A TRACING-BASED TENNIS COACHING AND SMART TRAINING PLATFORM USING ARTIFICIAL INTELLIGENCE AND COMPUTER VISION

Feihong Liu[1], Yu Sun[2]

[1]Crean Lutheran High School, 12500 Sand Canyon Ave, Irvine, CA 92618
[2]California State Polytechnic University,
Pomona, CA, 91768, Irvine, CA 92620

ABSTRACT

Athletes in technical sports often find it difficult to analyze their own technique while they're playing [1]. Often, athletes look at the technique of professional players to identify problems they may have. Unfortunately, many types of techniques, such as forehand and backhand swings in tennis, are relatively similar between a beginner and a professional, making it more difficult for comparison. On the other hand, techniques that appear different between professionals and casual can also present different challenges. This is especially true for serves in tennis, where the speed of the swing, the motion of the player, and the angle of the camera recording the player all pose a challenge in analyzing differences between professional and learning tennis players [2]. In this paper, we used two machine learning approaches to compare the serves of two players. In addition, we also developed a website that utilizes these approaches to allow for convenient access and a better experience. We found that our algorithm is effective for comparing two serves of different speeds and synchronized the videos effectively.

KEYWORDS

Pose-estimation, Machine Learning, Scikit-learn.

1. INTRODUCTION

In recent years, Human pose estimation (HPE) has garnered popular attention due to the wide range of its applications in gaming, video surveillance, sign language, etc [3]. Human pose estimation (HPE) is a computer vision technique that aims to accurately recognize the posture of people in its input data (e.g., images, videos, or signals) [4]. HPE accomplishes this through two main steps: localizing anatomical joints, and grouping the joints into a valid configuration [5]. Of course, human pose estimation is complicated by several factors including varying visibility of joints, lighting, and cluttered backgrounds [6]. In fact, these factors challenge even state-of-the-art methods such as Facebook's AI research Detectron2 [7]. Faced with these challenges, many researchers in recent years have started using data taken from multiple dimensions to increase the accuracy of HPE. However, 3D data, although more accurate, was usually much larger than 2D data and took more time to process. Run-time becomes much more important in data with multiple targets. In these cases, either the top-down or bottom-up framework is generally used.

Thanks to its abilities, HPE could be very useful in analyzing individual sports such as tennis, where targets are limited, and motions are predictable. For many players, human pose estimation

David C. Wyld et al. (Eds): AIFZ, SIGPRO, NWCOM, CLSB, MLNLP, DTMN, BDIOT, CSITY - 2022
pp. 89-100, 2022. CS & IT - CSCP 2022 DOI: 10.5121/csit.2022.121608

can offer a better way to compare two players' techniques. For techniques such as the serve, it can be difficult to analyze the specific differences because many body parts are moving together. To the observer, the movement of the torso can be misinterpreted as a movement of the arms. Timing can also be an issue, as players tend to serve at varying speeds throughout the serve (having a slow toss of the ball but having a fast swing). Pose estimation offers a way to compare, with more specificity, the techniques of two players.

Improvements in tennis serves are mainly achieved through coaching, either through digital advice, or an in-person coach. Generally, several things are advised for improving a serve: stance, toss, racquet drop, contact spot, and finish [8]. While this seems straightforward, for many casual tennis players, identifying their own issues is the most difficult step. Self-monitored practice is often inaccurate as players themselves misjudge their flaws. Having an in-person coach offers an advantage because coaches are able to make better observations and provide feedback on players' techniques. In addition, coaches are shown to adopt an external focus of attention which can stimulate growth in players [9]. These methods lack a combination of convenience and accuracy. While the advice is straightforward, in-person coaching lacks convenience while self-monitored practice lacks accuracy.

In addition to these methods, other methods also exist to help growth in tennis serves. SwingVision, an editor's choice app backed by various professionals, uses a camera to keep records of tennis points. The app utilizes an iPhone to record points, and detects the types of swing used and other related measurements such as ball speed and the location where it landed. SwingVision is able to detect whether a player used a forehand groundstroke, a slice or an overhead, and even detects whether the player added spin to the ball. This allows for users to see where they can improve on in terms of strategy, and help them learn from their points. Although SwingVision is very useful for reviewing match play, it does not offer ways of improving technique. It gives a way to review the techniques used during a game, but doesn't provide insight into improvement of those techniques. Other services such as TennisGate improves accessibility by making a platform that connects players with online coaches. TennisGate also posts many articles and tutorials on improving technique in tennis. Although this improves on in-person learning by making coaching more convenient, players can often find online coaching less effective and less personalized.

In this paper, we implemented pose estimation to compare two videos of tennis serves. We wanted to make an algorithm that adjusts for different racquet speeds, so a casual tennis player can compare their relatively slow serve with a professional with a faster swing. We also identified keyframes in which players can compare their form with another player. Even with videos that start at different times, this algorithm can identify when the serve starts and edit the video accordingly. We approached doing this through two methods: supervised and unsupervised learning. For both methods, we are using MediaPipe to do pose estimation, which can generate 2d and 3d predictions. For the supervised learning method, we are using various sklearn classification models including Nearest Centroid, Nearest Neighbors analysis and Multiclass SVM. For the unsupervised learning method, we are using SkLearn's k-means clustering to group frames to account for differing speeds [10]

Compared to other approaches, this approach is much more specific. Being specialized in the tennis serve only, this algorithm can give more specific advice on improvements and allow players to form their opinions with a better comparing tool. This algorithm breaks apart the video of tennis players, giving a clear comparison between key moments of their swing and another player's swing and allowing players to compare with more clarity.

The rest of the paper is organized as follows: Section 2 presents related works in our subject; Section 3 gives the details on the challenges that we met during the experiment and designing the sample; Section 4 focuses on the details of our solutions corresponding to the challenges that we mentioned in Section 3; Section 5 presents the relevant details about the experiment we did. Finally, Section 6 gives concluding remarks, as well as pointing out the future work of this project.

2. RELATED WORK

Various past research papers discuss similar topics that will occur in this paper. In this section, we will briefly talk about some existing papers published in our subject.

One of the more recent papers, a paper titled *Classification of Tennis Shots with a Neural Network Approach* was published in 2021. This research paper discusses the use of Neural Networks to classify tennis shots [11]. This paper uses data collected from accelerometers, gyroscopes, magnetometers and audio signals to classify tennis shots in five categories: forehand topspin, forehand slice, backhand topspin, backhand slice, and serve. Data is passed into a FCN and a ResNet, and their performance is compared. Our research mainly differs from their research because we are solely focused on classifying various parts of a serve. Our method for obtaining data is purely visual while theirs is biological. In addition, our research is useful in comparison of two serves while their research is useful in the classification of a tennis shot.

In another research paper published by the Lublin University of Technology, human movement analysis was implemented in tennis with a Graph Convolutional Networks approach. This research paper approaches classification of tennis shots through Spatial-Temporal Graph Neural Networks [12]. Data is collected through the Vicon motion capture system, and stored as 3d coordinates and inputted into the neural network to classify between forehand shots, backhands shots and no shots. Our research differs from this research paper as we focus on various parts of the serve while this paper focuses on different swings entirely. This paper also explores the performance of Spatial-Temporal Graph Neural Networks while we explored the different performances of various supervised and unsupervised learning models. In comparison, our research is more applicable and requires little special setup.

Finally, deep learning methods were applied to action recognition for tennis in a research paper titled *Deep Learning for Domain-Specific Action Recognition in Tennis*. This research paper focuses on the performance of certain neural network architectures in application for sports [13]. Visual data from tennis players are collected and features are extracted through a neural network. These features are then passed into another network which is used to classify various tennis actions. This paper also seeks to offer a method to achieve good results for the THETIS dataset, which is composed of low-resolution images of tennis actions. Our research differs from theirs because we seek to synchronize and identify specific parts of tennis serves while theirs seeks to classify different tennis actions. Their paper demonstrated that different neural networks can produce accurate and interpretable results and can potentially be applied to other sports. Our approach provides several methods that a tennis serve can be classified and compares those methods.

3. CHALLENGES

In order to build the project, a few challenges have been identified as follows.

3.1. Adjusting for Varying Starting Points and Speeds for Videos

When comparing two videos, the starting time and speed of the serve may vary. While player 1 can start their swing at the very start of the video, player 2 can start their swing a minute in. In addition, player 1 can also swing a bit slow in the beginning but accelerate their motion while player 2 can maintain a constant speed throughout their serve. This makes comparison of the two videos very complex, as the algorithm ultimately needs to adjust for the speeds and starting points of each player. Initially, we proposed to directly train a neural network to detect when each player is in key stages of a serve. We would train a model with datasets for different stages such as the stance, the racquet drop, and the contact point. However, there were very limited datasets, and would make the algorithm very inefficient. Ultimately, we decided to utilize clustering to adjust for these differences. We would implement clustering first within each video and group frames that are more similar based on the pose of the player. We would then match these clusters from each video with each other to match them up. Doing so, we will generate an array for frames that we have to match up between the videos. Thus, we can then adjust the speeds of the videos between these keyframes to match up key points of the serve and synchronize the two videos.

3.2. Maximizing Accuracy

A common challenge in pose estimation is the different perspectives used for the camera. If two videos of players are taken at a similar perspective, the algorithm performs much better. If videos of players are taken at different perspectives, the algorithm decreases in accuracy. To account for the variety of camera angles, we decided to use an algorithm that uses 3d data to generate results that have relatively little reliance on camera angle. To do this, we took the 3d coordinates of the predicted landmarks and used them to calculate angles between different body parts. We used these angles as data points for both the supervised and unsupervised learning approaches. In addition, we also implemented data augmentation on images for the supervised learning approach to simulate changes in camera angles in real life.

3.3. Data

To classify different stages of a server, a lot of images are required to train an image classification algorithm. However, datasets for images of tennis serves are very scarce. Datasets for tennis mostly concentrated around the historic record for ATP players. Ultimately, we decided that it would be better to create our own dataset. While looking for resources for tennis serves, we found that videos on youtube were generally a good source of tennis serves. We made a program that downloads youtube videos using its url and goes through each frame, allowing the user to manually label each image. We divided the tennis serve into six portions, from the preparation of the serve to the contact point and the finish. To make this process faster, we implemented an algorithm that lets the user look at a specific range of time in a video. After labeling about 20 videos, we got around 400 images for each label. Since a lot of these were quite similar, we manually looked through these images and deleted similar images. We eventually ended up with about 100 images per label. This number is vastly less than what is normally needed for a good algorithm. In addition, these images will undergo a series of algorithms that can eliminate a lot of these images. Thus, we used an algorithm to randomly alter certain features of the image. These augmentations include rotation, skew, zoom and reflection. We generated images with different combinations of these augmentations, and ended up at around 1300 images per label. This allowed the accuracy of our supervised learning models to increase greatly, and also allowed leniency in the input data.

4. SOLUTION

Our algorithm is built in python using the mediapipe and sklearn libraries. To make our algorithm more accessible, we integrated this algorithm using Flask. When certain inputs are given by a user on the website, an HTTP response will be forwarded to the controller, which will allow our algorithm, written in python, to process the data inputted and return values back to the website. Our algorithm does this in several ways, mainly through the two approaches of unsupervised classification and supervised classification.

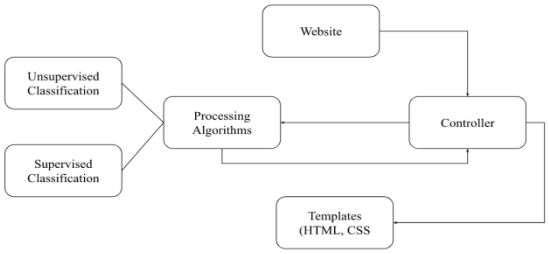

Figure 1. Overview of our solution

4.1. Unsupervised Classification

After a video is uploaded and forwarded to the controller, our algorithm will go through the frames of the videos, identifying a skeleton of the person in the video. Using the angles of the person's joints, we used sklearn's nearest neighbors clustering algorithm to identify five groups of frames within this video. These groups will represent different parts of the serve. After the initial pose-estimation algorithm, an array is generated with elements representing the labels for each frame for each video.

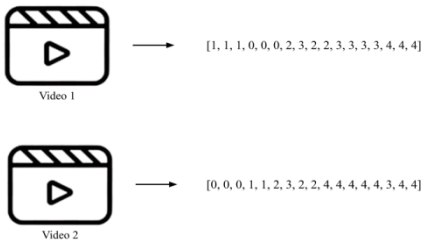

Figure 2. Labels arrays generated for video

However, these generated frames are not guaranteed to match, so we first implemented an algorithm that can match the groups between the two videos.

```python
# Separate the frames processed by the labels from K-means
def process_result_labels(results_labels):
    array = results_labels.tolist()
    return_array = [(0, 0)]
    # loop through array and return ranges for each label
    for i in range(1, len(array)):
        if array[i] != array[i - 1]:
            # if the length of this range is too small
            if i - return_array[-1][1] + 1 <= 2:
                return_array[-1] = (return_array[-1][0], i)
            # general case
            else:
                return_array.append((return_array[-1][1] + 1, i))
    return_array.append((return_array[-1][1] + 1, len(array)))
    return return_array[1:]
```

Figure 3. Screenshot of code 1

First, the array for video 1 is processed using the algorithm above. A new array is created containing the ranges of the occurrence for the labels. For example, if the input array was [0, 0, 0, 1, 1, 1, 2, 2, 3, 3, 4], the output array will be [(0, 2), (3, 5), (6, 7), (8, 9), (10, 10)] because the 0 occurred at indexes 0 - 2, etc. For simplicity, we added an additional if statement that limits the length of these ranges. If the range is less than 2, then the interval is considered to be too small, and this interval is instead combined with the previous one. In the previous example, instead of outputting [(0, 2), (3, 5), (6, 7), (8, 9), (10, 10)], the algorithm will consider the (6, 7) too small, combining it with (3, 5) to be (3, 7), etc.

```python
def calculate_distance(angles, frame, labels, prediction_label):
    # take the frame being predicted, along with its matching number.
    min_value = [0, 0]
    test_frame = np.array(frame)
    # return the indexes where the predicted label occur in vid2 (use [0] because it returns list, and then dtype)
    indexes = np.where(prediction_label == labels)[0]
    for i in indexes:
        current_angles = np.array(angles[i])
        score = LA.norm(current_angles - test_frame)
        if min_value == [0, 0] or score < min_value[1]:
            min_value = [i + 1, score]
    # return the frame with the least value (most matched)
    return min_value[0]
```

Figure 4. Screenshot of code 2

Using the midpoints of the ranges of the previous arrays, frames of each range in video 1 is compared with frames in video 2. Frames of video 1 are predicted using the model generated from video 2, and then compared with the angles of the frames of video 2 to match certain frames. This process is roughly illustrated in the Figure below.

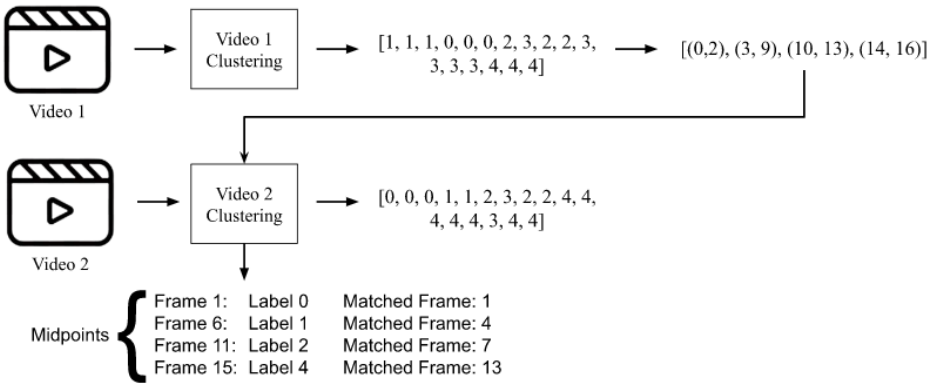

Figure 5. General illustration of our algorithm

In some cases, however, these matched frames might decrease. Thus, we made an algorithm that selects the longest increasing groups. Afterwards, points where labels change are marked as critical frames, and annotated and saved in a folder. Using the matched frames, we adjusted the speed linearly of the two videos so that the critical frames match in time.

4.2. Supervised Classification

The initial process of the supervised classification approach is similar to the unsupervised approach. Once a user inputs a video, the algorithm will extract angle data from the frames of the video using Mediapipe. However, for supervised classification, a model will be trained using manually collected and labeled images and the angles from the inputted videos will be classified using the trained model.

To train our model, we initially had three approaches: raw images, 3d models or just the angles. Because of the different perspectives for images, the results of the first approach were very inaccurate, while the second approach was impractical as well. We ultimately decided to get the angles for the training images and use those to train the model. We used Nearest Centroid, Nearest Component analysis, Nearest Neighbors and Multiclass SVM as classification models. After a series of experiments to test for the optimal augmentations to apply to images (see experiment 1 and 2), we are ready to implement the model to synchronize videos.

After the angles for the frames of each video are predicted using the trained model, a list is generated with each element representing the predicted label for each frame. However, processing this list proved to be quite different from our approach for unsupervised learning. The main difference is that, for supervised learning, it makes sense only if the predicted labels to be increasing while the unsupervised learning approach might not group frames based on the progression of the serve. We implemented a series of processing algorithms that essentially looks for the longest subsequence where the number of increasing labels are at a maximum (i.e. if a subsequence of length 100 with only label 1 was compared with a subsequence of length 40 but had increasing labels from 1-4, our algorithm would prioritize the latter).

```
def process1(arr):
    processed_angles_1 = [[], [], [], [], [], []]
    last_point = 0
    for i in range(1, len(arr)):
        if arr[i] != arr[i - 1]:
            processed_angles_1[arr[i - 1]].append((last_point, i - 1))
            last_point = i
    # print(processed_angles_1)
    return processed_angles_1

def process2(arr):
    processed_angles_2 = [[], [], [], [], [], []]
    for i in range(len(arr)):
        for j in arr[i]:
            if j[1] - j[0] >= 3:
                processed_angles_2[i].append(j)
    # print(processed_angles_2)
    return processed_angles_2
```

```
def process3(arr, index, curr, length):
    current = curr
    if index < 6:
        if arr[index]:  # if the label at index is not empty
            for i in range(len(arr[index])):
                if not curr:  # curr empty
                    current.append(arr[index][i])
                    # print(1, index, current)
                    process3(arr, index + 1, current, 1)
                    break  # loop shouldn't continue after recursion above (index can be at 5, but goes back to 1 in for loop)
                elif arr[index][i][0] > curr[-1][-1]:  # current tuple comes after previous
                    current.append(arr[index][i])
                    # print(2, index, current)
                    process3(arr, index + 1, current, length + 1)
                    break  # loop shouldn't continue after recursion above (index can be at 5, but goes back to 1 in for loop)
                elif arr[index][i][0] < curr[-1][-1]:  # current tuple begins before previous
                    # print(current)
                    if current not in processed_array_1:
                        processed_array_1.append(current)
        else:
            process3(arr, index + 1, current, length)
    else:
        # print(current)
        if current not in processed_array_1:
            processed_array_1.append(current)
```

Figure 6. Screenshot of processing algorithms

After these processing algorithms are run, we will be able to select particular critical frames and synchronize our videos by linearly adjusting the speed of the videos to match these frames.

5. EXPERIMENT

First, we wanted to test the performance of various supervised learning methods in comparison with our initial approach. To do this, we had several algorithms that each used different models for supervised learning. In our experiment, we used Nearest Centroid, Nearest Component Analysis, Nearest Neighbors classification, as well as Multiclass SVM. First, we made an algorithm that allows users to manually label image frames from videos of tennis serves. We iterated through a set of youtube videos and manually labeled them from 1 to 6, each number representing a certain portion of the serve. For example, number 1 represents the preparation for the serve, number 5 represents when the player hits the ball, and other numbers represent other processes throughout the serve. For each label, we collected about 400 images. However, many of these images are very visually similar. After cutting down similar images, we resulted in over 100 images. Because there were very limited methods for obtaining more images of tennis serves, we decided that this number would be enough, but that we can also implement data augmentation to simulate different camera orientations and allow the algorithm to be more accurate with variations in video perspective. The results for each model are shown below.

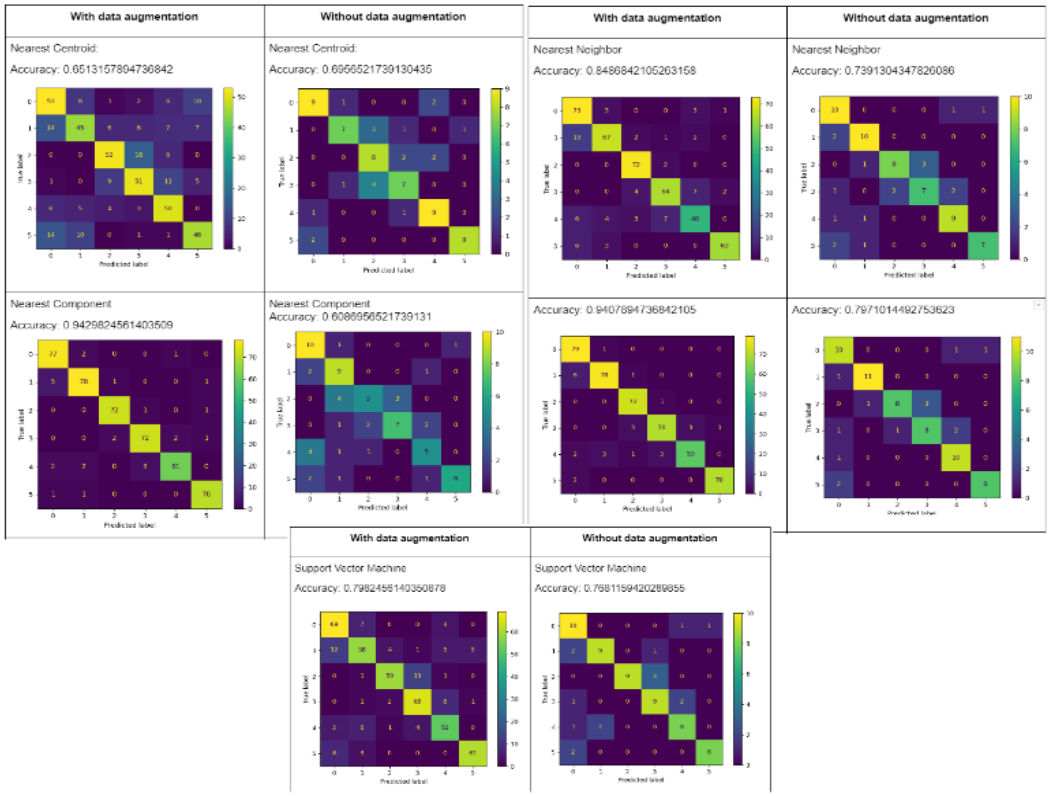

Figure 7. Experiment 1 results (the nearest neighbor approach on the top right uses uniform weights while the one below uses weights dependent on distance)

The addition of data augmentation improved the accuracy of three out of the four supervised learning algorithms. These results are very beneficial to our goal because it demonstrates that supervised learning is a potent tool for our problem, and can be implemented to synchronize videos.

When testing various supervised learning models, we found that certain augmentation features decreased the accuracy of our algorithms while others improved our algorithm. To find what augmentations worked best, we tested different combinations of augmentations and tested the accuracy of each model. Our experiments can be roughly divided into two portions. The first portion of our experiments will test the accuracy of various supervised learning models when using data from augmented images that have one of the four augmentation techniques. Importantly, every image has only one augmentation applied. In the second portion, we will test the accuracy of various supervised learning models using augmented images that have multiple augmentations applied. In this case, every image has at least two augmentations simultaneously applied to it. For both portions, every augmentation will have a 95% chance of acting on the image and 5% chance of doing nothing. This is because we want to preserve a portion of the original images as they are most reliably realistic.

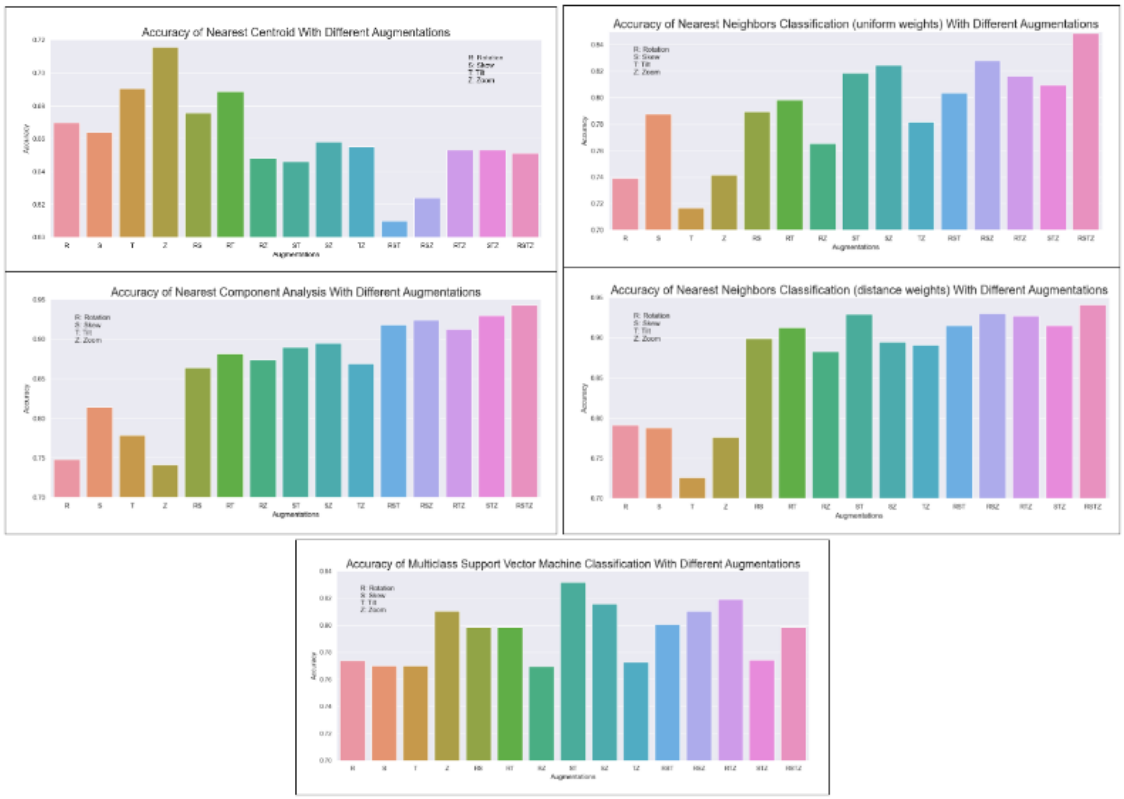

Figure 8. Experiment 2.1 results

The results above show that various models have different performances depending on the augmentations implemented. The Multiclass Support Vector Machine Classifier, for example, performed best when applied to images with Skew and Tilt applied. However, since we want a model that performs best with images of all four augmentations, the Nearest Component Analysis method appears to be the optimal choice.

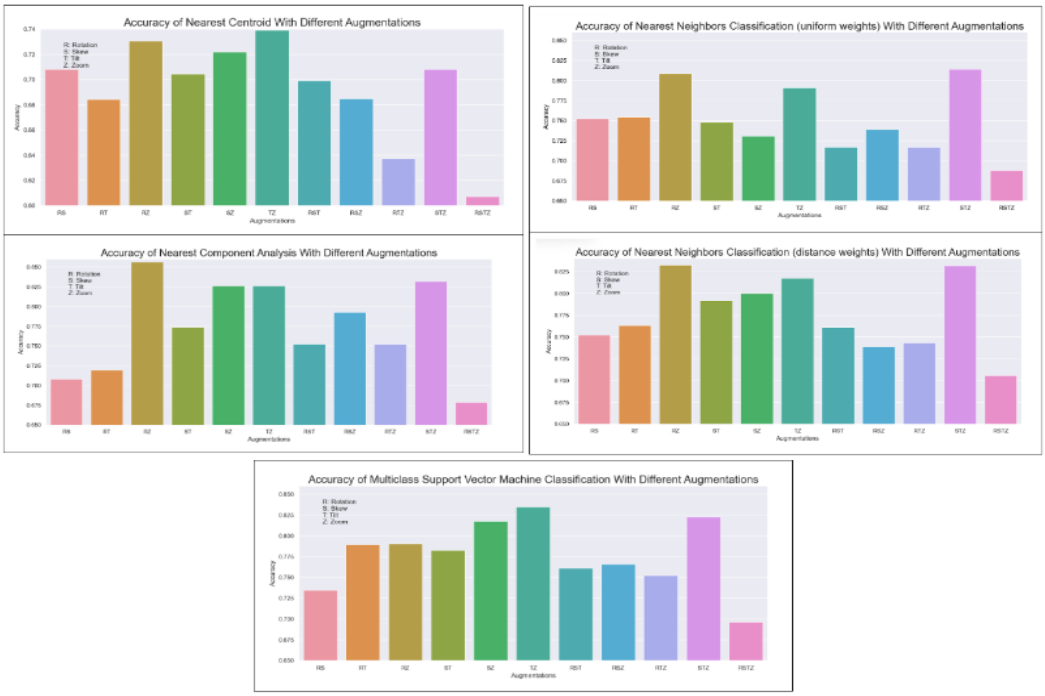

Figure 9. Experiment 2.2 results

The results for portion 2 shows much more fluctuation in accuracy than portion 1. The combination of different augmentations was also generally less effective in improving the accuracy of our model compared to portion 1. However, realistically, our model should utilize both data from portion 1 and portion 2, because camera perspectives are not limited to only one of the listed changes. After combining data from portion 1 and 2, we found that the Nearest Neighbors models with distance weights performed best. Although its accuracy is slightly lower than that of the nearest neighbors component analysis, this model had marginally better runtime.

6. CONCLUSIONS

By using various models in supervised and unsupervised learning, we developed an algorithm that offers an easy way to synchronize and compare two serves. We approached this through two ways: supervised and unsupervised learning. For the former, our algorithm will get pose data from all the frames of two uploaded videos. This data will then be used to cluster the frames within each video, respectively, into 6 groups. These groups, one from each video, will then be compared with each other to match similar frames. Using this clustering technique, our algorithm is able to synchronize two videos without manual labeling. The latter was implemented with several models. Data is generated through several algorithms and labeled manually. Using the data given, we used nearest centroid, nearest neighbor, nearest component analysis and multiclass SVM to classify a new frame [14]. This way, two videos can be synchronized as well. This can be very useful for tennis practice, as tennis players can easily compare their serve with another's serve, no matter who it is. Currently, we have created a website using Flask that allows users to use these algorithms and compare their performance. By inputting two videos, the website will give users the option to choose a processing method, and return two videos that are synchronized. Of course, various parts of our algorithm are still flawed in some ways. Firstly, the unsupervised learning approach is very limited by runtime. Videos that are significantly larger will experience significantly longer wait time. While the supervised learning approach performs slightly better with larger video files, the risk of long runtimes is still present. Because the pose data from each

frame is taken, the addition of frames will prolong runtime a decent bit. Since the data from videos are compared with each other through various loops in the unsupervised learning approach, the runtime in those processes will increase drastically with file size as well. In addition, our models can be easily fooled by adversarial attacks, because our approach assumes that the video is limited to only the serve. If someone was waving their hand repeatedly in a video, our algorithm might falsely believe that this is a tennis serve and work incorrectly.

Various improvements can be potentially made to improve our algorithm. Getting more data to train our algorithm for supervised learning, for one, can significantly improve the accuracy and consistency of our program. In addition, after identifying the key frames using supervised classification, our processing algorithms sometimes give bad groupings of frames. Thus, improving these processing algorithms can also lead to improvement of the resulting synchronized videos. Finally, training a unique pose-estimation program or modifying our approach can also result in a more efficient algorithm. In the future, many of these aforementioned improvements can be made to offer tennis players a better means of comparing their serves with others.

REFERENCES

[1] Ivanenko, Stanislav, et al. "Analysis of the indicators of athletes at leading sports schools in swimming." Journal of Physical Education and Sport 20.4 (2020): 1721-1726.
[2] Wei, Xinyu, et al. "Predicting serves in tennis using style priors." Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. 2015.
[3] Andriluka, Mykhaylo, et al. "2d human pose estimation: New benchmark and state of the art analysis." Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2014.
[4] Toshev, Alexander, and Christian Szegedy. "Deeppose: Human pose estimation via deep neural networks." Proceedings of the IEEE conference on computer vision and pattern recognition. 2014.
[5] Munea, Tewodros Legesse, et al. "The progress of human pose estimation: a survey and taxonomy of models applied in 2D human pose estimation." IEEE Access 8 (2020): 133330-133348.
[6] Patel, Manisha, and Nilesh Kalani. "A survey on Pose Estimation using Deep Convolutional Neural Networks." IOP Conference Series: Materials Science and Engineering. Vol. 1042. No. 1. IOP Publishing, 2021.
[7] Gao, Fei, et al. "Segmentation-based Background-inference and Small-person Pose Estimation." IEEE Signal Processing Letters (2022).
[8] Myers, Natalie L., et al. "Reliability and validity of a biomechanically based analysis method for the tennis serve." International journal of sports physical therapy 12.3 (2017): 437.
[9] Keller, Martin, Jonas Schweizer, and Markus Gerber. "Pay attention! The influence of coach-, content-, and player-related factors on focus of attention statements during tennis training." European Journal of Sport Science (2022): 1-9.
[10] Lugaresi, Camillo, et al. "Mediapipe: A framework for perceiving and processing reality." Third Workshop on Computer Vision for AR/VR at IEEE Computer Vision and Pattern Recognition (CVPR). Vol. 2019. 2019.
[11] Ganser, Andreas, Bernhard Hollaus, and Sebastian Stabinger. "Classification of Tennis Shots with a Neural Network Approach." Sensors 21.17 (2021): 5703.
[12] Skublewska-Paszkowska, Maria, Pawel Powroznik, and Edyta Lukasik. "Learning three dimensional tennis shots using graph convolutional networks." Sensors 20.21 (2020): 6094.
[13] Vinyes Mora, Silvia, and William J. Knottenbelt. "Deep learning for domain-specific action recognition in tennis." Proceedings of the IEEE conference on computer vision and pattern recognition workshops. 2017.
[14] Radev, Dragomir R., et al. "Centroid-based summarization of multiple documents." Information Processing & Management 40.6 (2004): 919-938.