Serving Insights: Improving Data-Driven Badminton Analytics with Computer Vision and Machine Learning

Word Count - 5166

Abstract - Recent improvements in the fields of computer vision and machine learning have led to a data revolution in sports. The introduction of novel approaches for measuring performance and strategy has transformed coaching and training methods. However, sports like badminton are yet to experience this boom due to the lack of an accessible end-to-end solution for data collection and analysis. This study aims to address this need by developing a pipeline for extracting positional data and training a model to classify the shots taken by badminton players.

This pipeline is developed with automation in mind, but falls back to human inputs where manual annotations produce more accurate results than trained models. The pipeline begins by prompting the user to identify the corners of the court and isolating the court area. Subsequently, video frames that do not contain the complete court view are filtered out through feature matching. Thereafter, pose estimation models are used to detect pose data and calculate joint angles for each player. This data is normalized and transformed into features in relation to the player's court position. Lastly, the processed data is used to train machine learning models to classify the strokes played during a rally.

The data obtained from this workflow can be used to train advanced models and develop new metrics to obtain a deeper understanding of effective playing styles. This sets the stage for a significant transformation in badminton research and training. It will also provide a roadmap for exploring other racquet sports like tennis and squash.

Keywords – Sports Science, Machine Learning, Computer Vision

1 INTRODUCTION

Data-driven analytics has the potential to serve as the new foundation for a robust and explainable measure of performance in sports. The field of sports science has been transformed by an "analytics boom" that has revolutionized coaching, analysis and preparation. This is due to two key factors:

Improvements in Data Collection – The rapid development and widespread accessibility of new tools for data collection has provided sports scientists with new data sources for analysis. For example, GPS sports vests, which have become a popular tool to track football player's performance during matches and training, contain sensors that are able to provide "660 raw data points per second" [1].

Increasing Model Complexity – Improvements in the accessibility of machine learning techniques have significantly lowered the barrier to entry for training complex models. This, along with the substantial increase in the quantity and quality of training data, has resulted in the rapid development of new models and metrics in different sports.

However, not every sport has been able to experience an "analytics boom". This can primarily be attributed to the inequality in the accessibility of this data. Data collection processes can be expensive if third-party services are hired, and time-consuming if teams or players choose to collect it on their own.

To address this problem and make data-driven insights accessible to badminton players of all levels, we propose constructing a pipeline for data collection that automates the process of data collection and processing from videos of badminton matches.

2 DATA PROCESSING PIPELINE

The pipeline is currently designed to accept videos of men's singles badminton matches from the user, and produces the following outputs:

1. Location of players on the court
2. Joint angles of players
3. Joint distances of players
4. Type of shot taken by each player during a match

This section details the processing steps that are taken to derive the desired outputs from the input match video.

2.1 VIDEO PRE-PROCESSING

In the proposed pipeline, we assume that the first frame of the input video contains a full view of the badminton court. This condition was enforced to simplify the process of isolating the court area. Given the development time constraints of this project, the decision was made to simplify the initial stages of data extraction to reduce computational overhead while creating a standard input format.

2.1.1 Isolating Court Area

Isolating the court area is an essential step for removing noise from the input video. The primary

source of noise in recordings of badminton matches comes from persons standing outside the court area, such as umpires and line judges. This affects the accuracy of downstream pose estimation tasks, which are essential to the output of our data pipeline.

Using the OpenCV Python package [2], the first frame of the video is displayed to the user. Fig. 1 shows a sample of the output displayed to the user during this stage.

Figure 1 First Frame of the Video Displayed to the User

The user is then prompted to select the four corners of the court in the following order: bottom left, top left, top right, bottom right. Once the user has selected the corners of the court, the program slightly extends the corner points by a percentage of the video's dimensions to ensure that no important information is lost during the masking process. The original coordinates selected by the user are also saved for future use.

Using OpenCV, a black mask of the same size as the frame is created, and a white polygon is created using the extended court coordinate points. The resulting bitwise AND operation isolates the court area while excluding the pixels in the rest of the frame. This mask is subsequently applied to the rest of the video. Fig. 2 shows a sample of the output from this masking step. Comparing Fig.1 and Fig.2, we can see that the noise from the umpires and linesmen have been removed, resulting in greater accuracy in downstream analysis tasks.

Figure 2 First Frame of the Video after Masking

2.1.2 Filtering Frames with Feature Masking

Videos of broadcasted tournament matches may contain segments that are not relevant for the task of match analysis, such as replays and close-ups. As such, an automated solution is needed to eliminate gameplay interruptions and discard frames that do not contain the full court view.

To achieve this, we use the feature detector and feature matching functionalities provided by the OpenCV Python package. Using the ORB (Oriented FAST and Rotated BRIEF) feature detector, 3000 keypoints and descriptors are detected and computed in each frame of the input video. These detected features are subsequently matched with a reference frame that contains the full court view using a BFMatcher (Brute-Force Matcher). Frames with more than 1250 valid matches of detected features are retained for the final output video. Fig. 3 illustrates an example of feature matching between two frames in a video. The lines between the frames on the left and right connect matched features between the two frames.

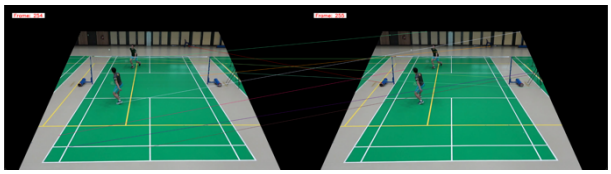

Figure 3 Output of Feature Matching between Two Valid Frames

If the frame does not contain at least 1250 valid matches, transformations are applied to make the frame appear greyed and blurred. This effect was chosen to prevent the poses of the players from being detected during replays, while allowing users to discern what is happening in these frames. Fig. 4 showcases an example of a frame with the effects applied.

Figure 4 Example of Discarded Frame after Feature Matching

After these pre-processing steps, the videos are ready to be used as input for detecting the poses of the badminton players.

2.3 POSE ESTIMATION

2.3.1 Pose Estimation Model

OpenPose [3] was selected as the model of choice for the pose estimation task due to its state-of-the-art performance and capabilities in multi-person pose estimation. The versatility, reliability and active community support offered by OpenPose made it an ideal candidate for the backbone of our data pipeline.

The OpenPose model was integrated with the Sports2D Python package [4]. Designed for analysing 2D sports footage, the package abstracts the model inference process and can directly generate the pose estimation data needed for our pipeline. Fig.5 highlights an example frame with the inferred locations of the badminton players' pose keypoints. The coordinates of the following keypoints are provided for every detected keypoint in the frame are listed in Table 1.

Table 1 Detected OpenPose Keypoints

Body Part	Keypoints
Head	Nose, REye, LEye, REar, LEar, Neck
Shoulders	RShoulder, LShoulder
Elbows	RElbow, LElbow
Wrists	RWrist, LWrist
Hips	CHip, RHip, LHip
Knees	RKnee, LKnee
Ankles	RAnkle, LAnkle
Feet	RBigToe, LBigToe, RSmallToe, LSmallToe, RHeel, LHeel

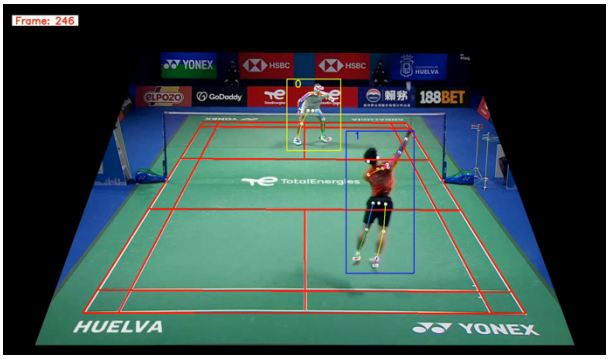

Figure 5 Example of Annotated Frame after Inference with Sports2D and OpenPose

The Sports2D package further provides a functionality to track the coordinates of a detected player across frames. The package outputs a series of CSV files containing the movements of each tracked player across the duration of the video. The files contain the x and y coordinates and the likelihood of accuracy for the detected keypoints listed in Table 1.

2.3.2 Pose Estimation Post Processing

2.3.2.1 Perspective Transformation

The output coordinates of pose detection models are provided with reference to the input video resolution. Therefore, it is necessary to obtain the location of the badminton players with respect to the court to be able to compare readings across different videos.

Using the court coordinates obtained in Section 2.1.1, a homography matrix is calculated via OpenCV's getPerspectiveTransform function. This returns a 3x3 transformation matrix that can be used to map points from the image plane of a camera to the plane of a badminton court. The location of the court corners obtained via the user are mapped to the points listed in Table 2 below.

Table 2 Coordinates of Badminton Court Corners in Reference Plane

Corner	Coordinate
Bottom Left	(0, 1340)
Top Left	(0,0)
Bottom Right	(610,0)
Top Right	(610,1340)

The homography matrix is applied to the output coordinates from the Sports2D package to obtain the location of the player with respect to the badminton court.

2.3.2.2 Sports2D Output Segregation

While Sports2D provides a player tracking functionality, instances of 'swapping' IDs between players were observed during testing. This can be observed in Fig. 6 and Fig.7. The bottom player was assigned an ID of 4 in Fig.6, but in a subsequent frame in Fig. 7, was assigned an ID of 3.

Figure 6 Example of Player ID Swapping – Initial Assignment

Figure 7 Example of Player ID Swapping – Subsequent Assignment

Therefore, further post-processing is required to assign the correct set of coordinates to the players located in the top and bottom half of the badminton court. This ensures that the output data is accurate and does not contain unnecessary information such as the poses of line judges or ballboys, which are often also detected in the footage.

We begin by identifying and extracting frames that have been filtered using the methodology in Section 2.1.2. These frames are characterized by a low likelihood of detection for key body parts such as hips, knees, ankles and heels, and negative y coordinates after transformation. These frames are still represented in our final output to maintain the integrity of the dataset.

Next, for each tracked player output by Sports2D, pose coordinates are filtered based on if they belong to the top or the bottom half of the court. This is achieved using the court coordinates in the reference plane defined in Table 2.

The final output from this step is segregated CSVs for top and bottom half coordinates for each player ID tracked by Sports2D. From this, we can reconstruct the final movements made by the top and bottom player.

2.3.2.2 Assigning Coordinates to Top and Bottom Players

Finally, the data from the segregated top and bottom halves is used to generate the coordinates of the top and bottom players of the match. The CSVs from Section 2.3.2.1 are merged and sorted in ascending order by frame number and the y coordinate of the detected right ankle. This joint was chosen as it is the closest marker to the badminton court plane and hence has the lowest error during perspective transformation. Since the coordinate origin in OpenCV is the bottom left of the video, we assume that the top player will have a larger right ankle y axis value as they occupy the highest most valid location on the transformed badminton court.

2.3.2.3 Pose Data Filtering

Tracked pose data can contain noise due to inaccuracies in detections. Therefore, there is a need to perform data smoothing to reduce distortions while preserving the shape and features of the original data.

The Savitzky-Golay Filter was chosen for this task due to its ability to reduce noise and preserve data features with its sliding window approach. When applied to the pose coordinates, the filter performs the following steps:

1. Read the data points within the specified window length.
2. Fit a polynomial to these points using the defined polynormal order.
3. Replace the central point in the window with the point of the polynomial.
4. Repeat steps 1-3 until the entire series has been processed.

However, before filtering the data, it is necessary to re-calculate the position of the players with respect to the original dimensions of the video. This is to eliminate errors that may arise from the homography transformation process.

Therefore, an inverse perspective transform is calculated using the homography matrix derived in Section 2.3.2.1. After the original coordinates for the top and bottom player are obtained, the data is filtered using an implementation of the Savitzky-Golay Filter in the SciPy Python package [5]. The input parameters are as follows:

1. Window Length – half of the input video's FPS (frame per second)
2. Polynomial Order – 1

The final output from this step is the filtered pose data for the top and bottom player with respect to the original coordinates of the input video. From this, we can proceed to calculate input features for the shot classification models.

2.3.2.4 Feature Calculations

Given the coordinates of the detected joints of the players, it is possible to calculate some additional features that can be used to provide more user-friendly output and serve as a more powerful input for shot classification models.

Joint Angles

The coordinates of detected joints are represented as 2D vectors. Therefore, to calculate the angle between joints, the dot product formula must be used, as shown in Fig. 8.

$$\cos\theta = \frac{u \cdot v}{||u||\,||v||}$$

Figure 8 – Dot Product Formula

This calculation is performed for all combinations of detected joints. The final angle reported is the minimum of the two combinations. Additionally, for the ball and socket joints (shoulders and hips), the abduction and horizontal abduction angles are calculated. The abduction angle refers to the shoulder angle between the elbow and the hip, while the horizontal abduction angle refers to the angle between the elbow and the opposite shoulder.

Euclidian Distances

Using the formula in Fig.9, the Euclidian Distances between detected joints can be determined.

$$d(x,y) = \sqrt{\sum_{i=1}^{n}(y_i - x_i)^2}$$

Figure 9 – Euclidian Distance Formula

However, variations in camera configuration can result in a distortion of the raw distances between joints. To account for these, min-max normalization is carried out to standardize the distances between joints in each frame to a range of 0 to 1. This mitigates the effect of variations in camera setups. The normalized value is derived using the formula in Fig. 10.

$$X' = \frac{X - X_{min}}{X_{max} - X_{min}}$$

Figure 10 – Min-Max Normalization Formula

This ensures that all distances are within the same range, allowing for the comparison of data points across frames. However, a drawback of this approach is that it compresses the values into a square "box," which can distort the shape of the detected pose, leading to the loss of information. Fig. 11 depicts a side-by-side comparison of the original video frame and the distorted image after normalization.

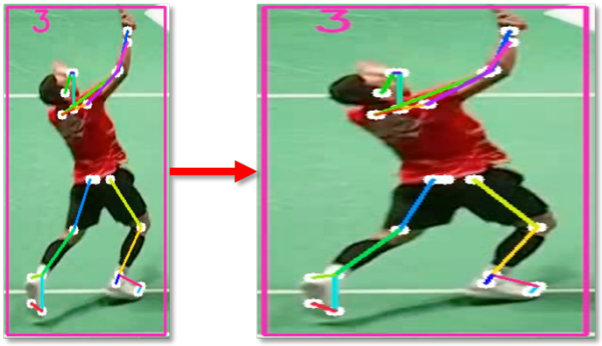

Figure 11 – Before-and-After Comparison of Min-Max Normalization

To address this distortion, it is necessary to use another normalization approach that considers the aspect ratio of the input video. The aspect ratio is defined as the ratio of the width to the height of the input frame. Multiplying the x-axis coordinates by the aspect ratio before normalization ensures that the proportional relationship between width and height is maintained. This step is crucial because it aligns the x-coordinates with the scale of the y-coordinates, preserving the true geometric shape of the player's pose. The outputs from this stage are CSV files for the bottom and top players, containing the following information for all keypoints listed in Table 1:

1. Detected x and y coordinates.
2. Likelihood of detection.
3. Coordinates of the centre of mass of players.
4. Joint angles of keypoints.
5. Normalized distances between from each keypoint to the other keypoints.

For the purposes of model training, a truncated version of this dataset is created and saved. As different input videos can contain different camera angles, some data points obtained from our pipeline are prone to higher deviations. To prevent our classification models from learning incorrect information, we chose to retain features that are resistant to these deviations. Therefore, only the calculated angles and normalized aspect ratio distance are used training our shot classification models.

3 SHOT CLASSIFICATION MODELS

To achieve the task of automatically classifying the type of shot taken by a player, it is necessary to train a classification model that takes in a sequence of frames just before and after a player hits a shuttlecock. It is also necessary to construct a training dataset consisting of different shots taken by a player and their movements as recorded by our pipeline, as detailed in Section 2. This section aims

to elaborate on our training data collection process, model training methodology and explain our findings.

3.1 TRAINING DATASET

To construct our training dataset, badminton match videos were sourced from YouTube, ensuring a diverse range of gameplay scenarios and player actions (IRB-2023-442). These videos were then downloaded and systematically processed using the data processing pipeline detailed in Section 2. After extracting the positional data, manual annotation of videos was performed to mark the frames during which player actions were carried out. To streamline the annotation process, we utilized the Computer Vision Annotation Tool (CVAT) [6]. Currently, the data from six quarter-final, semi-final and final matches of the YONEX French Open 2023 have been processed and annotated. However, this annotation process is still ongoing as we look to construct an extensive training data set consisting of a diverse range of camera angles and players.

3.1.1 Annotation Tags

Table 3 lists the annotation tags that are grouped by category.

Table 3 Annotation Tags Grouped by Category

Category	Values
Player	Top Player, Bottom Player
Grip	Forehand, Backhand, Overhead
Shot Type	Lob, Drop, Smash, Drive, Block, Net, Lift, Tap, Push, Serve Low, Serve High
Shot Direction	Straight, Middle, Cross
Outcome	Winner, Forced Error, Unforced Error

For every shot taken in the input video, the corresponding frame is marked, and the following tags applied:

1. **Player** – to identify if the top or bottom player played the shot
2. **Grip** – to identify the type of grip used during the shot
3. **Shot Type** –
    a. **Lob –** refers to a shot played from the player's back court to the opponent's back court.
    b. **Drop –** refers to a shot played from the player's back court to the opponent's front court, low and close to the net.
    c. **Smash –** refers to a shot that is mostly played from the player's back court in a fast, attacking manner.
    d. **Drive –** refers to a flat shot that can be played from the player's mid-court or backcourt. It can be a straight shot, cross-court shot, or even to the body of the opponent.
    e. **Block –** refers to a shot mostly played when receiving a smash from the opponent.
    f. **Net –** refers to a shot played from the player's front court to the opponent's front court, near the net.
    g. **Lift –** refers to a shot played from the player's front court to the opponent's back court, high in the air.
    h. **Tap –** refers to an attacking shot played at the player's front court.
    i. **Push –** refers to a flat lift shot played from the front of the player's court to the middle or back court of the opponent.
    j. **Serve Low –** refers to the shot played by the player at the beginning of a possession, from the player's front court to the opponent's front court
    k. **Serve High –** refers to the shot played by the player at the beginning of a possession, from the player's front court to the opponent's back court
4. **Shot Direction –**
    a. **Straight –** If the shuttle does not cross the middle line when moving from the player's court to the opponent's court, it is considered a straight shot.
    b. **Middle –** If the shuttle goes toward the center zone of the court, it is considered a middle shot.
    c. **Cross –** When the shuttle crosses the vertical middle line of the court, it is considered a cross shot.

At the beginning of a sequence, the player playing the serve and the type of serve is marked. At the end of a sequence, one of the following tags is applied:

1. **Winner –** When the opponent misses the shuttle, or the shuttle touches the ground before the opponent has a chance to play it.
2. **Forced Error –** A point lost by the opponent as a direct result of a well-executed or strategically placed shot, which compels the opponent into making a mistake, such as hitting the shuttle into the net or out of bounds.

3. **Unforced Error –** A point lost due to the player's own mistake, typically occurring in the absence of significant pressure from the opponent.

After the annotation process has been completed for a match, the tags are merged into the complete and truncated datasets generated from our data collection pipeline. Additionally, a 'padding' size of frames half the frame rate of input video is applied for the frames before and after the shot was taken. In the context of this training dataset, the training videos were 30 frames per second (FPS) and thus 12 frames were padded. For example, if a serve was taken at the 100th frame of a video, the sequence from frames from frame 88 to 112 are marked as a serve. This allows us to easily obtain and process the movements made by the player before and after the shot was taken. This data is stored in CSV files and serves as the input data for the shot classification models.

3.2 MODEL TRAINING

3.2.1 Data Loading

The CSV files from the data annotation process are loaded and processed using the Pandas Python package [7]. The following is the data loading methodology:

1. Remove rows containing values of -999. These are used to mark frames that have been filtered with feature masking.
2. Extract the columns containing pose angles and normalized distances.
3. Using the Pandas package, mark the row numbers containing the shot types for classification.
4. Group consecutive frames corresponding to each shot type as a sequence.
5. Track the maximum sequence length across all files and apply padding if necessary to ensure a uniform input to the model.
6. Map the shot types for classification to integer values.

Data loading and model training is handled by the PyTorch Python package [8]. A custom class is implemented to facilitate data loading and to enable batch processing.

3.2.1 Grip Classification Model Training

To reduce the complexity of the initial exploration of the model training methodology, we chose to focus on an annotation category that contains a fewer number of classes. The "Grip" category from Table 3 meets this criterion and hence Forehand (FH), Backhand (BH), and Overhead (OH) were used as the target labels for shot classification.

3.2.1.1 Model Architectures

The performance of three different model architectures were tested:

1. Recurrent Neural Networks (RNNs),
2. Long Short-Term Memory networks (LSTMs), and
3. Convolutional Neural Networks (Conv2D)

These model architectures were chosen due to their capabilities in handling sequential and spatial data. RNNs excel at capturing temporal dependencies within sequential data, making them suitable for analysing the frame-by-frame progression of player movements. LSTMs were selected as they are better able to capture long-term dependencies in temporal data. Lastly, Conv2Ds were selected due to the model architecture's strengths in spatial feature extraction. By comparing these architectures, we aim to determine which model best balances the temporal and spatial aspects of the input data, providing the most accurate shot classification.

3.2.1.2 Model Training Parameters

The following parameters were constant across the training of the three models:

1. Number of Epochs – 10
2. Learning Rate – 0.001
3. Train / Test Split – 60:40
4. Loss Function – Cross Entropy Loss
5. Optimizer – Adam Optimizer

For each architecture, three models were trained:

1. Model using the Top Player Data
2. Model using the Bottom Player Data
3. Model using the Combined Dataset

This decision was made to evaluate if segregating the dataset for the top and bottom player provides better results. The following section details the and statistical measures calculated for the test set of each model architecture.

3.2.1.2.1 Model Results

Bottom Player Data

Table 4 F1 Score for Models on Bottom Player Test Data Set

F1 Score	RNN	LSTM	Conv2D
Class			
FH	0.689	0.831	**0.940**

OH	0.739	0.862	**0.954**
BH	0.834	0.933	**0.948**

Table 5 Precision Scores for Models on Bottom Player Test Data Set

Precision	**RNN**	**LSTM**	**Conv2D**
Class			
FH	0.717	0.835	**0.946**
OH	0.734	0.857	**0.957**
BH	0.752	0.933	**0.922**

Table 6 Recall Scores for Models on Bottom Player Test Data Set

Recall	**RNN**	**LSTM**	**Conv2D**
Class			
FH	0.664	0.826	**0.933**
OH	0.744	0.866	**0.951**
BH	0.936	0.933	**0.975**

Top Player Data

Table 8 F1 Score for Models on Top Player Test Data Set

F1 Score	**RNN**	**LSTM**	**Conv2D**
Class			
FH	0.788	0.889	**0.900**
OH	0.653	**0.889**	0.881
BH	0.810	0.893	**0.902**

Table 9 Precision Scores for Models on Top Player Test Data Set

Precision	**RNN**	**LSTM**	**Conv2D**
Class			
FH	0.885	0.908	**0.947**
OH	0.538	**0.856**	0.810
BH	0.873	0.832	**0.948**

Table 10 Recall Scores for Models on Top Player Test Data Set

Recall	**RNN**	**LSTM**	**Conv2D**
Class			
FH	0.710	**0.871**	0.857
OH	0.833	0.925	**0.965**
BH	0.756	0.858	**0.860**

Combined Player Data

Table 11 F1 Scores for Models on Combined Player Test Data Set

F1 Score	**RNN**	**LSTM**	**Conv2D**
Class			
FH	0.708	0.809	**0.915**
OH	0.767	0.767	**0.932**
BH	0.877	0.900	**0.934**

Table 12 Precision Scores for Models on Combined Player Test Data Set

Precision	**RNN**	**LSTM**	**Conv2D**
Class			
FH	0.590	**0.933**	0.867
OH	0.822	0.659	**0.982**
BH	0.831	0.850	**0.940**

Table 13 Recall Scores for Models on Combined Player Test Data Set

Recall	**RNN**	**LSTM**	**Conv2D**
Class			
FH	0.884	0.715	**0.968**
OH	0.656	**0.917**	0.887
BH	0.928	**0.957**	0.928

3.2.1.3 Observations

Tables 4-13 reveal several insights into model performance on the shot classification task.

Across bottom, top, and combined player data, the Conv2D model consistently yields the highest F1 scores, with particularly strong performance on FH and OH shots. For instance, on the bottom player dataset, Conv2D achieves F1 scores of 0.94 on FH, 0.954 on OH, and 0.948 on BH, significantly outperforming both RNN and LSTM. This suggests that Conv2D is more effective at capturing the spatiotemporal patterns inherent in the sequence data, possibly due to its ability to preserve and learn local spatial features within the input frames. We therefore recommend adopting this model architecture for subsequent studies.

Additionally, all three model architectures were effective in accurately classifying OH shots. For example, on the combined dataset, even the RNN, which performs worst overall, achieves a relatively high OH F1 score of 0.767. This suggests that OH shots exhibit distinctive pose features that are easier for models to learn, while FH and BH share overlapping kinematic signatures that lead to greater misclassification.

There is also a notable difference in performance between separate models for top and bottom players against a combined model. While the

Conv2D model maintained high accuracy even in the combined setting, the RNN and LSTM architectures generally experienced performance degradation when trained on the merged dataset. This trend indicates that the positional context of the player introduces variability in pose sequences, which can hinder model generalization. By training separate models for each player position, this variability is minimized, allowing the models to focus more effectively on learning position-dependent shot patterns.

4 FUTURE DIRECTIONS

This study has identified several avenues for future research and improvements to our current pipeline.

4.1 AUTOMATED COURT CORNER DETECTION

Currently, user interaction is needed to identify the court corners during the initial data processing steps. This design choice was made due to the time constraints of this study. However, this can be burdensome for users who wish to process multiple videos. Therefore, future iterations of this pipeline can develop and implement keypoint detection models that automatically detect the location of court features.

Dynamic detection of court features can also enable frame-wise calculation of homography matrices. This is valuable in scenarios where the broadcast camera is panning between different parts of the court. By continuously updating the homography matrix to maintain a consistent top-down or standardized court view, the system can perform player tracking and spatial localization across varying perspectives. This significantly improves the robustness, scalability, and automation level of the video analysis pipeline, particularly for use in live match analysis or large-scale datasets with non-static cameras.

4.2 TRAINING SHOT CLASSIFICATION MODELS

The results from our grip classification models show that the Conv2D architecture performs extremely well for action recognition on our input dataset. Future development of this pipeline should hence focus on training similar classification models for the other tags in Table 3, along with exploring methodologies to improve the performance of existing models.

4.3 SHUTTLECOCK TRACKING

We propose the implementation of TrackNetV3 [9] to enhance the precision of shot detection in badminton matches. TrackNetV3 offers the capability to track the location of a shuttlecock in a video. From this, we can determine the trajectory and velocity of the shuttlecock throughout a game. Furthermore, by analyzing the changes in shuttlecock velocity, we can obtain the exact frame when a shot is taken. This in turn enables us to use our trained shot classification models to infer the type of shot taken by a player. Although the current project timeline did not allow for the exploration and implementation of TrackNetV3, it represents a promising direction for future research to refine and enhance the shot detection and classification process in badminton video analysis.

5 CONCLUSION

In summary, this work lays the groundwork for a scalable, intelligent badminton video analysis system. The project began with the creation of a data-processing pipeline. This included video pre-processing, isolating the court area, filtering frames using masking techniques, and using advanced pose detection models. Post-processing steps involved perspective transformation, segregation of outputs, assignment of coordinates to players, and feature calculations.

During the training of the grip classification model, we collated and annotated a training dataset and explored the performance of various model architectures on three different datasets. Our findings indicate that the Conv2D model architecture outperform LSTM and RNN, and training separate models for top and bottom players significantly reduces classification errors.

Building on these insights, we propose several avenues for future improvement. Automating the court corner detection step using keypoint-based models can enhance the usability and scalability of the system. Additionally, expanding the classification pipeline to support other shot attributes would significantly broaden the system's analytical capabilities. Finally, incorporating shuttlecock tracking using TrackNetV3 will provide additional data points to improve the quality of results.

6 REFERENCES

[1] M. Carey, "How the growth of wearable technology is transforming football - The Athletic," *The Athletic*, 2023.

[2] G. Bradski, "The OpenCV Library," *Dr. Dobb's J. Softw. Tools*, 2000.

[3] Z. Cao, G. Hidalgo Martinez, T. Simon, S. Wei, and Y. A. Sheikh, “OpenPose: Realtime Multi-Person 2D Pose Estimation using Part Affinity Fields,” *IEEE Trans. Pattern Anal. Mach. Intell.*, 2019.

[4] D. Pagnon, “Sports2D - Angles from video,” *GitHub repository*. GitHub, 2023, doi: 10.5281/zenodo.7903963.

[5] P. Virtanen *et al.*, “{SciPy} 1.0: Fundamental Algorithms for Scientific Computing in Python,” *Nat. Methods*, vol. 17, pp. 261–272, 2020, doi: 10.1038/s41592-019-0686-2.

[6] CVAT.ai Corporation, “Computer Vision Annotation Tool (CVAT).” Nov. 2023, [Online]. Available: https://github.com/cvat-ai/cvat.

[7] W. McKinney, “{D}ata {S}tructures for {S}tatistical {C}omputing in {P}ython,” in *{P}roceedings of the 9th {P}ython in {S}cience {C}onference*, 2010, pp. 56–61, doi: 10.25080/Majora-92bf1922-00a.

[8] J. Ansel *et al.*, “PyTorch 2: Faster Machine Learning Through Dynamic Python Bytecode Transformation and Graph Compilation,” Apr. 2024, doi: 10.1145/3620665.3640366.

[9] Y. J. Chen and Y. S. Wang, “TrackNetV3: Enhancing ShuttleCock Tracking with Augmentations and Trajectory Rectification,” *Proc. 5th ACM Int. Conf. Multimed. Asia, MMAsia 2023*, Dec. 2023, doi: 10.1145/3595916.3626370.