal separation data. among the most frequently used kernels we find :

$$\textbf{Linear kernel} : k(x, x') = x^T \cdot x' \tag{1}$$

$$\textbf{Polynomial kernel} : k(x, x') = (\gamma x^T.x' + b)^d \tag{2}$$

$$\textbf{RBF kernel} : k(x, x') = e^{-\gamma||x-x'||^2} \tag{3}$$

Where d, γ and b are model parameters.

3.2 Artificial neural networks

Artificial neural networks (ANN) are inspired by the biological neurons. ANN are complex networks of interconnected elementary computation units that can handle binary or continuous values. ANN is composed of an input layer, an output layer and one or more hidden layers. Each layer of the network includes one or more neurons associated with the neurons of the previous and the next layer. The extracted values of the preceding layer are weighted by certain weights for each neuron, plus the bias term, then the output is calculated by an activation function f. There are several types of activation functions that can be used in ANN, the functions used in this work are :

$$\textbf{Sigmoid function} : f(x) = \beta \frac{1 - e^{-\alpha x}}{1 + e^{-\alpha x}} \tag{4}$$

$$\textbf{Gaussian function} : f(x) = \beta e^{-\alpha x.x} \tag{5}$$

Where α and β are model parameters.

3.3 The k-nearest neighbors

The k-nearest neighbors is one of the oldest and simplest methods of classification. The idea behind the KNN algorithm is quite simple, given a vector $x^{(i)}$ and a set N of m_N labeled neighbors, the task of the classifier is to predict the class label of $x^{(i)}$ based on the class labels of the set N using the majority vote.

In KNN the most important parameter is the number of neighbors k. The choice of k is essential in building the k-nearest neighbors model. So k can strongly influence the generalization performance. The value of k should be large enough to minimize the probability of error, but also reasonably low compared to m_N the size of the set N.

3.4 Bayes classifier

Bayesian classifier is a supervised learning and a statistical method based on Bayes theorem. Given a sample $x^{(i)}$ and a set of training samples S, each one with its class label C_l with $l \in [1, L]$ and L the number of classes, the classifier predicts that $x^{(i)}$ belongs to the class with the highest posterior probability:

$$P(C_l|x^{(i)}) = \frac{P(x^{(i)}|C_l)P(C_l)}{P(x^{(i)})}$$

Where $P(x^{(i)}|C_l)$ is the a posteriori probability of $x^{(i)}$ given C_l, $P(C_l)$ is the a priori probability of the classe C_l and $P(x^{(i)})$ the probability of the example $x^{(i)}$.

4 Method

The coordinates $\mathbf{x_i}$, $\mathbf{y_i}$ and $\mathbf{z_i}$ $(0 < i < 14)$, depend on the size and the position of the person in the scene. The approach used in this work is to use the coordinates of the joints, not relative to the image's origin, but relative to other joints, which will limit the dependence on the joints's position in the scene. For example the position of the hand relative to the elbow (above or below) etc. We then proposed the features vector to be as the following:

$$[\mathbf{y_2 - y_3}, \mathbf{y_2 - y_4}, \mathbf{y_3 - y_4}, \mathbf{z_2 - z_4}, \mathbf{x_2 - x_3},$$
$$\mathbf{x_2 - x_4}, \mathbf{x_3 - x_4}, \mathbf{y_5 - y_6}, \mathbf{y_5 - y_7}, \mathbf{y_6 - y_7},$$
$$\mathbf{z_5 - z_7}, \mathbf{x_5 - x_6}, \mathbf{x_5 - x_7}, \mathbf{x_6 - x_7}, \mathbf{x_4 - x_9},$$
$$\mathbf{x_7 - x_{12}}, \mathbf{y_4 - y_9}, \mathbf{y_7 - y_{12}}, \mathbf{y_9 - y_{10}}, \mathbf{y_{12} - y_{13}}]$$

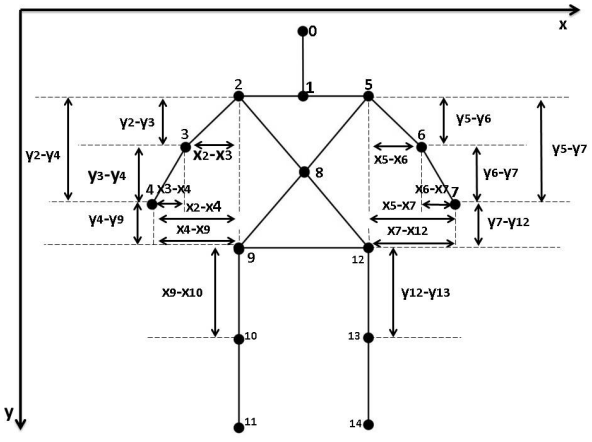


Figure 4: The chosen joints

We used a simple poses vocabulary composed of 18 poses, as shown in Table 1:

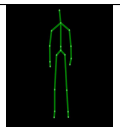
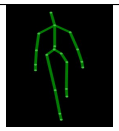
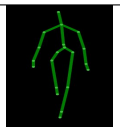
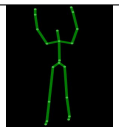
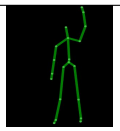
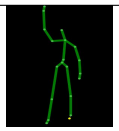
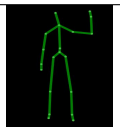
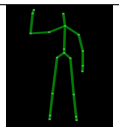
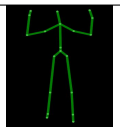
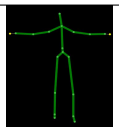
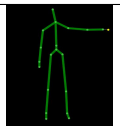
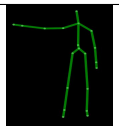
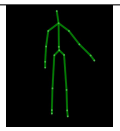
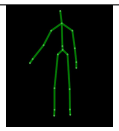
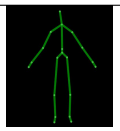
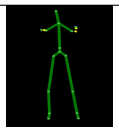
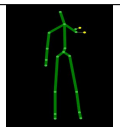
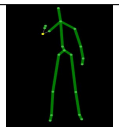
 Pause	 Right knee up
 Left knee up	 Both hands up
 Right hand up	 Left hand up
 Right hand 90	 Left hand 90
 Both hands 90	 Both hands side
 Right hand side	 Left hand side
 Right hand down	 Left hand down
 Both hands down	 Both hands forward
 Right hand forward	 Left knee forward

Table 1: The poses vocabulary

	SVM			ANN		KNN	NB
Training data	Linear	Polynomial	RBF	Sigmoid	Gaussian	-	-
22 %	99.9929 %	99.9786 %	99.9625 %	99.9833 %	99.9667 %	99.9429 %	89.0571 %
33 %	99.9917 %	99.9833 %	99.9929 %	99.9667 %	99.9667 %	99.975 %	99.6917 %
44 %	100 %	99.98 %	99.9833 %	100 %	100 %	99.97 %	99.72 %
55 %	100 %	99.9875 %	100 %	-	-	99.9875 %	99.675 %
66 %	100 %	100 %	100 %	-	-	100 %	99.6833 %
77 %	100 %	100 %	100 %	-	-	100 %	99.8 %
88 %	100 %	100 %	100 %	-	-	100 %	99.9 %

Table 2: Classifiers's performance

A subject performs a finite number of poses belonging to the defined vocabulary of 18 poses. We recorded 100 examples for each pose, making a total of 1800 examples.

One can imagine in a basic scenario the person in front of the camera will be able to train the algorithm with a minimum set of poses. Therefore, the main objective is to find the best parameters for each algorithm trained with different data fractions. For this reason we apply cross-validation technique. Concretely for each algorithm parameter, the data composed of m = 1800 samples is mixed. We use m_{train} random samples of data to train the algorithms, and remaining data (m_{test}) for testing in order to compute prediction's performance (the accuracy). This operation is repeated 10 times, and each time, the data is mixed again.

For each learning algorithm we choose the parameter value that give best performance in the cross-validation stage, then using those parameters, we construct a classification model, that will allow to make the poses classification.

In the prediction stage, a user performs poses in front of kinect. The algorithms process every frame of the kinect skeleton data, extract feature vectors, and apply the model constructed in the training stage, then the class of every frame is predicted and displayed in real time.

5 Results and Discussion

In SVM we worked with three kernels, each kernel has its own parameters. So we vary those parameters, and we compute the performance on the testing data (show Table 2). In the linear kernel there are two parameters to change, the number of training data and the hyperparameter C. The numerical results that we have obtained show that the minimum performance of the linear kernel is 99.05 %, with 88 % of the data used for training, and C = 10. While the maximum performance is 100 % for several values of C and training data. In the polynomial kernel there are four parameters to change, the number of training data, the hyperparameter C, the degree d and the parameter γ. The minimum performance obtained for the polynomial kernel is 5.65 % with 88 % of the data used for training, C = 100, γ = 0.1 and d=15. In general, the performance decreases when $d \geq 8$. Whilst the maximum performance is 100 %. In the RBF kernel, there are three parameters to change, the number of training data, the hyperparameter C and the parameter γ. The maximal performance is 100 %, and becomes minimal when $C < 100$, or $\gamma \geq 0.01$.

In ANN we can found many activation functions, here we worked with both the Gaussian and the sigmoid functions. The sigmoid activation function gives good results when $n_L \leq 4$, where n_L is the number of network layers, and β = {0, 0.9, 1}, and this, whatever the α value and the fraction of training data. The Gaussian activation function gives good results when $n_L \leq 4$, and $\beta = 0$ or $0.4 \leq \beta \leq 1$ with α = 0.4, regardless of the fraction of training data.

In the k-nearest neighbors algorithm the adjustable parameters are the number of nearest neighbors k, and the fraction of training data. The performance of KNN decreases depending on training data. Hence, when the training data fraction is less than 33 % the prediction error of the KNN algorithm starts to increase.

In the Bayesian classifier the parameter that we adjust is the fraction of training data. When the fraction of training data is greater than 22 %. The minimal value is 15.76 % and the maximal value is over 99.94 % recorded when the training data respectively representing 11 % to 72 % of all data.

Conclusions

Choosing the right model required several steps; collecting data, selection of the training set, and the choice of optimal parameters of each algorithm.

This study has shown that the SVM classifier provides best performance among the algorithms that we used. Its generalization performance reached 100 % for the dataset used in this work, and the other algorithms also give very good results. Other learning algorithms may be added and compared with those used in this study.

The application that we realized allows to classify user's poses using the kinect skeleton data, can classify the poses of any user whatever his size and his position in he scene, can be used to control a robot, a robotic arm, or in other applications of robotics, can be the basis of another application which classifies gestures by dividing gestures into multiple poses.

References

[1] Sambit Bhattacharya, Bogdan Czejdo, and Nicolas Perez. Gesture classification with machine learning using kinect sensor data. *Emerging Applications of Information Technology*, 2012.

[2] Abhijit Jana. *Kinect for Windows SDK Programming Guide*. Packt Publishing, 2012.

[3] Simon Lang. Sign language recognition with kinect. *Free University of Berlin*, 2011.

[4] Nianjun Liu and Brian C. Lovell. Gesture classification using hidden markov models and viterbi path counting. *VIIth Digital Image Computing: Techniques and Applications, Sydney*, 2013.

[5] Leandro Miranda, Thales Vieira, Dimas Martinez, Thomas Lewiner, Antonio W. Vieira, and Mario F. M. Campos. Real-time gesture recognition from depth data through key poses learning and decision forests. *Proceedings of the 2012 25th SIBGRAPI Conference on Graphics, Patterns and Images*, 2012.

[6] Edon Mustafa and Konstantinos Dimopoulos. Sign language recognition using kinect. 2014.