Text in italics is explanatory and should be deleted in completed documents.

Project Name:	**Tennis Shot Action Recognition using Pose Estimation**
Name: Sandeep Polavarapu Venkata Naga	**Submission Date: Nov 11, 2022**
Project Period:	(e.g., June 1, 2022 – October 31, 2022)
Reporting period	(e.g., June 1, 2022 – October 31, 2022)

Section One: Summary

1. Using the THETIS dataset (a list of videos of tennis shots with the type of shot played), the model obtains the pose points (joints of the human body) and uses the temporal information about the movement of these joints over time to predict which shot is played. The model used for pose detection is selected from SOTA-available frozen models like Mediapipe and OpenPose. The action recognition model utilized is selected and trained on LSTMs, ConvCNN, and ViT. The purpose of the research is to understand the impact of using temporal information on poses and their ability to predict actions.

2. Using a temporal model on pose points of a player 2D image will result in comparable accuracies and lower compute time and complexity when compared with the advanced 3D-based SOTA models.

3. Tennis-based sports analytics has really picked up in the recent past. In tennis, a player must have a good pose to perform shots correctly and in the required manner. In this project, the proposal is to analyze the pose of a given tennis player and predict the shot he is playing.

Section Two: Activities and Progress

Part 1: Data Labeling: The number of frames was different across videos, so the skip-frame technique was used to select 40 frames in every video. The label of the data was determined from the file name and 4 labels were identified: forehand, backhand, serve and smash.

Part 2: Data Filtering: RGB videos were selected during the download. Videos with less than 40 frames were discarded.

Part 3: Affine Transformations: Since each video in the dataset is taken from a constant view and angle, it seems irrelevant to apply these transformations.

Part 4: Human Pose Estimation: Instead of OpenPose, MediaPose was utilized and the pose point for all the videos (1980) with 40 frames each was obtained.

Part 5: Final Data Creating: The data was created as a NumPy array of (No of time steps X (Pose points + 1)). +1 is for the label. No of time steps = 40. Pose points are given by 33 points with 4 values in each (x, y, z, visibility) = 133 values. Due to the immense processing power, I had to parallelize and utilize 8 cores to process data. (Took about 3 hours with parallel scripts).

Part 6: Model for Shot Prediction: The baseline to be utilized will be the LSTM model. Advances in this include Bi-LSTM, ConvLSTM, and ViViT which will be tested if time permits.

The Experimentation setup has been modified. Experiment 1: LSTM becomes our baseline. The SVM model will be tested later as LSTM would take more time to understand, train, and develop. Using more advanced models will be tested once the baseline works.

Section Three: Outputs and Deliverables

The data has been processed into a single large NumPy array of size (No of time steps X (Pose points + 1)).
Pose Detection was performed and the joint points were obtained correctly.

Attached in the supplementary files is: Image with poses identified.

Section Four: Evaluation

The LSTM model would be built on time and I would be able to provide decent results for the classification task. The Experimental setup also has been completed and is generalized.

Using a ConvLSTM and ViViT need to be reevaluated because they are heavy and complex models and would require not only a more sophisticated setup but also longer training with powerful machines.

These results could instead be taken as a comparative reference from papers that have implemented models on the same dataset.

Section Five: Risks, Issues, and Challenges

The dataset is relatively very huge and requires a lot of time to process. The obtained data is quite complex. It has 33 joints with 4 data points for each joint. There are 40-time steps and there is difficulty in organizing the data loader to feed this data into the LSTM model in the right format.

Parallel scripts were written to process the data which reduced computation time by 8x.

The data difficulty is a system design process. The LSTM model class has been written, and now I am trying to debug and figure out the correct way to input data via PyTorch DataLoader.

Training time is really high, thus the training will be performed on GPU-enabled machines. (Google Colab or a friend's Laptop or UMD Clusters are the available options).

Section Six: Collaboration

None. This is an individual project.

Section Seven: References and Related Works

- https://www.researchgate.net/publication/336659904_Prediction_of_Future_Shot_Directi
- https://medium.com/geekculture/swing-like-a-champ-with-computer-vision-backhand-trac
- https://github.com/chonyy/AI-basketball-analysis
- http://thetis.image.ece.ntua.gr/
- http://cs230.stanford.edu/projects_winter_2020/reports/32209028.pdf
- https://dl.acm.org/doi/abs/10.1145/3347318.3355523
- https://scholar.google.com/scholar?cites=7911374952132776078&as_sdt=20000005&sc

Section Eight: Requests

Utilizing LSTMs seems to be a complicated task on its own. Implementing ConvLSTM and ViViT seems to be rather difficult given the time and compute power constraints.

The request to limit the scope of the project to LSTMs on Pose Detections and compare with results from already implemented papers.

Section Ten: Next Steps

The first step would be to get the data structured so that it can be provided to the input.

Currently, we are considering 33 joints. For this problem, it could experiment with if only the upper body joints can be used. Another alternative feature that can be extracted is the angle and distance between certain relevant joints instead of the joint coordinates themselves.

Hyperparameter tuning and some data augmentation/computer vision techniques to enhance the image will be implemented.

Section Eleven: List of Equipment and Environments

The main code is run on M1 Macbook Air, 2020 model with 8 cores. This was utilized for downloading, storing, and processing the code

Google Colab with GPU access and is being considered for training.