CV-BASED TENNIS COACHING

Prepared for: Exam of ‘AI Lab: Computer Vision and NLP’ by Professor Pannone

Prepared by: Federica Bruni (1933963) , Maria Emilia Russo (1966203)

TABLE OF CONTENTS

1. INTRODUCTION

1.1 - Motivation

The idea for this project came from the fact that we are great tennis fans and we would love to have Federer's backhand or Zverev's serve but we are very far from there.
People say that by watching tennis matches your game improves but it is not that easy. Tennis requires time and a lot of patience and get tennis lessons is not so easy: they need good weather, your coach must be available, you must have money to spend (tennis is as beautiful as expensive), a court must be available etc.
So our idea instead is: why don't we use computer vision to improve our tennis game?
This is how everything started.

1.2 - Goal

Our main goal is by using a single camera to analyse our tennis game.

So we now fix our goals:

1. We want to build a model able to predict (TODO)
2. We want to be able to compare our shots with PRO's ones and get a score out of how close we are to their level and see our movement comparison.

1.3 - Assumptions

In our project we're doing two big assumptions:

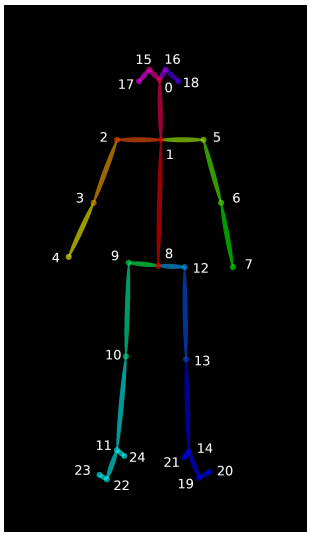
- we're considering only right-handed players
- we're considering only videos with a back view of the analysed player

2. DESIGN

2.1 - Pose estimation

Since our main goal is to able to predict strokes. In order to do that we need to identify strokes and this cannot be done without estimating the position of the tennis player first. So this is going to be our first step. Pose estimation is a rather complex world and it is not the focus of our project this is why we chose an open source code to implement the pose estimation task.

Notice that our initial attempt was to be able to identify actions from the player side, not the back side.
But we soon found out how difficult that is because it is much more complex to identify different body parts from the side view where many of them overlap rather than from the back where the situation is much clearer. Also consider that tennis is a very fast sport, everything moves within one blink so it would have been much more complex with the side view.



We now present our choice: 'OpenPose' which seems to be the state of the art within human pose estimation.
We used video as a means to recognise body movements, focusing on each frame of a given sequence. In order to get a detailed interpretation of people in an image or a video, their body parts must be identified by some key points. Key points, in this case, refer to body parts.

Openpose has also another big advantage: allows us to train on keypoints and not on images.
Training on images is costly and time consuming and also keeps many more informations than needed.
Training on body keypoints instead allows us to be more efficient and more focused on what really matters: body movements.

2.2 - Action recognition

What does identify a stroke?
First of all we considered 3 types of strokes: Forehand, Backhand and Service.
Our first attempt was to identify a stroke by the relative position of the ball wrt the player at the hit.
If the ball was over the player's head it is a serve, if it is on the right of the player at hit frame than it is a forehand otherwise it is a backhand.
But detecting a still tennis ball has nothing to do with detecting a tennis ball in a tennis match.

On the left: tennis ball from google

On the right: tennis ball from our dataset

They don't have neither the same color not the same shape. We need to train our own model for this.
So we trained our database to detect those kind of tennis balls but it was a loss of time. No matter how many trained samples we trained still it was really hard to get good results. Because it is very fast and it has no particular shape or color. It was a mess. Time to change our plans.

Our second attempt was to identify a stroke based on the key-points of our tennis player body. To classify and train our model we adopted the SVM approach. We used the SVM to find the maximised margin to fit the data.

We tried 3 different kernels:

KERNEL	ACCURACY
RBF	75%
SIGMOID	57%
LINEAR	95%

and we ended up using the linear one.

2.3 - Action Scoring

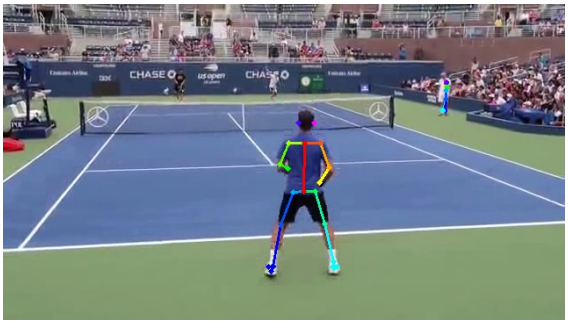
Goal: compare a common person stroke to a PRO one and return a score out of 100 on how close his stroke is to the PRO one.
First of all we need to normalize data because these people could have different height and be in different parts of the video. In order to account for the size inconsistencies, we perform L2 normalization of the points in order to transform it into a unit vector.
After normalising each Keypoint we needed a similarity quantity to find a distance between PRO and common person. But which?
Our idea is that we have 25 key-points for each video and their value change with time.
Which means that a movement can be seen as a time series.
Then we could use DTW to compare the two graphs and get 25 scores for each of the 25 key-points. A n average of these 17 scores could be then taken as the total score.

3. IMPLEMENTATION AND COMMENTS

3.1 - Openpose

For pose estimation we adopted Openpose.
Openpose is an open source tool authored by CMU Perceptual Computing Lab, and it is the first real-time multi-person system to jointly detect human body, hand, face and foot keypoints.
It takes as input a .avi file and outputs frames, video and json files that record keypoints.
IN:

OUT:



In order to use Openpose we needed to build a system of directories which could be used by OpenPose to save its output.
In order to do so we first implemented the set_up() function which setups the two main directories: Train and Test in which then the user could insert its videos.
Then the populate() function was created. This function checks in the train folder if all files have been already processed with Openpose, if not it runs open pose over that file. But before doing so it needs to build the output folder and the 3 subdirectories: video, images, json.
This was all for the pre-processing phase.

Than the next step was to build a train data frame that keeps all key-points for each frame in each train video.

2.2 - Action recognition

The first step for the action recognition section is to manually label the train videos. This step could have been a nightmare, assigning each frame with a label number. We insted make it efficient taking two assumptions:

1) all training files are saved with shot name + _ + number

2) we assume that each training file represents only one stroke type

(hence it can give that name to the file)

so our implementation directly taking the name from the filename adds the corresponding label to all file's frames.

Then we merged this data frame with the one obtained by the Pose estimation section.

At this point we have a datagram which for each frame holds: label, file number, frame number, key-points (x and y) .

Then we will use Scikit-Learn's support vector classifier to train an SVM model on our data. Then we train the model using 80-20 splitting of train-test data and a linear kernel (look at design section 2.2 to know more about kernel selection).

Then we do a similar process as we did in section 2.1 for the test data.

Then we use the trained model to predict the actions in the test data.

We also wanted to display such predictions so we built a utility function that displays for each frame the predicted action.

OUTPUT:

3.3 - Action scoring

In order to score our videos we need a different setting: we need to have the x and y coordinate of key-points in a matrix for each video.
So we built a function that given an input video returns the matrix of key-points and the number of rows in the matrix (we don't need to know the number of columns since they are fixed, 2 x and y).
We then used a package called dtaidistance, which was used to compute DTW scores between two sequences when given in the form of Numpy arrays.
In this way, we were able to use the temporal information gained from pose estimation at each frame in order to compare the actions performed by two people.