POLITECNICO
MILANO 1863

SCUOLA DI INGEGNERIA INDUSTRIALE
E DELL'INFORMAZIONE

Trajectory and Spin Detection in a Bowling Throw

Authors: Davide Corradina, Michele Fassini

Advisor: Prof. Vincenzo Caglioti

Academic year: 2024-2025

Abstract

This paper presents a novel, fully classical computer vision pipeline for recovering both the two-dimensional trajectory and rotational spin of a bowling ball from a single arbitrary video. No deep-learning models, special markers, or calibrated multi-camera rigs are required: our method operates on low-resolution footage captured from a freely placed, potentially rotating camera.

We first localize the playing surface by combining Hough-based line detection with template matching of pin outlines to identify the near and far lane boundaries. Next, per-frame circular Hough detection isolates the bowling ball's center in image space. A homography computed from the detected lane lines then maps every ball detection into a reconstructed, bird's-eye view of the lane, yielding a continuous, metric-scale trajectory. To estimate spin, we compute dense optical flow within the ball's silhouette across consecutive frames, from which we robustly recover the instantaneous axis of rotation and angular velocity throughout the throw. Crucially, our approach accounts for perspective distortion and is tolerant to video noise, camera motion, and variable lighting, enabling accurate spin estimation even when the ball's surface texture is minimal.

The main contributions of this work include a calibration-free lane reconstruction method that extracts accurate homography from low-quality video, an integrated trajectory-and-spin recovery pipeline that jointly analyzes a bowling ball's motion and rotation using only classical vision methods, and a robust optical-flow-based technique for extracting angular velocity and rotation axis under challenging conditions.

The full implementation of the system, along with example videos and documentation, is available at: GitHub repository.

1. Introduction

Accurate measurement of both the trajectory and rotational spin of a bowling ball is critical for performance analysis, equipment design, and athlete coaching in competitive bowling. Precise quantification of these parameters enables coaches to tailor training regimens, optimize ball selection, and provide real-time feedback on release technique. Traditional inertial or marker-based systems, while accurate, impose cost and hardware constraints that limit deployment in everyday practice sessions and lane-side environments.

Recently, computer vision and deep-learning approaches have emerged as powerful alternatives for sports analytics, but they typically demand large annotated datasets, specialized hardware, or multi-camera rigs. In contrast, we propose a fully classical, calibration-free vision pipeline capable of extracting two-dimensional trajectory and spin information from a single, arbitrary video. Our method requires only low-resolution footage from a freely placed camera—potentially subject to rotation and without any specialized markers—making it highly practical for in-lane analysis and broad adoption. The approach is robust to variable frame rates, enabling consistent analysis even when videos are recorded using different devices or settings.

The core components of our approach include automatic lane detection via Hough-based line extraction and pin-template matching, per-frame ball localization using circular Hough transforms, homography-based lane reconstruction to recover metric trajectories in bird's-eye view, and dense optical flow within the ball silhouette to estimate instantaneous axis of rotation and angular velocity. This end-to-end pipeline handles perspective distortion, video noise, and lighting variation, allowing robust performance under challenging conditions. This is the first end-to-end classical vision system that jointly tracks a bowling ball's three-dimensional path and rotational dynamics from a single camera.

In summary, while deep learning and professional multi-sensor systems offer state-of-the-art accuracy, classical pipelines—based on geometric reconstruction, Hough detection, and optical flow—remain valuable for low-cost, markerless setups like ours. Our work leverages and integrates these classical components into a practical system for tracking both the trajectory and spin of a bowling ball from unconstrained video input.

The remainder of this paper is organized as follows. Section 2 reviews related work in sports-vision analytics and classical spin estimation. Section 3 details our lane detection and homography-based reconstruction, describes our ball localization and trajectory mapping, presents the optical-flow spin estimation pipeline. Section 4 reports video results and discusses performance. Finally, Section 5 concludes and outlines directions for future work.

2. State of the Art

This section reviews prior work relevant to vision-based ball tracking, spin estimation, and camera-based sports analytics. The discussion is organized into classical methods for lane and trajectory reconstruction, traditional object detection and tracking, spin estimation techniques, and modern data-driven approaches. Our focus is specifically on pre-deep-learning methods, with brief comparisons to more recent alternatives to contextualize our contributions.

2.1. Line Detection

Line detection is fundamental in structuring scenes for sports analytics by identifying lane boundaries and other geometric references. The classical Hough Transform [5] is a robust method for detecting straight lines in edge-detected frames and is commonly used to extract lane edges in sports like bowling or tennis. The Probabilistic Hough Transform [9] improves efficiency by reducing the number of edge points sampled, making it suitable for real-time applications. Alternatively, edge-based approaches combined with RANSAC line fitting have shown good robustness in variable lighting conditions.

2.2. Trajectory Reconstruction Using Homography

Classical techniques often use planar homographies to map 2D video frames onto a model of the playing field or lane. For instance, Yu et al. [16] employed homography to map ball detections from broadcast tennis videos onto a calibrated court, enabling bounce point estimation despite camera motion. Van Zandycke and De Vleeschouwer [14] introduced a method that decouples diameter estimation and 2D localization, enabling accurate 3D reconstruction from a single image when camera calibration is known.

2.3. Ball Detection and Tracking

Ball detection has traditionally relied on shape-based methods such as the Hough Circle Transform. Bradski and Kaehler [3] described a complete pipeline using circle detection refined by heuristics and classifiers for real-time object recognition, which has been widely adopted due to its simplicity and robustness. Širmenis and Lukoševičius [15] tested a similar approach in basketball, combining background subtraction with Hough detection to achieve reliable shot tracking, albeit with sensitivity to occlusion. In soccer, Park and Seo [10] used a fusion of color, shape, and motion cues integrated into a particle filter for robust real-time tracking under variable lighting.

2.4. Spin Estimation Techniques

Accurate spin estimation is challenging due to limited texture on the ball surface and motion blur. Boracchi et al. [1] proposed a method for recovering the spin of a ball using a single image by leveraging the ball's known geometry and image measurements. Tamaki et al. [13] proposed a model-based registration approach that fits the observed motion of ball features to a rotational model, solving for the angular velocity and axis of spin. More recently, event-based cameras have been applied to this problem. Gossard et al. [6] exploited the sparse output of event cameras to detect distinctive logo features on a table tennis ball, enabling real-time estimation of spin axis and velocity. Additional precision can be achieved using high-speed, multi-exposure techniques as described by Shum and Komura [12], though such setups require controlled lighting and patterned balls.

2.5. Modern Learning-Based and Professional Systems

Although our work is focused on classical techniques, recent progress in deep learning has significantly advanced detection and tracking accuracy. Learning-based models like those reviewed by Zhang et al. [17] offer accurate lane detection using deep CNNs but require large datasets and are often domain-specific. Dosovitskiy et al. [4] proposed FlowNet, a CNN architecture trained to estimate optical flow end-to-end, outperforming classical

methods in many cases but requiring large datasets and computational resources. Professional systems such as Hawk-Eye [7] use multi-camera rigs and complex calibration pipelines to deliver sub-centimeter accuracy in both trajectory and spin estimation but are prohibitively expensive and inaccessible to most users. Lightweight, open-source examples—such as YOLO-based ball trackers demonstrated by Roboflow [11]—highlight the trade-offs between accessibility, accuracy, and hardware requirements.

3. Proposed Approach

The proposed pipeline is organized into a series of well-defined modules, each responsible for a key component of the analysis: lane detection, ball detection, lane reconstruction, trajectory reconstruction, and spin estimation. The following sections provide a detailed description of each module and its role in the overall system.

3.1. Lane Detection

Given an input video, the goal of the lane detection algorithm is to identify the four boundary lines that define the bowling lane from which the player is throwing the ball. The method operates under the following assumptions: the lane is approximately centered within the frame, all four lane boundaries are visible in the initial frames of the video, the lane surface exhibits the standard wooden appearance commonly used in bowling alleys, and the scene is reasonably well illuminated.

3.1.1 Background Motion Estimation

In order to improve the quality of the detection, a background motion estimation is performed to determine how much the camera is moving during the video. This step helps distinguish between actual object motion (e.g., the ball or the player) and camera-induced motion, which could otherwise interfere with line detection and tracking.

The method relies on detecting and matching visual features between consecutive frames using the ORB (Oriented FAST and Rotated BRIEF) algorithm. By tracking how these features shift over time, the algorithm estimates a global transformation—specifically, a homography—between each pair of frames. To ensure robustness against moving foreground elements, RANSAC is used to filter out inconsistent matches, focusing on the static background.

From the estimated homographies, only the translation components are extracted, and their magnitude is used to quantify the amount of motion.

This information is then used to improve subsequent detection steps, helping to ensure the stability and accuracy of lane boundary extraction even in non-static video recordings.

3.1.2 Bottom Line Detection

Edges extraction. The detection process focuses on the lower part of each frame, where the bottom line of the lane is expected. Each frame is cropped to retain only the bottom quarter and converted to HSV color space to better isolate the lane surface. Two color ranges—light brown and rose—are targeted using binary masks, which are then combined to highlight the lane region.

After optional blurring to reduce noise, the image is converted to grayscale. Edges are extracted using the Canny algorithm with adaptive thresholds computed via Otsu's method. The resulting edge map is analyzed using a Probabilistic Hough Transform to detect straight lines.

Best estimate selection. Only those segments with a nearly horizontal orientation are retained, than the desired line is chosen as the most marked between the remaining ones. This procedure is repeated across all video frames, producing a sequence of candidate lines that are passed to the next stage for refinement.

Post-Processing. Once candidate bottom lines are extracted across all frames, a series of steps is applied to refine the trajectory and remove spurious detections. First, each line is converted to a point by computing its intersection with a reference direction (orthogonal to the line passing through the origin). This creates a sequence of points representing uniquely the lane edge over time.

Outlier removal is performed based on frame-to-frame distance and statistical deviation from the overall distribution. If the video exhibits sufficient camera motion, additional smoothing (via median and Savitzky-Golay filters) and interpolation are applied to reconstruct a smooth and continuous trajectory. For static videos, a single average line is computed instead.

3.1.3 Lateral Lines Detection

Edge extraction. The lateral line detection process builds upon the previously identified bottom line, which serves as a geometric reference along the lane floor. For each frame, a central point on this bottom line—aligned with the horizontal center of the image—is used to spatially separate candidate lines into left and right groups relative to the bowler's viewpoint.

Each frame is first pre-processed with Gaussian blur followed by Canny edge detection to highlight lane markings and lane boundaries. The Probabilistic Hough Transform then extracts line segments representing potential lane edges, it is particularly useful in this application because often the player cover, partially or totally, the desired line.

Best estimate selection. To focus on meaningful lane boundaries, lines that are nearly horizontal or positioned below the bottom line are filtered out, since valid lateral lines are generally vertical or diagonal and located over the baseline.

The filtered lines are used to compute their intersection with the bottom one and their relative position with respect to the central reference point classifies them as either left or right boundaries. Among candidates on each side, the closest line to the center is selected to ensure consistent and stable lane boundary detection across frames.

Post-Processing. The postprocessing of the detected lateral lines is performed independently between left and right lines, following a strategy analogous to that applied to the bottom line.

Outlier removal plays a crucial role in enhancing the reliability of the lane lines. Lines exhibiting slopes that significantly deviate from the average are discarded to ensure ge-

ometric consistency, similar to the approach used for bottom line refinement. Then, the closest points on each line relative to the origin are extracted.

Depending on the estimated average movement in the video sequence, two different outlier filtering methods are employed. For dynamic scenarios with noticeable movement, density-based clustering (DBSCAN) is applied to eliminate isolated or spurious points. Further, spatial filtering is performed independently on the horizontal and vertical coordinates to remove points exhibiting abrupt deviations. Missing or removed points are then interpolated to maintain continuity. In contrast, for nearly static sequences where motion is minimal, an iterative distance-based filtering is used, followed by averaging the remaining points to produce a stable line estimate. This assumes limited variation over time and leverages temporal smoothing to reduce noise.

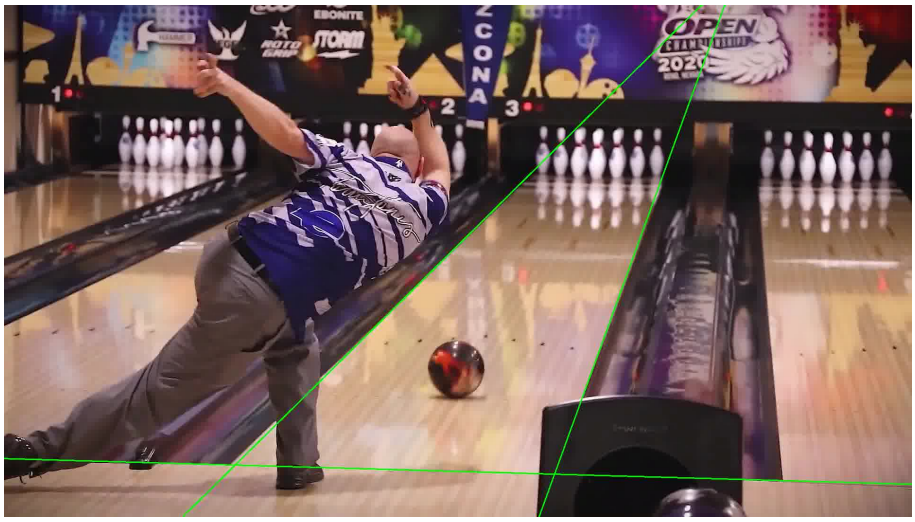

Figure 1: Bottom and lateral lines processed

3.1.4 Upper Line Detection

Pin template dimension estimate. First, the triangular region defined by the intersection of bottom and lateral lines is computed. This triangle delimits the area where the upper line is expected. The frame is masked to isolate this triangular region, focusing the analysis on the relevant part of the image.

Within the triangle, a color filter in the HSV color space is applied to extract pixels corresponding to brown and rose hues as in the previous steps.

An initial estimate of the upper line's vertical position is computed by scanning from the bottom vertex of the triangle upward. The method searches for the first horizontal row where more than 98% of pixels are black, indicating a predominantly dark area. This row defines a horizontal line across the frame as the rough upper line position.

Template matching. To refine this estimate, template matching is applied to the filtered image. A template representing the pin is resized according to the apparent scale in the frame. The scaling factor is derived from the ratio between the real lane width and the measured distance between the intersections of the upper line estimate with the lateral lines. The template matching process identifies the location within the filtered frame that

best matches the resized template, providing the bottom point of the pin along the upper line.

Due to the high difficulties encountered in the estimation of its slope, we considered the horizontal lines passing through the point, because it has been evaluated as a negligible source of error.

Outliers removal. The vertical positions of the top-left and top-right corner points are extracted frame-by-frame. Outliers in these vertical coordinates are identified by comparing each value with a local neighborhood of frames, exploiting the temporal continuity of the lane shape. Detected outliers are removed, and the missing values are replaced using linear interpolation between adjacent valid frames. This step ensures a smoother temporal evolution and reduces abrupt changes caused by detection errors.

Upper corners estimate. In cases where interpolation alone is insufficient—particularly when large portions of the upper line are missing—the vertical positions of the top corners are estimated proportionally based on the relative displacement of the corresponding bottom lane boundaries. This proportional scaling leverages the geometric relationship between the top and bottom lines of the lane, maintaining consistent lane geometry even when the upper boundaries are partially unavailable.

Non-visible bottom line. A specific challenge arises when the bottom lane line is no longer visible in the frame, due to occlusions or limited camera field of view (as in Figure 2). In such situations, the method falls back on a predictive strategy: the vertical position of the upper lane corners is maintained based on the last reliable measurements and the bottom corners are computed from the last available relative displacement with respect to upper ones. This approach prevents drastic and unstable fluctuations in the lane geometry, allowing the system to sustain a reasonable approximation of the lane boundaries even in the absence of bottom line data.

Smoothing. To further enhance temporal consistency, a sliding window approach is employed to smooth sudden fluctuations in the vertical position of the upper lane corners. By tracking the displacement trends over multiple consecutive frames, the method reduces jitter and ensures a stable lane contour throughout the video sequence.

Still videos. Finally, in scenarios characterized by minimal vehicle or camera movement, a simplified correction is applied by fixing the vertical position of the upper lane corners at the average height observed during the initial frames, because in the last part of the video the ball hit the pins and the detection is poor. This approach reduces noise in static or near-static conditions, providing a stable upper boundary representation without complex temporal adjustments.

Figure 2: Lane detection - non visible bottom line.

3.2. Ball Detection

Once the lane vertices have been identified in each frame, we proceed to locate the bowling ball's center.

Initial detection. To reduce false positives, we first apply a binary mask that restricts detection to the lane area. At first we further limit this mask the lower portion of the lane, which usually does not include the reflection of the pins which can confuse the detector.

The ball is initially detected using OpenCV's `HoughCircles` with lenient parameters to maximize recall. We declare a valid detection only if a circle is found in at least five consecutive frames and the estimated centers lie within a small tolerance of one another; this stabilizes the initialization and filters out spurious detections.

Figure 3: Initial phase of detection.

Adaptive Search Window. After a successful initialization, we switch to a focused search strategy because now we are almost certain that the detected circle corresponds to

the actual ball we are tracking. Let r be the radius estimated in the current frame. We center a square search window that has a side length that is

$$L = \max\big(\lfloor 0.6\, r \rfloor,\, 10\big)$$

more than r on the previous ball center, and rerun `HoughCircles` within this window, using now less restrictive parameters. Throughout the ball-tracking process, thanks to the previous lane detection, it is possible to estimate the ball's size at the beginning of the lane. This allows us to set more restrictive parameters for the `HoughCircles` detection. Additionally, once the ball's search area is narrowed, we can tighten the radius bounds for detection:

$$r_{\min} = \max\big(0, \lfloor 0.9\, r \rfloor\big), \quad r_{\max} = \lfloor 1.08\, r \rfloor.$$

This constraint dramatically reduces false positives while tracking the ball continuously.

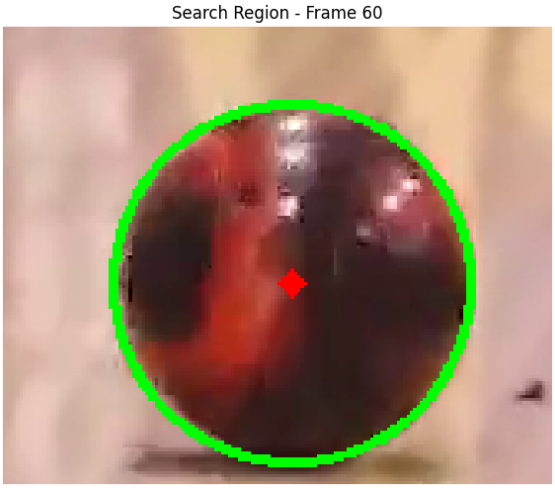

Figure 4: Adaptive search windows in action.

Re-initialization. Even with the narrowed search area, the probability of the ball being there is very high—but not absolute. It's also possible that circle detection might fail for several consecutive frames. This would result in the ball being lost in the area of interest for the rest of the video. To handle this issue, if the ball is not detected for two consecutive frames within the current search area, the search area is reset the detection mask to the full lane and restore the original, lenient `HoughCircles` parameters. Once the ball is re-detected in five consecutive frames, we reenter the adaptive window mode.

This cyclic process ensures robustness to occlusions, motion blur, and temporary loss. A ball detection trajectory obtained using this procedure is shown in the following graph.

To achieve the cleanest and most accurate ball detection possible, we incorporate and implement several post-processing techniques.

Radius outlier removal. The ball's radius generally follows an exponential trend, so we compute the exponential curve that best fits the data and remove any detections where the radius deviates by more than 10 pixels from this curve. Outliers in this case are rare due to the strict radius constraints in the ball detection phase.

Lane-referenced filtering. Since the camera may rotate, the obtained coordinates might not follow a clearly defined trajectory. For this reason, post-processing on the ball's position

is performed only after transforming the points using the homography of the reconstructed lane. After reconstructing the lane and mapping the points onto it (see following sections), the obtained coordinates now follow the actual ball trajectory, making post-processing much easier.

At this point, we remove all points whose coordinates are beyond the back line of the lane and those closer than 50cm from the foul line (we assume no ball touches the ground within 50cm of the foul line).

End-of-slide pruning, interpolation and smoothing. We remove any detections occurring after five consecutive points within 10cm of the back line (to eliminate false rebounds or pin reflections once the ball has struck the pins).

We finally apply a moving-median filter and a modified Z-score test (based on MAD) to eliminate residual outliers. We interpolate any missing positions, then apply a Savitzky–Golay filter to the trajectory to suppress high-frequency noise.

This pipeline yields a clean, continuous estimate of the ball's center. As shown in Figure 5, the trajectory data undergoes significant refinement through cleaning and interpolation.

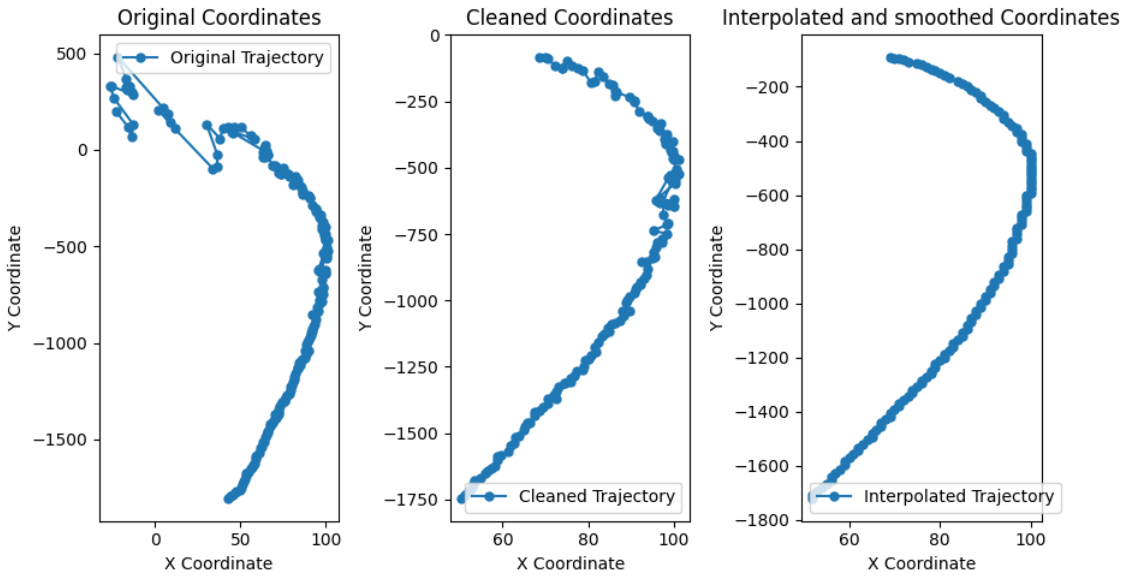

Figure 5: Post-processing stages of the detected ball center trajectory. The left plot shows the raw coordinates as captured, which may include noise or outliers. The middle plot displays the cleaned coordinates, where anomalies have been removed. The right plot presents the final interpolated and smoothed trajectory, suitable for further analysis.

3.3. Lane Reconstruction

After identifying the four corners of the lane in the input video, we compute the homography matrix that maps points from the video lane to points on a canonical, reconstructed lane.

Original lane reconstruction. The bowling lane has standardized dimensions: 1.0668m in width and 18.29m in lenght. Our goal is to map the four detected corner points in the video to the corresponding points on the real-world lane, defined as follows:

$$\text{Bottom-left} : [0, \text{height}]$$
$$\text{Bottom-right} : [\text{width}, \text{height}]$$
$$\text{Top-left} : [0, 0]$$
$$\text{Top-right} : [\text{width}, 0]$$

To compute this mapping, we use the `findHomography` function provided by the OpenCV library, which estimates the homography matrix between the two sets of points. Once this matrix is obtained, we can transform coordinates from the original video frame into the reference frame of the standardized lane.

Deformed lane reconstruction. Since the real lane is approximately 18 times longer than it is wide, directly visualizing the trajectory on a 1:1 scale would be not pleasant to look at. To improve clarity, and following the visual style commonly used in professional bowling competitions, we adopt a deformed lane representation with resized dimensions (e.g., 684x2340 pixels).

3.4. Trajectory

Trajectory on the Reconstructed Lanes. In order to visualize the ball's trajectory on the reconstructed lane, we must determine the contact point between the ball and the ground. Since the ball's radius and center are already known from previous detection steps, we estimate the ground contact point as the lowest point of the detected circle. By applying the computed homography to these adjusted coordinates, we obtain the ball's position on the canonical lane for each frame of the video.

To project the trajectory onto this deformed lane, we take the previously transformed ball positions and rescale them according to the new lane dimensions. This allows for a more compact and visually effective representation of the ball's movement.

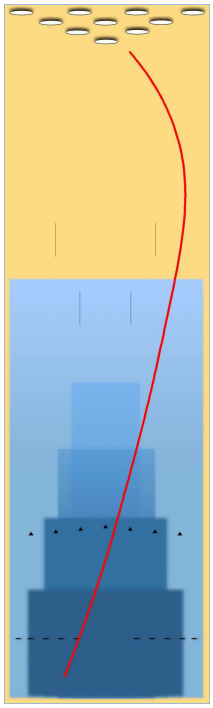

Figure 6: Trajectory on the reconstructed lane (deformed).

Trajectory on the Original Video. Visualizing the ball's trajectory directly on the video presents additional challenges. Simply plotting the contact points frame by frame is not sufficient: if the camera is rotating, those fixed points will appear to drift or stay static relative to the moving frame, resulting in an inaccurate representation of the trajectory.

To overcome this, we take advantage of the homography computed for each frame. Specifically, for every current frame, we apply the current frame's homography to all previously rectified ball contact points. In other words, at each frame, the displayed trajectory consists of all previous points transformed by the current homography. This ensures that the trajectory appears correctly aligned with the lane, even when the camera rotates, effectively "anchoring" the trajectory to the lane in the video.

Figure 7: Trajectory shown on video before the end.

3.5. Spin

Starting from the knowledge of the position and dimension of the ball in each frame is possible to compute the rotation of the ball. Due to the limits of the setup of the video (number of cameras and definition of the frames) it is required that the ball isn't monochromatic and has some color features.

Spin estimation. To estimate the rotational motion of the ball on a frame-by-frame basis, we computed the three-dimensional axis and angle of rotation between consecutive video frames by tracking visual surface features and applying the Kabsch algorithm to align their 3D positions.

First, from the data previously obtained during ball detection, we restricted the region of interest (ROI) in each frame to the area enclosed by the detected circular boundary of the ball. Within this ROI, we applied the Shi-Tomasi corner detection algorithm to extract salient feature points, which are more likely to be accurately trackable across successive frames.

The displacement of these feature points between two frames was then computed using the pyramidal Lucas-Kanade optical flow method [2] (see Appendix A). To ensure the reliability of the motion vectors, forward-backward error validation was performed. In this process, a feature tracked forward from frame t to $t+1$ is tracked backward, and only

features with minimal forward-backward displacement error and displacement magnitudes within a valid range (i.e., not too small or excessively large) were retained.

Each surviving 2D feature point $\mathbf{p} = (x, y)$ was lifted to a 3D point on the ball surface by assuming the ball is a perfect sphere of radius r. The z-coordinate was reconstructed using the equation of a sphere:

$$z = \sqrt{r^2 - (x - x_c)^2 - (y - y_c)^2}$$

where (x_c, y_c) is the center of the ball in the image frame. This allowed us to reconstruct two point clouds in R^3: $\{\mathbf{P}_i\}$ and $\{\mathbf{Q}_i\}$, corresponding to the positions of the same features in frames t and $t + 1$, respectively.

The optimal rotation between point sets is computed using the Kabsch algorithm [8] (see Appendix B).

The rotation angle θ is derived from the trace of the matrix:

$$\theta = \cos^{-1}\left(\frac{\mathrm{Tr}(\mathbf{R}) - 1}{2}\right)$$

The corresponding axis of rotation $\mathbf{a} = (a_x, a_y, a_z)$ is computed from the antisymmetric components of $\mathbf{R}$ as follows:

$$\mathbf{a} = \frac{1}{2\sin\theta}\begin{bmatrix} R_{32} - R_{23} \\ R_{13} - R_{31} \\ R_{21} - R_{12} \end{bmatrix}$$

If $\theta \approx 0$, a default axis (e.g., $\mathbf{a} = [0, 0, 1]$) was used to avoid numerical instability. After all this procedure, we have obtained the unit vector defining the rotation axis $\mathbf{a}$, and the angle of rotation θ (in radians). Following a frame in which we can see the the features extracted, the optical flow lines and the center of rotation obtained.

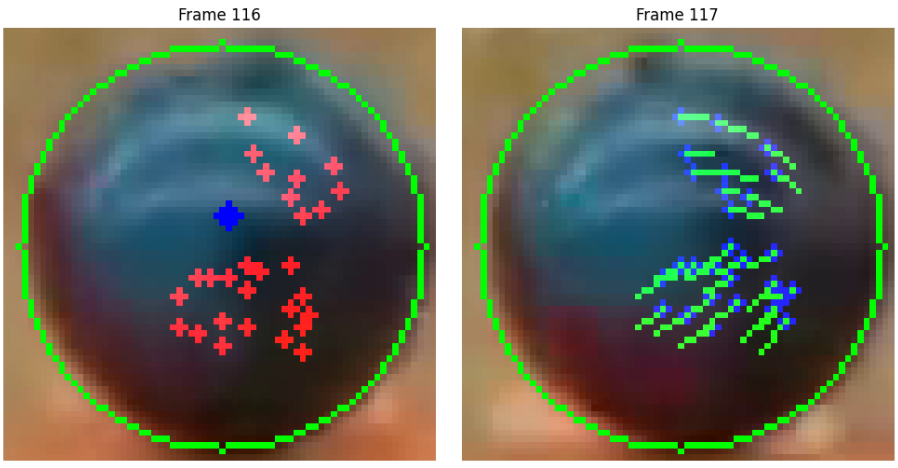

Figure 8: Frame in which the procedure to calculate the spin is running. On the left the blue point is the point around which the ball rotates and the red ones are the points extracted by the features. On the right the movements of the red points are showed.

Knowing the rotation angle between each couple of frames and the frame rate is trivial to compute the angular velocity of the ball around the axis:

$$\omega = \theta \cdot \mathrm{FPS}$$

Rotation axis post-processing. The data obtained for the spin estimation is significantly affected by noise, especially due to the low video quality and limited visibility of the ball in the final frames of the trajectory. For this reason, careful post-processing was necessary to ensure meaningful results.

Firstly, the estimation of the z-component of the rotation axis was ambiguous in sign: the same physical rotation can be represented using a positive or negative z-value. To resolve this, we computed the mean of all z-values. If the average was positive, we enforced consistency by multiplying all negative z-values by -1, and vice versa.

Secondly, based on the observation that the x- and y-components of the rotation axis do not change sign throughout a single throw (a condition verified across all analyzed videos), we computed the mean sign of the x and y components separately. Frames in which either component had the opposite sign were removed as outliers.

Additional filtering was applied to remove abrupt variations between frames. Since the x and y values of the rotation axis should follow a relatively smooth, approximately linear trajectory — generally with x increasing toward 1 and y approaching 0 — we fitted a line to the valid points and discarded any frame whose values deviated significantly from this fit.

After these steps, we applied interpolation and smoothing using a Gaussian filter to further suppress residual noise, especially in the final portion of the trajectory where the ball is often occluded or blurry.

Angular velocity post-processing. However, similar treatment was necessary for the angular velocity. These values, too, were highly sensitive to video quality and frequently included outliers. In this case, we identified outliers as values whose Z-score exceeded a predefined threshold. Once the outliers were removed, the angular velocity values were interpolated and smoothed to produce a continuous and noise-reduced curve.

This rigorous post-processing pipeline significantly improved the robustness and interpretability of the spin measurements across all analyzed bowling throws.

Video sphere realization. From the information about the axis and the angles of rotation it is possible to generate a realistic 3D animation of the evolution of the ball during the throw.

The approach involves creating a virtual sphere with a distinctive pattern to clearly visualize its rotation (in red the equator and the rotation centers) as shown in Figure 9. At each frame, the sphere is oriented by aligning its "north pole" with the given rotation axis extracted from the data. Then, the sphere is spun around this axis by the corresponding rotation angle.

Additionally, a red arrow is drawn to represent the instantaneous angular velocity, indicating both its magnitude and direction. The visualization is generated from a fixed viewpoint to maintain consistent perspective throughout the animation. The final output is a video showing the ball's rotation accurately reflecting the provided motion data, making it easier to analyze and interpret the dynamics of the throw.

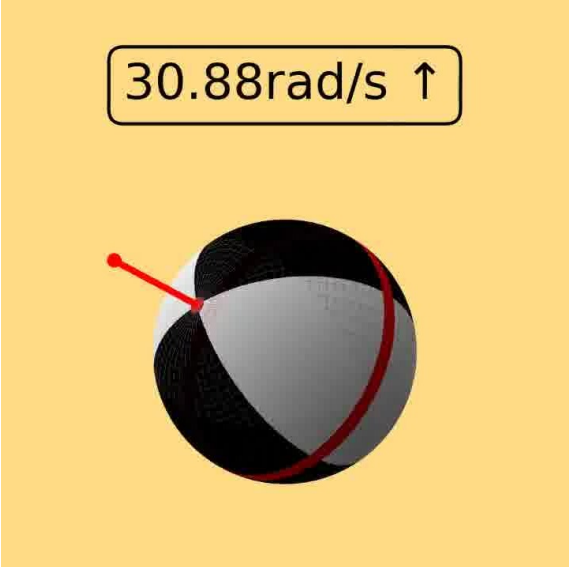

Figure 9: Live representation of the rotating ball.

4. Final Results

The automated pipeline is capable of processing a video of a bowling throw to execute all previously described steps, computing the ball's trajectory and rotation on a frame-by-frame basis. It produces multiple output videos, each highlighting a specific component of the analysis: lane detection, ball detection, reconstructed trajectory, overlayed trajectory on the original video, and rotational motion.

To enhance the interpretability of the results, a custom summary video is also generated. This video showcases the main outputs of the analysis (as illustrated in Figures 10 and 11). In both the original footage and the reconstructed lane view, the ball's trajectory is visualized in real time. Additionally, the upper-right corner displays a synthetic ball rotating in accordance with the extracted angular data, as described in Section 3.5.
The video configuration used in this work is unique within the context of bowling analysis, as it extends the traditional professional setup by incorporating information about the ball's rotation.

Figures 10 and 11 present four frames each, extracted from two different videos. Video 10 was recorded during a professional competition, while Video 11 features an amateur player. The primary difference between the two recordings lies in the camera setup: the former uses a more distant angle with digital zoom, which alters the perceived proportions of the lane. Notably, in Video 10, the camera moves during the shot and the bottom edge of the lane disappears in the last frames. It also compensates for the camera motion by dynamically adjusting the trajectory visualization to remain consistent with the movement of the lane. Despite this, the algorithm successfully performs the complete analysis with robustness and accuracy.

For a more comprehensive evaluation of the accuracy and quality of the output videos, readers are encouraged to consult the GitHub repository linked in the abstract.

A significant distinction between the two examples is the angular velocity of the ball. Table 1 reports the peak angular velocities computed for four test videos, alongside typical velocity ranges observed in professional and amateur players.

Bowler	Angular velocity
Video 1 - Professional	26 – 30 rad/s
Video 2 - Professional	20 – 22 rad/s
Video 3 - Amateur	11 – 13 rad/s
Video 4 - Amateur	14 – 17 rad/s
Generic Amateur Range	5 – 20 rad/s
Generic Professional Range	20 – 40 rad/s

Table 1: Comparison of peak angular velocities with typical ranges for amateur and professional players.

The results shown in Table 1 are consistent with expectations. Specifically, Videos 1 and 2 fall within the professional range, while Videos 3 and 4 align with the amateur range. This consistency validates the reliability of the angular velocity estimation performed by the pipeline.

Figure 10: Four frames describing Video 2.

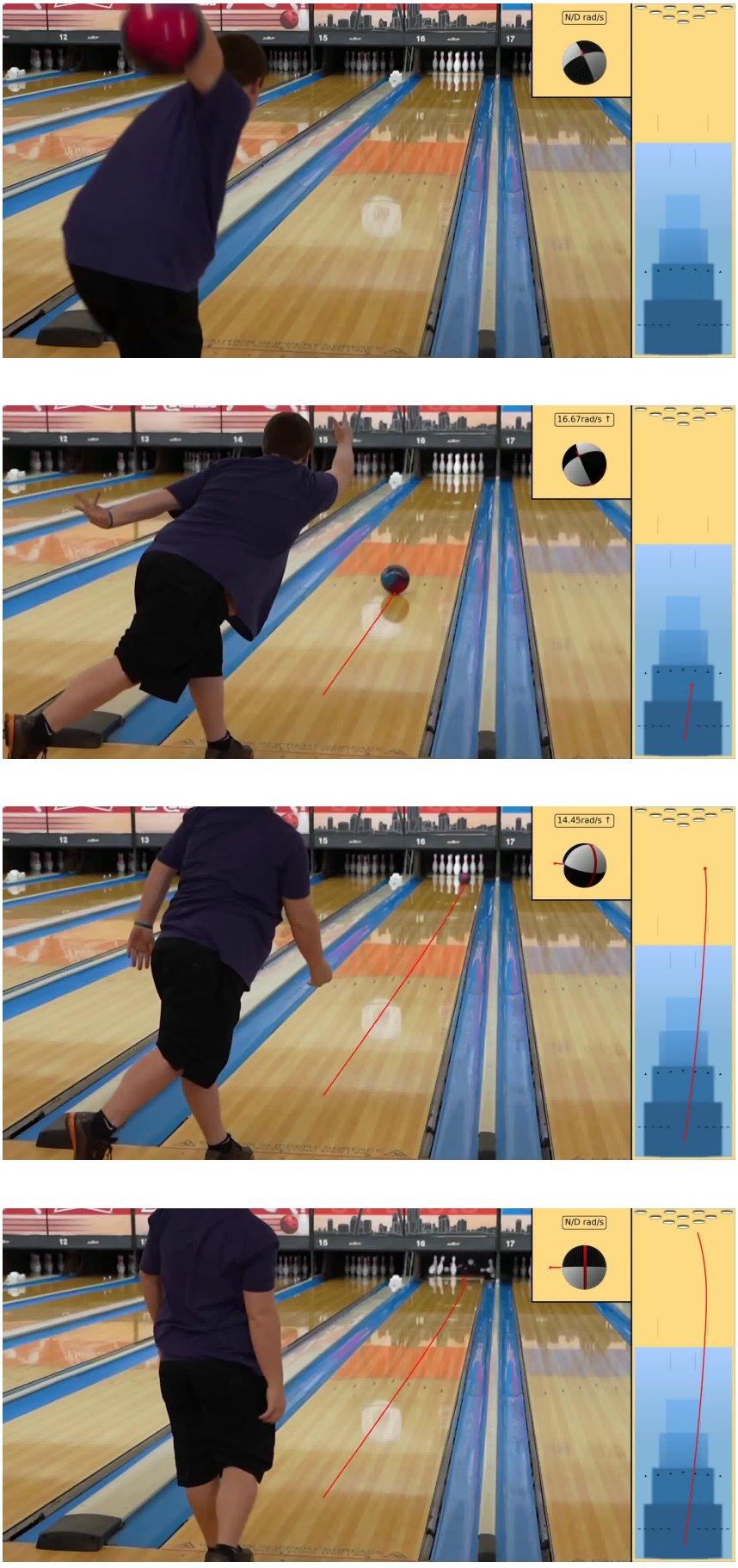

Figure 11: Four frames describing Video 4.

5. Conclusions

In this work, we have presented a complete pipeline for analyzing the trajectory and spin of a bowling ball from monocular video footage using classical computer vision techniques. Without relying on deep learning methods, our approach is robust to varying camera positions, resolutions, and even camera rotations.

The proposed system is capable of detecting the bowling lane through line detection and template matching, identifying the ball using the Hough Circle Transform, reconstructing a canonical view of the lane via homography, and estimating both the trajectory and the rotational behavior of the ball. For spin analysis, we adopted optical flow techniques and developed a thorough post-processing procedure to mitigate the high level of noise due to video quality and motion blur. The use of homography not only enabled accurate lane reconstruction but also made it possible to align ball position data across frames consistently, even under dynamic camera conditions.

This project has provided valuable insights into the practical challenges of real-world video analysis, especially in constrained domains like sports analytics. It has reinforced the importance of geometric modeling, robust estimation techniques, and the role of post-processing in refining noisy data.

Several directions remain open for future work. A potential improvement to our work is the development of a specific low-cost tool—such as a website or a mobile app—that enables anyone with a simple phone or camera to analyze their bowling throws easily. To achieve this, it would be valuable to adapt our existing pipeline by incorporating certain assumptions that translate into clear instructions for the user, for example, maintaining a still camera and specifying an optimal camera position during recording.

Another promising avenue would be the integration of deep learning-based object tracking or segmentation techniques to improve ball detection robustness, particularly in scenes with occlusions or poor lighting. Additionally, extending the system to handle multi-view setups could improve 3D accuracy of both trajectory and spin estimation. Finally, integrating the system into real-time coaching tools or automated scoring platforms would represent a significant step toward practical deployment in professional or amateur bowling environments.

Overall, the project lays a solid foundation for future research and development in sports analytics using traditional computer vision methodologies.

Appendix

A. Pyramidal Lucas-Kanade Optical Flow

To estimate the motion of features on the surface of the bowling ball between video frames, the **pyramidal Lucas-Kanade (LK) optical flow** method was employed. This algorithm allows for tracking feature points with sub-pixel accuracy even in the presence of large displacements, by leveraging a coarse-to-fine approach via image pyramids.

Background. The classical Lucas-Kanade method assumes that the motion between two images is small and approximately constant within a local window around each feature. It solves for the optical flow vector $\mathbf{v} = [u, v]^T$ that satisfies the brightness constancy constraint:

$$I(x + u, y + v, t + 1) = I(x, y, t)$$

Using a first-order Taylor approximation and a small window around each pixel, this leads to an overdetermined system which is solved in the least-squares sense.

However, this method breaks down under large motions, which are common in high-speed bowling sequences. To overcome this, the **pyramidal extension** is used.

Pyramidal Approach. The pyramidal Lucas-Kanade algorithm constructs a Gaussian pyramid of the image frames — each level is a downsampled, lower-resolution version of the original. Optical flow is computed at the coarsest level (lowest resolution) and then iteratively refined at each higher-resolution level.

The steps are:

1. Build image pyramids for frames I_t and I_{t+1}.
2. Initialize flow $\mathbf{v}_L = 0$ at the coarsest level L.
3. For each pyramid level from coarse to fine:
   (a) Warp the second image using the current estimate of the flow.
   (b) Apply the standard Lucas-Kanade optical flow to refine the motion estimate.
   (c) Upscale the flow to the next finer level.

Application to Spin Estimation. In this project, the pyramidal LK method was used to track surface features of the bowling ball across video frames. The 2D points of these features were projected into 3D space. The resulting 3D point served as inputs to the Kabsch algorithm (Appendix B), which computes the rigid-body rotation between frames, enabling accurate estimation of the spin axis and angular velocity of the ball.

The use of a pyramidal approach ensured robustness to fast rotational motion and partial occlusions, both of which are common in realistic bowling sequences.

B. Kabsch Algorithm for Optimal Rigid Alignment

In the process of estimating the spin of a bowling throw, it is necessary to determine the rotational motion of the ball between successive time steps. Given a set of 3D points tracked on the ball's surface (e.g., via visual markers or texture features), the **Kabsch algorithm** is employed to compute the optimal rotation that aligns two corresponding sets of points, minimizing the root mean square deviation (RMSD) between them.

Purpose. The Kabsch algorithm provides the optimal *rigid rotation matrix* $R \in R^{3\times3}$ that aligns two sets of points $P = \{\mathbf{p}_i\}$ and $Q = \{\mathbf{q}_i\}$, corresponding to the same features observed at two different time frames. This is crucial for estimating the angular displacement and, subsequently, the *spin axis and angular velocity* of the ball.

Mathematical Formulation. Given two sets of N corresponding 3D points P and Q, the goal is to find a rotation R that minimizes:

$$\text{RMSD}(R) = \sqrt{\frac{1}{N}\sum_{i=1}^{N} \|R\mathbf{p}_i - \mathbf{q}_i\|^2}$$

The steps of the Kabsch algorithm are as follows:

1. Compute and subtract the centroids of P and Q to center the point clouds:

$$\bar{\mathbf{p}} = \frac{1}{N}\sum_{i=1}^{N}\mathbf{p}_i, \quad \bar{\mathbf{q}} = \frac{1}{N}\sum_{i=1}^{N}\mathbf{q}_i$$

$$\mathbf{p}'_i = \mathbf{p}_i - \bar{\mathbf{p}}, \quad \mathbf{q}'_i = \mathbf{q}_i - \bar{\mathbf{q}}$$

2. Compute the covariance matrix H:

$$H = P'^{\top}Q'$$

3. Perform SVD on H:

$$H = U\Sigma V^{\top}$$

4. Compute the optimal rotation matrix:

$$R = VU^{\top}$$

   If $\det(R) < 0$, indicating a reflection, correct it by negating the last column of V:

$$V' = V, \quad V'_{[:,3]} := -V_{[:,3]}, \quad R = V'U^{\top}$$

Application to Spin Estimation. In our context, the rotation matrix R obtained between two frames allows for the extraction of the *axis-angle representation* of the motion. From this, the *angular velocity vector* $\boldsymbol{\omega}$ and the *spin axis* can be determined. This method assumes rigid body motion and is robust to noise in feature positions, making it particularly suitable for analyzing high-speed ball rotation from 3D point trajectories. It is important to note that the algorithm operates under the assumption that the points extracted by the Optical Flow algorithm are coherent: the tracked features are well taken and consistently correspond to the same physical points across frames.

References

[1] G. Boracchi, V. Caglioti, and A. Giusti. Single-image 3d reconstruction of ball velocity and spin from motion blur: An experiment in motion-from-blur. 2008.

[2] Jean-Yves Bouguet. Pyramidal implementation of the lucas kanade feature tracker: Description of the algorithm. Technical Report OpenCV Tech Report, Intel Corporation, Microprocessor Research Labs, 2001.

[3] G. Bradski and A. Kaehler. *Learning OpenCV: Computer Vision with the OpenCV Library*. O'Reilly Media, 2008.

[4] A. Dosovitskiy, P. Fischer, and E. et al. Ilg. Flownet: Learning optical flow with convolutional networks. In *Proc. IEEE Int. Conf. on Computer Vision*, pages 2758–2766, 2015.

[5] R. O. Duda and P. E. Hart. Use of the hough transformation to detect lines and curves in pictures. *Communications of the ACM*, 15(1):11–15, 1972.

[6] T. Gossard, J. Krismer, A. Ziegler, J. Tebbe, and A. Zell. Table tennis ball spin estimation with an event camera. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, 2024.

[7] Hawk-Eye Innovations. Hawk-eye: A global leader in the live sports arena, 2023. `https://www.hawkeyeinnovations.com/`.

[8] Wolfgang Kabsch. A solution for the best rotation to relate two sets of vectors. *Acta Crystallographica Section A: Crystal Physics, Diffraction, Theoretical and General Crystallography*, 32(5):922–923, 1976.

[9] N. Kiryati, Y. Eldar, and A. M. Bruckstein. A probabilistic hough transform. *Pattern Recognition*, 24(4):303–316, 1991.

[10] K. Park and Y. Seo. Probabilistic tracking of the soccer ball. In *Proceedings of the International Conference on Image Processing (ICIP)*, pages 50–60, 2004.

[11] Roboflow. Ball tracking in sports with computer vision, 2024. `https://blog.roboflow.com/tracking-ball-sports-computer-vision/`.

[12] H. P. H. Shum and T. Komura. Tracking the translational and rotational movement of the ball using high-speed camera movies. In *Proceedings of the IEEE International Conference on Image Processing (ICIP)*, pages III–1084, 2005.

[13] T. Tamaki, T. Sugino, and M. Yamamoto. Measuring ball spin by image registration. In *Proceedings of the 10th Korea-Japan Joint Workshop on Frontiers of Computer Vision (FCV)*, pages 269–274, 2004.

[14] G. Van Zandycke and C. De Vleeschouwer. 3d ball localization from a single calibrated image. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, pages 483–491, 2022.

[15] J. Širmenis and M. Lukoševičius. Tracking basketball shots – preliminary results. In *CEUR Workshop Proceedings*, volume 2915, pages 45–57, 2021.

[16] X. Yu, C.-H. Sim, J. R. Wang, and L. F. Cheong. A trajectory-based ball detection and tracking algorithm in broadcast tennis video. In *Proceedings of the IEEE International Conference on Image Processing (ICIP)*, pages 1049–1052, 2004.

[17] Y. Zhang, Z. Tu, and F. Lyu. A review of lane detection based on deep learning methods. *Mechanical Engineering Science*, 1507, 2024.