FieldVision: A Top-Down Mapping of Tennis Players Using MeanShift and Adaptive Kalman Filters

Nicolas Carpenter, Viet Chu
ID: 1533367, 1692927

April 14, 2024

Abstract

This paper presents a system for tracking player movements in sports, with a specific focus on tennis. The system uses Meanshift tracking with background subtraction to track players from relatively static backgrounds. This is complemented by the implementation of an adaptive Kalman filter, which not only predicts the future state of each player but also dynamically adjusts its parameters in response to occlusions and variable detection confidence. This ensures reliable and accurate tracking under varying conditions. Additionally, the Direct Linear Transformation (DLT) algorithm is utilized to project the 2D positions of players onto a standardized tennis court model, enabling precise analysis of movement and positioning. The synthesis of these methodologies provides a comprehensive tool for analyzing player dynamics and offering insights for coaching.

Department of Computing Science
University of Alberta

Contents

1 Introduction

Tracking objects in video sequences is a critical task in computer vision with extensive applications across various fields, including surveillance, interactive systems, and sports analytics. In tennis, effective tracking of players provides crucial insights that are beneficial for coaching, detailed game analysis, and enhanced broadcasting experiences.

The inherent challenges of tracking in sports, such as rapid player movements, diverse lighting conditions, and frequent occlusions, demand robust and adaptive tracking solutions. Tennis, with its fast-paced nature and players often obscured by nets or other players, embodies these challenges. Therefore, a tracking system must be capable of accurately following players amidst these dynamic conditions.

This paper discusses an approach leveraging the MeanShift tracking algorithm integrated with KNN background subtraction and an Adaptive Kalman Filter. Furthermore, we refine tracking accuracy by applying domain-specific constraints that consider the structured environment of tennis. The adaptability of our model ensures consistent and reliable tracking throughout the match.

2 Methods

2.1 Meanshift Tracking

Our model uses OpenCV's implementation of the MeanShift algorithm, which is based on Comaniciu et al. (2000), to track tennis players throughout matches. This choice is driven by the algorithm's computational efficiency and robust tracking capabilities under diverse lighting conditions and rapid movements typical of tennis environments.

2.1.1 Details of MeanShift Tracking

The MeanShift algorithm in OpenCV provides a non-parametric feature-space analysis. It operates by iteratively shifting to the densest part of a feature space, defined by the color distribution within the target window. This iterative process involves computing the centroid of the distribution of the target points within the search window, represented by the equation:

$$x_{t+1} = \frac{\sum_{i=1}^{n} x_i K(x_i - x_t)}{\sum_{i=1}^{n} K(x_i - x_t)} \tag{1}$$

where x_t is the current location, x_i are the sample points within the window, and K is the kernel function defining the weight of nearby points, typically based on color features.

2.1.2 Integration with KNN Background Subtraction

To enhance the performance of MeanShift tracking in the highly dynamic and variable conditions present in tennis videos, we integrate it with KNN background subtraction. This

combination significantly improves the segmentation of players from the complex backgrounds of tennis courts, enabling more precise tracking of player movements. The choice of KNN background subtraction over traditional histogram-based methods is supported by its superior performance in handling varying lighting and complex dynamic changes in video scenes, as detailed in Salhi & Jammoussi (2012).

2.1.3 Incorporating Domain-Specific Constraints

To further refine our tracking system's accuracy, we incorporated a net constraint specific to tennis, inspired by Fang et al. (2014). This constraint leverages the prior knowledge that players typically do not cross to the opposing team's side of the net during a game. By applying this constraint, the algorithm can more accurately segment and track individual players by maintaining their location relative to their starting half of the court. This enhances the system's robustness against complex player dynamics and overlapping by simplifying the tracking problem using the predictable layout of the tennis court.

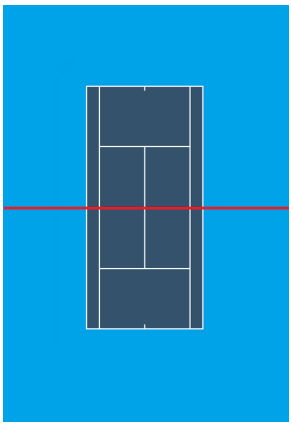


Figure 1: Red line indicates the team boundary that cannot be crossed.

2.2 Adaptive Kalman Filter

In tennis, player movements are quick and characterized by frequent and abrupt directional changes. Additionally, occlusions are common, with players often obscuring each other or being hidden by various on-court elements. To effectively manage these challenges, our system employs an Adaptive Kalman Filter in conjunction with the MeanShift algorithm. This hybrid approach significantly outperforms a sole reliance on MeanShift, which may struggle with the dynamic and unpredictable nature of sports environments.

2.2.1 Kalman Filter Model

First, we consider the standard Kalman Filter as defined by Kalman (1960), which offers a robust estimation framework for predicting the state of a moving object in noisy environments. Understanding this basic model is essential as it forms the basis upon which our adaptive enhancements are built. We will now delve into the components of the Kalman Filter, outlining how each part contributes to its overall functionality.

State and Measurement Models The state vector $\mathbf{x}_k$ encapsulates the physical properties of the object being tracked. It includes both positional and velocity components, enhancing the prediction capabilities of the system:

$$\mathbf{x}_k = [x_k, y_k, \dot{x}_k, \dot{y}_k]^T \tag{2}$$

This composite model allows the filter to anticipate future locations based on both the current position and velocity, reflecting the inherently dynamic nature of moving objects.

The measurement vector $\mathbf{z}_k$ contains the observed positions that we receive from our MeanShift tracker:

$$\mathbf{z}_k = [x_k, y_k]^T \tag{3}$$

These observations can be generally noisy, and the Kalman Filter's role is to mitigate this noise, thereby enhancing the reliability of state estimations over time.

State Transition and Measurement Matrices The state transition matrix Φ is fundamental to the filter's prediction phase and is defined to encapsulate the expected motion of the object between measurements:

$$\Phi = \begin{bmatrix} 1 & 0 & dt & 0 \\ 0 & 1 & 0 & dt \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4}$$

Where dt is our sampling interval, which is the inverse of the video's fps. The configuration that we chose assumes constant velocity, simplifying the computation while still yielding satisfactory predictions.

The measurement matrix H, which directly maps the state vector to the measurement vector, focuses solely on positional data:

$$H = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \tag{5}$$

This implies that while the measurements directly observe positions, velocity is inferred and updated within the state estimation process based on the differences in positional data over time.

Covariance Matrices The configuration of the covariance matrices Q and R is pivotal in tuning the Kalman Filter to the specific dynamics and noise characteristics of the tracking environment. These matrices play critical roles in how the filter predicts and corrects the state vector, balancing between the model's inherent predictions and the incoming measurements.

- **Process Noise Covariance (Q):** This matrix represents the expected variability or uncertainty within the model itself, particularly accounting for the unpredictable nature of player movements:

$$Q = \begin{bmatrix} 0.01 & 0 & 0 & 0 \\ 0 & 0.01 & 0 & 0 \\ 0 & 0 & 0.05 & 0 \\ 0 & 0 & 0 & 0.05 \end{bmatrix} \tag{6}$$

  The values on the diagonal for the position components (0.01 for both x and y) suggest a moderate level of process noise, indicating a reasonable but not complete confidence in the model's positional predictions. The higher values (0.05) for the velocity components reflect greater uncertainty in estimating velocities, which is expected given the rapid and abrupt movements typical in sports.

- **Measurement Noise Covariance (R):** This matrix quantifies the anticipated errors in the measurements, arising from factors such as sensor inaccuracies, environmental influences, and notably, the inherent noise from the MeanShift tracker:

$$R = \begin{bmatrix} 0.1 & 0 \\ 0 & 0.1 \end{bmatrix} \tag{7}$$

  The elements 0.1 for the x and y positions indicate that the measurements are fairly reliable but include notable noise levels. This noise is significantly influenced by the MeanShift tracker, which, while effective in tracking player movements, can introduce measurement variability especially in dynamically changing scenes typical of sports environments. This setting acknowledges that while the measurements are trusted to a reasonable extent, they are not perfect and reflect the imperfections introduced by the tracking algorithm alongside other environmental factors.

Larger values in Q would make the filter more adaptive to changes in motion by trusting the incoming measurements more over the model predictions, whereas larger values in R would make the filter more conservative, relying more on the model by mistrusting the noisy measurements. The current settings have moderate trust in both the model's dynamics and the measurement accuracy, which is suitable for environments where both components are fairly predictable but subject to occasional deviations.

Prediction and Update Equations Having outlined the fundamental components of the Kalman Filter, we now focus on the prediction and update steps that iteratively refine the state estimates.

Prediction:

$$\hat{\mathbf{x}}_k^- = \Phi \hat{\mathbf{x}}_{k-1}^+ \tag{8}$$

$$\mathbf{P}_k^- = \Phi \mathbf{P}_{k-1}^+ \Phi^T + Q \tag{9}$$

Update:

$$K_k = \mathbf{P}_k^- H^T (H \mathbf{P}_k^- H^T + R)^{-1} \tag{10}$$

$$\hat{\mathbf{x}}_k^+ = \hat{\mathbf{x}}_k^- + K_k(\mathbf{z}_k - H\hat{\mathbf{x}}_k^-) \tag{11}$$

$$\mathbf{P}_k^+ = (I - K_k H)\mathbf{P}_k^- \tag{12}$$

Prediction Phase This phase projects the current state estimate ahead in time, maintaining a continuous estimate of the system's state even without new measurements:

- **State Prediction:** Equation 8 predicts the state at time k based on the previous updated state. The state transition matrix Φ models the expected evolution of the state over time.
- **Covariance Prediction:** Equation 9 updates the covariance to forecast the uncertainty of the state estimate at time k, incorporating the process noise Q to account for inherent model uncertainty.

Update Phase This phase uses new measurements to refine the predicted state, enhancing its accuracy and correcting any deviations:

- **Kalman Gain:** Equation 10 calculates the Kalman Gain to balance the weight between the predicted state and the new measurement, adjusting how much the measurement influences the updated state.
- **State Update:** Equation 11 corrects the predicted state using the new measurement $\mathbf{z}_k$, which helps align the model's prediction with the observed reality.
- **Covariance Update:** Equation 12 reduces the estimated error in the state prediction, increasing the certainty following the incorporation of new measurement information.

These structured steps ensure that the Kalman Filter continuously adapts its state estimates based on the latest observed data.

2.2.2 Adaptive Adjustments

After detailing the operation of the standard Kalman Filter, we now describe modifications that enhance its adaptability. Our Adaptive Kalman Filter model adjusts its parameters dynamically, responding to changes in detection conditions like occlusions and variable detection confidence.

Handling Occlusions During occlusions, where measurements become less reliable due to players being temporarily obscured or blocked, adjustment of the noise covariance matrices Q and R is critical. We assume that all players are being tracked, so we determine occlusion by analyzing the percentage overlap of regions of interest (ROIs) of each tracked player. If the overlap exceeds a threshold (40% in our case), an occlusion is assumed.

- **During Occlusion:** To increase the influence of the model's predictions during occlusions, the filter decreases Q. This adjustment makes the model more trustworthy compared to the less reliable measurements:

$$Q' = Q \times 0.1 \quad (\uparrow \text{model confidence by} \downarrow \text{process noise}) \tag{13}$$

  Simultaneously, the filter increases R to reflect the decreased reliability of the measurements made by our MeanShift tracker. This change makes the filter less responsive to the current measurement data, which is likely contaminated by noise due to occlusion:

$$R' = R \times 10^6 \quad (\downarrow \text{measurement confidence by} \uparrow \text{measurement noise}) \tag{14}$$

- **Post-Occlusion:** Once occlusion is no longer detected, the filter resets Q and R to their default values to normalize the tracking sensitivity.

Adjustment Based on Detection Confidence Detection confidence (α) is assessed by how much of the ROI is filled with our tracked object, which is quantified directly from the output of the MeanShift tracker. This percentage directly influences how Q and R are adjusted:

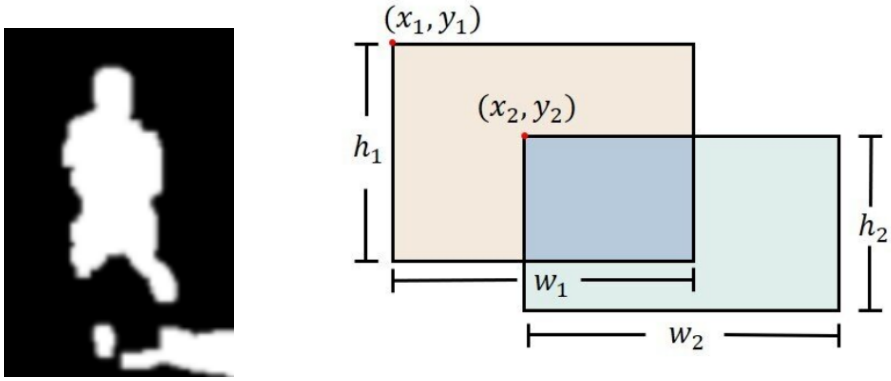
- **Confidence Adjustment:** The filter adjusts Q and R inversely based on the detection confidence level. A higher confidence level, indicated by a larger coverage of the ROI, decreases R more significantly than Q to skew the balance towards reliance on measurements. Conversely, a low coverage rate signals low confidence, prompting an increase in Q to emphasize the model's predictions:

$$Q' = \mathrm{Q} \times \alpha \qquad \text{where } \alpha \in [0.01,\, 0.99] \tag{15}$$

$$R' = \mathrm{R} \times (1 - \alpha) \qquad \text{where } \alpha \in [0.01,\, 0.99] \tag{16}$$

This strategy ensures that the filter is less reactive to noisy measurements during low confidence intervals, thus preventing erratic estimation behaviors. Conversely, when the confidence is high, the filter trusts the measurements more, providing more responsive tracking.

These adjustments are implemented directly within the filter's update cycle, ensuring that the state estimation is continually optimized for the current conditions.



(a) (b)

Figure 2: a) Alpha value calculated from the filled portion (white) of the ROI. b) Occlusion determined by the overlap percentage between ROIs (blue region).

2.3 Direct Linear Transformation

The Direct Linear Transform (DLT) algorithm stated in Hartley & Zisserman (2000) estimates a homography, denoted as H, between two images capturing a planar scene. This process relies on identifying four or more 2D point correspondences between the images. Subsequently, we apply the computed homography H to transform the perspective of the first image to align with the second image. This approach allows us to map specific points of interest, such as the corners of a tennis field onto a top-down 2D representation of the tennis court.

The usage of DLT algorithm in our project contains 2 steps: (1) Compute the homography H, and (2) Apply H to map the 2D locations of the players

The transformation is performed using matrix multiplication:

$$\mathbf{P}' = \mathbf{P} \cdot \mathbf{H} \tag{17}$$

where $\mathbf{P}'$ is the transformed position, $\mathbf{P}$ is the original position in homogeneous coordinates, and $\mathbf{H}$ is the homography matrix retrieved from the DLT algorithm.

$\mathbf{P}$ is represented as a 3D homogeneous coordinate vector:

Algorithm 1 DLT Algorithm for 2D to 2D point mapping

Input: Corresponding 2D points $\{(x_i, y_i) \rightarrow (u_i, v_i)\}_{i=1}^{N}$, where $N \geq 4$
Output: Homography matrix $\mathbf{H}$
Construct the design matrix $\mathbf{A}$:
  Initialize $\mathbf{A}$ as an empty matrix
for $i = 1$ to N **do**
    Append row $[-x_i, -y_i, -1, 0, 0, 0, u_i x_i, u_i y_i, u_i]$ to $\mathbf{A}$
    Append row $[0, 0, 0, -x_i, -y_i, -1, v_i x_i, v_i y_i, v_i]$ to $\mathbf{A}$
end for
Compute the Singular Value Decomposition (SVD) of $\mathbf{A}$:
  $\mathbf{A} = \mathbf{U} \cdot \mathbf{D} \cdot \mathbf{V}^T$
Extract the last column of $\mathbf{V}$ as the solution vector $\mathbf{h}$:
  $\mathbf{h} = [h_1, h_2, h_3, h_4, h_5, h_6, h_7, h_8, h_9]^T$
Reshape $\mathbf{h}$ into a 3×3 matrix $\mathbf{H}$:
  $\mathbf{H} = \begin{bmatrix} h_1 & h_2 & h_3 \\ h_4 & h_5 & h_6 \\ h_7 & h_8 & h_9 \end{bmatrix}$
Normalize $\mathbf{H}$ by dividing by h_9:
  $\mathbf{H} = \frac{1}{h_9}\mathbf{H}$
return $\mathbf{H}$

$$\mathbf{P} = \begin{bmatrix} x & y & 1 \end{bmatrix} \tag{18}$$

2.4 Integration

All of these methods come together to form our tracking system. The procedure is shown in Algorithm 2, which integrates MeanShift, the Adaptive Kalman Filter, and homography transformations to process video frames and produce standardized player positions for consistent analysis across different games and camera angles.

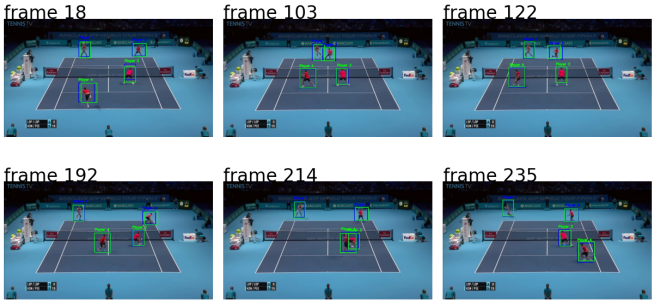
3 Results

Our experimental evaluation tested the proposed method across 5 tennis doubles clips collected from Tennis TV (2020), encompassing varied playing surfaces including composite and clay courts. Notably, the composite courts exhibited a spectrum of colors, posing a challenge to traditional segmentation techniques.
Each clip featured dynamic gameplay characterized by rapid movements of all players, with duration restricted to less than 15 seconds in 540p resolution and 30fps. Furthermore, we operated under the assumption of consistent camera parameters and illumination levels

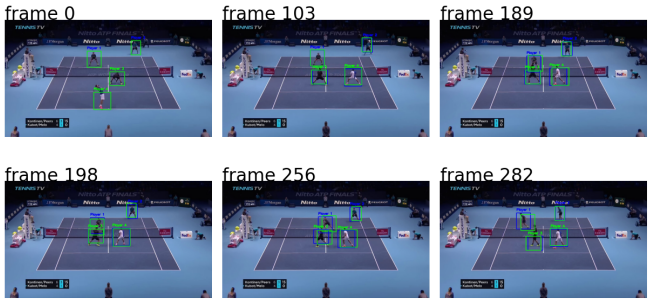
Algorithm 2 Meanshift and Adaptive Kalman Filter Tracking

Input: Video frames, initial player positions
Output: Tracked positions of players throughout the video
Initialize background subtractor and capture video
Select initial ROIs interactively from the first frame
Initialize Kalman filters for each ROI
Calculate homography to align video with standard court model using DLT
while frames are available **do**
    Apply background subtraction and preprocess the mask
    for each ROI **do**
        Predict new state with Kalman filter
        Update ROI based on prediction
        Apply Meanshift to locate the new center of ROI
        Update the ROI in the frame
        Calculate overlap and alpha value for adaptive adjustments
        Adjust Kalman filter based on occlusion and alpha
        Update Kalman filter with new measurement
        Store new player positions
    end for
    Transform and draw player positions on tennis court view
    Display updated frame and tennis court view
    Check for user termination command
end while

throughout the entirety of the video sequences. We also assume that the duration of each occlusion is typically less than 1 second. The ROIs of players on the same team are approximately similar in size when being initialized. Points of interest used to calculate the homography matrix H are the 4 corners of the tennis field, which are manually selected on the first frame of each video. We also prioritize tracking the lower half of the player over the upper half. Clips 1 and 3 depict scenarios where two players from the same team experience significant occlusion, while one player exhibits rapid movement in a predictable trajectory. There are also some camera movements in clip 1. Clip 2 illustrates a situation where players from opposing teams undergo partial occlusion for 3 seconds. In Clip 4, we encounter a more chaotic environment characterized by multiple instances of occlusion. Finally, clip 5 portrays players from the same team undergoing occlusion while moving in the same direction.



(a)

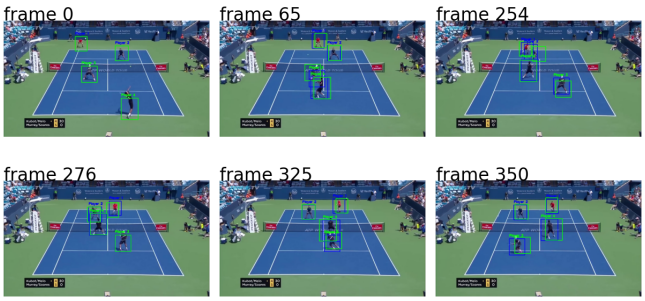


(b)

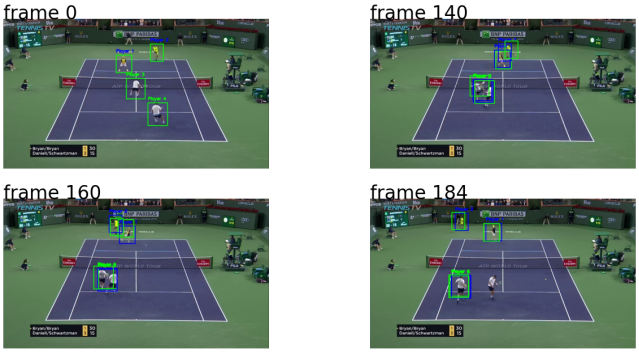
Figure 3: a) Clip 3: Short, but significant occlusions b) Clip 2: Long occlusions constrained by the net

The blue tracking ROIs are the Mean shift tracker and the green ROIs are the Kalman Filter predictions. In clip 3 frames 192-235, we can see that the tracker handles the occlusion well when player 4 is moving in a straight line at a very high speed while only obstructing player 3 for a short time. In clip 2 frames 198-282, it shows that the net

constraints keep the players on their side of the field, successfully handling longer occlusions. Clip 4 shows the tracker handling occlusions consecutively. Player 4 occludes player 3 in frame 65, player 2 occludes player 1 in frame 254, player 4 occludes player 2 in frame 276, then player 3 occludes player 4 in frame 325. The tracker fails in clip 5 when players 3 and 4 move in unison, however, it still succeeds in tracking players 1 and 2 when they collide.



(a)

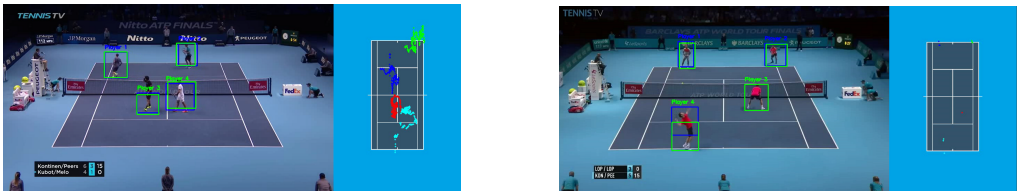


(b)

Figure 4: a) Clip 4: Multiple occlusions in the same video b) Clip 5: 2 players moving in the same direction

As for DLT mapping, overall, the mapping seems to correctly display the position of the player on the field. There are still occasional small inaccuracies in the top-down view of the players. This is because when players move closer or farther away from the perspective camera view, the player's size shrinks or expands accordingly while the size of the track window remains static. Hence, the DLT algorithm warps the incorrect point of interest in the 2D view.

When we relax the camera assumptions, the tracker shows bad mapping and tracking results due to the corners' movement. The camera movement also drastically changes the location of static objects in the scene, resulting in a noisier background, leading to falsely detected moving objects.



(a) (b)

Figure 5: a) Clip 2: DLT mapping b) Clip 3: Inaccurate mapping of player 4 on the 2D field

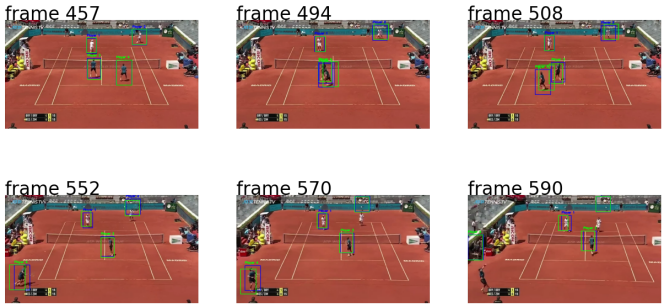


Figure 6: (Clip 1) Tracking players with camera movement

4 Conclusion

This project focuses on simultaneously tracking multiple moving objects (players) while accounting for occlusion in the case of a static camera view. After testing with different videos, we conclude that this method is robust in drastically varying coloring settings without many manual or case-specific preprocessing steps. The adaptive tracker consistently performs well when there are short occlusions and predictable player movement. The net constraint is a very powerful addition to the tracker as it accounts for long occlusions while not relying on statistical predictions. Combined with the net constraint, the adaptive tracker is capable of dealing with complex situations where multiple occlusions happen during a tennis double video sequence.
Some limitations of this method include slight mapping errors due to Mean shift inability to adjust the size of the ROIs window, and failure to track when two players collide when moving in the same direction. The window size problem can be addressed by using Cam shift, or by estimating objects' size using camera calibrations and triangulation.

References

Comaniciu, D., Ramesh, V., & Meer, P. (2000). Real-time tracking of non-rigid objects using mean shift. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, (pp. –). Hilton Head Island, SC: IEEE.

Fang, M.-Y., Chang, C.-K., Yang, N.-C., Kuo, C.-M., & Guang, S.-K. (2014). Robust player tracking for broadcast tennis videos with adaptive kalman filtering. *Journal of Information Hiding and Multimedia Signal Processing*, *5*(2), 242–262.

Hartley, R., & Zisserman, A. (2000). *Multiple view geometry in computer vision*. Cambridge University Press.

Jernbäcker, A. (2022). Kalman filters as an enhancement to object tracking using yolov7. Technical Report 99999-99, KTH Royal Institute of Technology. TRITA – SCI-GRU 2022:345.

Kalman, R. E. (1960). A new approach to linear filtering and prediction problems. *Journal of basic Engineering*, *82*(1), 35–45.

Salhi, A., & Jammoussi, A. Y. (2012). Object tracking system using camshift, meanshift and kalman filter. *World Academy of Science, Engineering and Technology, International Journal of Electronics and Communication Engineering*, *6*(4), 421–426.

Tennis TV (2020). 10 minutes of incredible doubles tennis. YouTube video. URL `https://www.youtube.com/watch?v=G2klD`$_1 15vM$