Artificial Intelligence Project:
3D Action Recognition Using EigenJoints

KRANTHI KUMAR — PRASHANT KUMAR
Supervisor: Dr. Amitabha Mukerjee

April 24, 2014

1 Abstract

The motive of this project is to recognize human actions from the joint coordinates obtained from RGBD cameras. We have followed the algorithm proposed by 'Xiaodong Yang and YingLi Tian' in the paper 'Effective 3D Action Recognition Using EigenJoints.'

2 Introduction

Activity Recognition: It is used to recognize the action performed by a person by encapsulating data (here movement of joints of a person) from the set of frames of the action.

From the beginning of 1980s several computer science communities had started working on this issue as they thought this would bring a great change in the human life and can be very useful in many of the sectors in society such as health care, content-based video search, human-computer Interaction, etc..

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 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 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.

3 Related Work

Reduction of complex computations for action classification and the accuracy of the action predictions are very obvious tasks of every researcher in this area. Any improvement in this area will be a great achievement, as machines can be interacted with humans very easily. Earlier action recognition was done with the help of 2D video sequences captured by RGB cameras. Trajectory based methods were also used for action recognition. Extracting joint information from these 2D video sequences is difficult and requires a lot of effort. But with the development of RGBD cameras we are able to capture the depth in the sequence which even allows us to distinguish between actions that have similar 2D projections.

4 Dataset

We have used Microsoft Action 3D dataset in this project.
It consists of 20 actions such as kicking, boxing, tennis serve etc..
Each action is performed by 20 persons, and a person performed each action 3 times.
Each action has 30-70 frames and each frame has 20 joint co-ordinates of the person performing the action.

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

5 Implementation

We are following the algorithm proposed by X.Yang et.al. [1]. The approach involves two steps:

- Feature Extraction - Calculation of EigenJoints.
- Non-Parametric Classification - Classification using Naive Bayes Nearest Neighbour algorithm (NBNN).

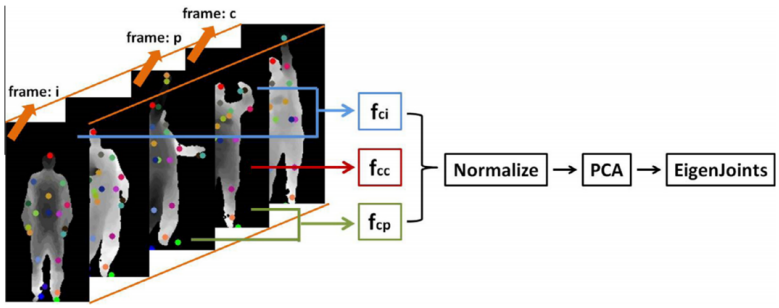
5.1 Feature Extraction

We already have 3D coordinates of the N joints in each frame available from the MSR Action3D dataset. So for each frame we have the following representation:

$$X = \{x_1, x_2, ..., x_N\}$$

Each x_i is a vector of x, y and z coordinates of joint i.
Feature extraction involves calculation of eigenjoint feature for every frame of an action and this is computed for every action.

The following steps are involved in the calculation of eigenjoint features for an action:



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.

For every frame 'c' of the action we compute the following:

- **Static Posture Feature** (f_{cc}) - It stores the pair-wise joint coordinate differences within 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 pair-wise 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 action by calculating the pair-wise joint differences between the current frame('c') and the initial frame('i')

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

Now, we concatenate them to get the feature vector $f_c = [f_{cc} f_{cp} f_{ci}]$. Normalization of f_c is done to ensure that elements with larger magnitude doesn't dominate elements with smaller magnitude.
Having said that, we see that the dimension of f_c is very large. f_{cc} has $N(N-1)/2$ elements, f_{cp} has N^2 elements and f_{ci} has N^2 elements. And each element is a vector of 3D coordinates. So if we have 20 joints i.e. N is 20, overall dimension of f_c will come out to be $3 * 990$ or 2970. So we reduce its dimension using PCA and keep only the top 128 eigenvectors which captures around 95% of the dataset[1].

5.2 Classification

After the feature extraction phase, non-parametric unsupervised learning is employed for classification. For this the author has used NBNN classifier. Unlike other classifiers NBNN does not quantize the descriptors so that the descriptors are in their original form.
Quantization of descriptors involves dimensionality reduction which results in decreased discriminative power of the descriptors. NBNN avoids this. In addition, it is different from other classifiers in the sense that it computes video to class distance rather than video to video distance as is done in other NN classifiers[2]. NBNN algorithm is given below:

Algorithm 1 NBNN(Q)

Require: A nearest neighbor index for each C, queried using $\mathrm{NN}_C()$.
Require: A query action Q, with descriptors d_i.

```
for all descriptors d_i ∈ Q do
  for all classes C do
    totals[C] ← totals[C] + ||d_i − NN_C(d_i)||^2
  end for
end for
return argmin_C totals[C]
```

Where Q is the the test action with d_i as the eigenjoint descriptor of frame i. $NN_C(d_i)$ is the nearest neighbour of d_i in class C.

6 Conclusion

We have successfully implemented the algorithm as specified in the above paper. We have used first two trails of each person corresponding to each action for training purposes and the third one for testing purpose.

7 Acknowledgement

We thank Prof. Amitabha Mukerjee, Dept. of Computer Science and Engineering for his valuable support throughout the project guiding us from time to time and looking into the project when it was needed. We also thank python community for their user friendly libraries.

References

[1] Effective 3D Action Recognition Using EigenJoints, X. Yang, Y.Tian, http://www-ee.ccny.cuny.edu/wwwn/yltian/Publications/JVCIR_EigenJoints.pdf

[2] In Defense of Nearest-Neighbor Based Image Classication, O. Boiman, E . Shechtman, M. Irani , http://www.wisdom.weizmann.ac.il/~irani/PAPERS/InDefenceOfNN_CVPR08.pdf