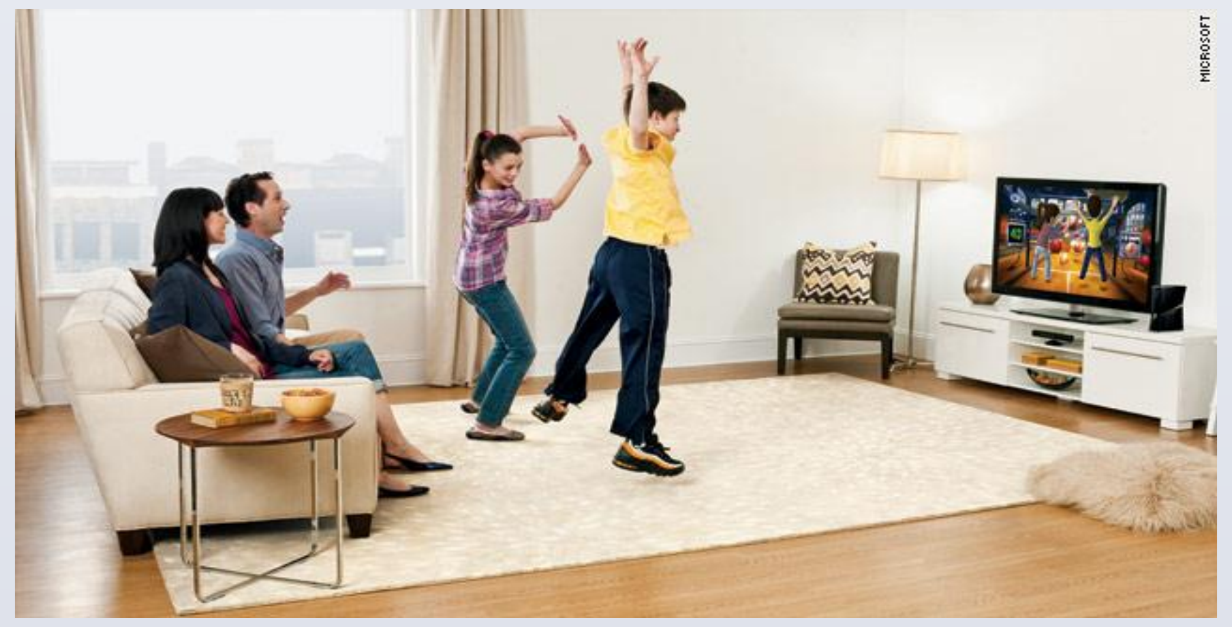
3D Action Recognition Using EigenJoints

Prashant Kumar|Kranthi Kumar
Supervisor: Dr. Amitabha Mukerjee
Dept. of Computer Science, IIT Kanpur

INTRODUCTION

Human action recognition is a very vast and very important area in artificial intelligence. This method of human action recognition can be adopted to solve most of the real world problems very efficiently. For instance, this can be used in areas like content-based video search, health-care and video surveillance. Traditionally human action recognition was accompanied by RGB cameras, which produce sequence of 2D frames in chronological order but the accuracy in action prediction by this set of data was not appreciable. With the development of RGBD cameras such as Microsoft Kinect and ASUS Xtion Pro and action recognition can be done efficiently. These depth cameras were used to calculate the depth information of each frame in addition to the data which was generated by simple RGB cameras, which added to the accuracy of action recognition.



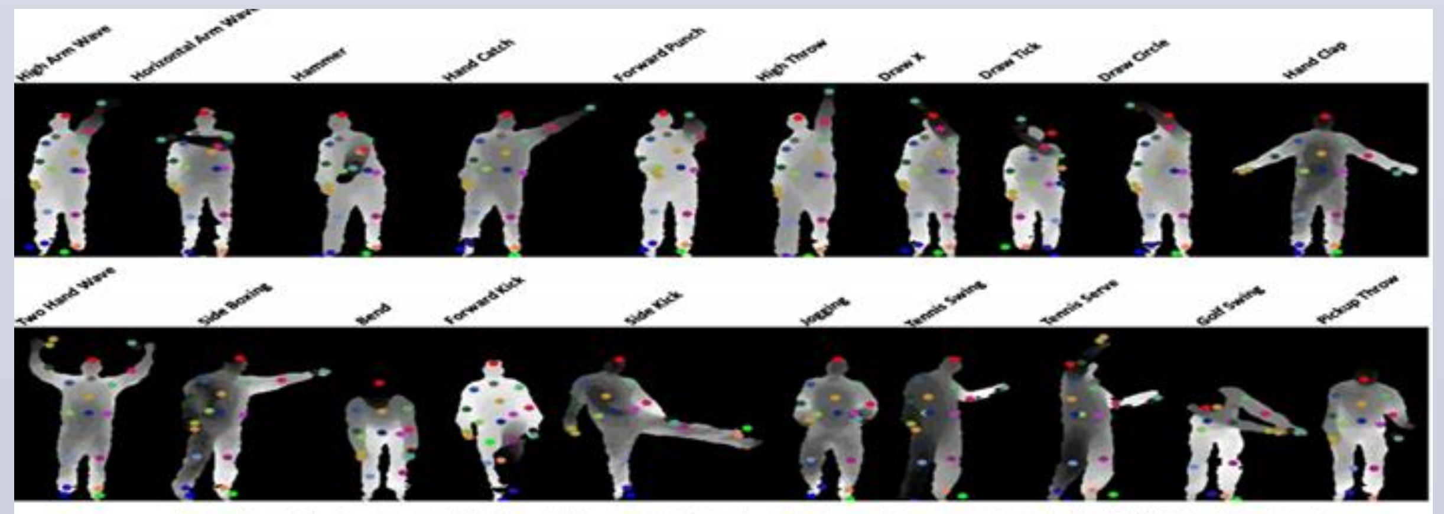
DATASET

We have used MSR Action3D Data Set.

It consists of :

- 20 actions
- each action performed by 10 persons
- each person performed action for 3 times
- each frame has 20 joints

Apart from this we are also generating our own test data, using Microsoft Kinect and recognize the action from MSR3D dataset.



Examples of depth maps and skeleton joints associated with each frame of twenty actions in the MSR Action3D dataset.

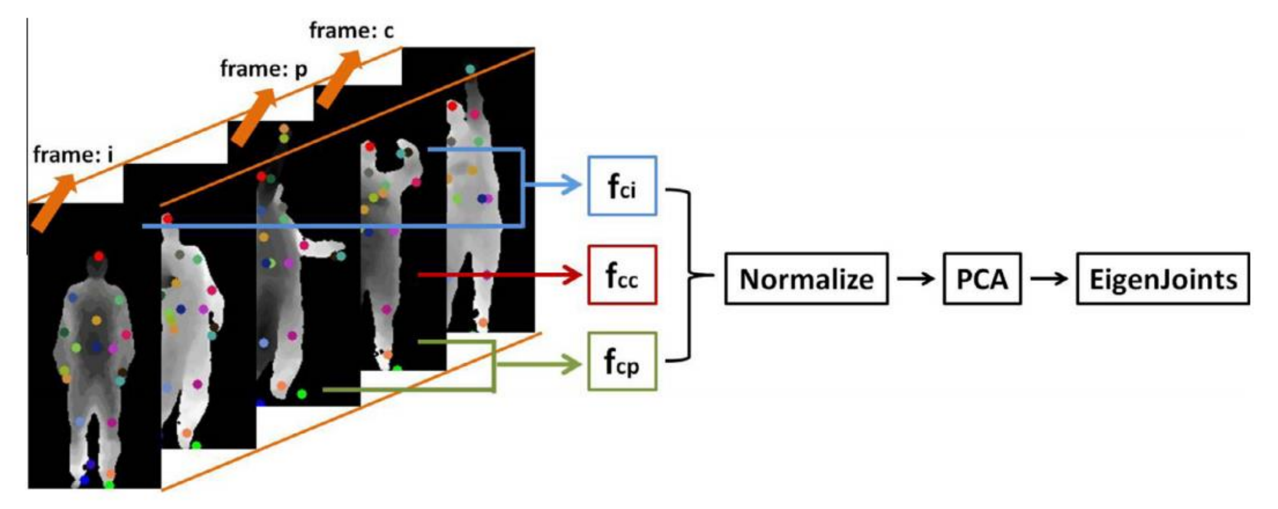
METHODOLOGY

- We have the 3D coordinates of N joints in each frame calculated using the depth map of an action.

$$X = \{\, x1,\ x2, \ldots\ldots, xN\}$$

Each xi is a 3D vector containing the x, y and z coordinates of joint i.

- The approach basically involves the two steps:
  - Feature Extraction : It involves eigenjoint calculation of each frame in an action.
  - Non-parametric classification: Naïve Bayes Nearest Neighbour classification is employed



The framework of representing EigenJoints. In each frame, we compute three feature channels of f_{ci}, f_{cc}, and f_{cp} to capture the information of offset, posture, and motion. The normalization and PCA are then applied to obtain EigenJoints descriptor for each frame.

- Feature extraction involves creating of three feature sets calculated using 3D position differences of skeleton joints. These feature sets are then merged into one big feature. We normalize the new feature and do a PCA on that to get the final representation called the 'EigenJoint'. The three feature sets mentioned above are:
  - Static posture feature (f_{cc}) : We calculate the pairwise joint differences between the current frame ‘c’.

$$f_{cc} = \{x_i - x_j | i,\ j = 1, 2, \ldots, N;\ i \neq j\}$$

  - Consecutive motion feature (f_{cp}) : The motion feature is represented by the pairwise joint differences between the current frame ‘c’ and the previous frame ‘p’.

$$f_{cp} = \{x_i^c - x_j^p | x_i^c \in X_c; x_j^p \in X_p\}$$

  - Overall dynamics feature (f_{ci}) : It captures the overall dynamics of the frame. It involves calculating the pairwise joint differences between the current frame ‘c’ and the first frame ‘i’ of the action in consideration.

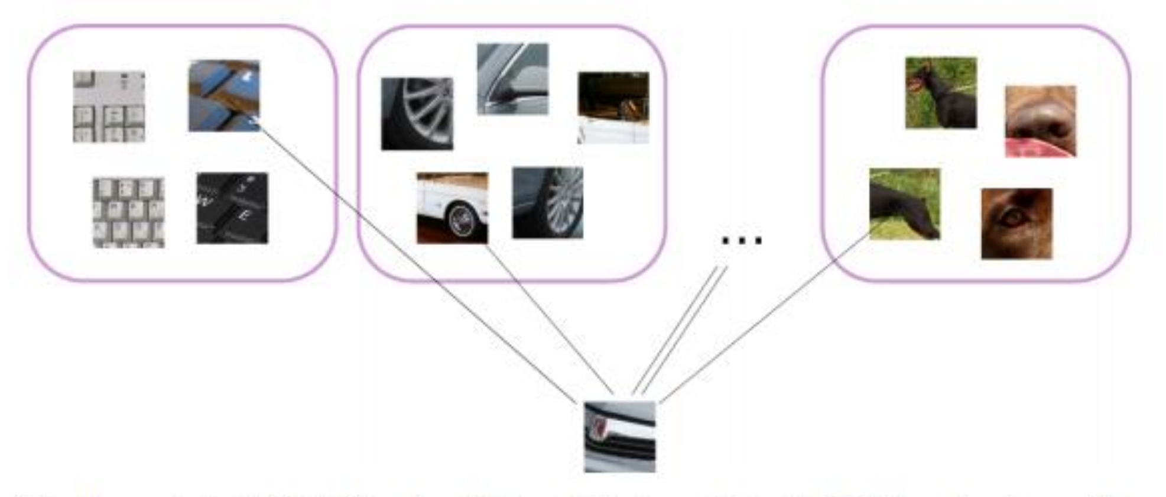
$$f_{ci} = \{x_i^c - x_j^i | x_i^c \in X_c; x_j^i \in X_i\}$$

$$f_c = [f_{cc}, f_{cp}, f_{ci}]$$

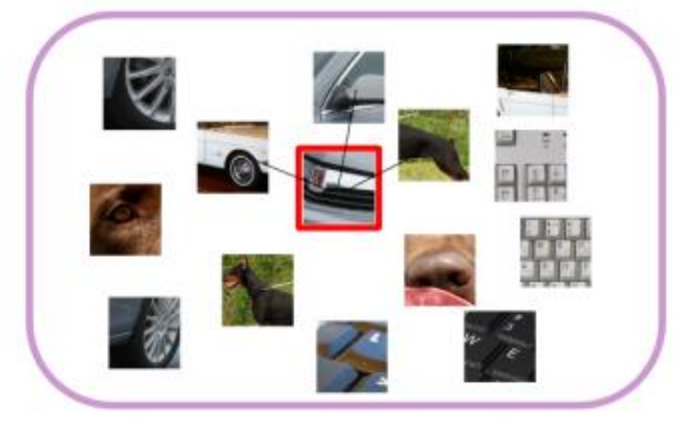
- Naïve Bayes Nearest Neighbour Classifier: The classification is done using unsupervised learning. Most classifiers quantize the local feature descriptors thereby decreasing the discriminating ability of the data. Unlike these classifiers NBNN classifier does not quantize and hence retains most of the information. In addition it computes video-to-class distance rather than video-to-video distance which other classifiers do.

$$C^* = \underset{c}{\operatorname{argmin}} \sum_{i=1}^{M} \|d_i - NN_c(d_i)\|^2$$

d_i is the EigenJoint descriptor of frame ‘i’ in the testing action video sequence. ‘M’ is the number of frames in the sequence and $NN_c(d_i)$ is the nearest neighbour of d_i in class ‘C’ .



(a) The original NBNN asks, “Does this descriptor look like a keyboard? a car? ... a dog?”



(b) Local NBNN asks, “What does this descriptor look like?”

CONCLUSIONS

NBNN classification requires no learning/training phase and its performance is comparable to other supervised learning methods like SVM.

REFRENCES

[1]Xiaodong Yang and YingLi Tian. Effective 3D Action Recognition Using

[2]Oren Boiman , Rehovot, Eli Shechtman ,Michal Irani , In Defense of Nearest-Neighbor Based Image Classification

http://research.microsoft.com/en-us/um/people/zliu/ActionRecoRsrc/