3D Bowling Ball Trajectory And Spin Analysis

Filippo Riva (10714936) and Rossi Christian (10736464)

Politecnico di Milano

Course by prof. Caglioti Vincenzo

Abstract—This project focuses on the detection and analysis of a bowling ball in motion on a bowling lane. The main objectives are to accurately track the ball's trajectory, as well as to estimate its spin and the orientation of its rotation axis over time.
The detection and initial localization of the ball are performed using an advanced object detection neural network, YOLOv8-large, which allows for real-time and robust identification even under varying lighting conditions and complex lane backgrounds. Following detection, the 3D positioning of the ball is reconstructed using geometric transformations based on the camera calibration parameters and lane measurements. To analyze the rotational behavior, optical flow techniques are applied to consecutive frames to estimate angular velocity, spin rate, and the rotation axis of the ball.

Contents

1. Introduction

This section introduces the problem and outlines the methodology used to analyze its trajectory and rotational behavior.
The overall procedure can be divided into several key stages:

- *Intrinsic calibration*: the camera's intrinsic parameters are determined using a checkerboard pattern, enabling accurate mapping between the 2D image plane and 3D space.
- *Video synchronization*: two video sources are synchronized using audio correlation techniques to ensure temporal alignment across frames.
- *Video undistortion*: lens distortions are corrected using the previously computed intrinsic parameters, providing a geometrically accurate view of the lane.
- *Lane detection*: the lane edges are manually identified to establish a reference frame for subsequent ball localization.
- *Extrinsic calibration*: the camera's position and orientation relative to the lane are computed from the lane geometry and intrinsic calibration, allowing the projection of 2D detections into 3D space.
- *Ball tracking*: the ball is detected in each frame using YOLO, and missing detections are estimated through temporal extrapolation.
- *Ball localization*: detected 2D positions are transformed into 3D coordinates using the extrinsic calibration.
- *Ball rotation estimation*: optical flow and frame-to-frame analysis are applied to estimate the ball's spin and the orientation of its rotation axis.

1.1. Experimental Setup

The experimental setup involved recording the ball's motion along a standard bowling lane using two cameras. Specifically, we used a Nothing Phone 2A and an iPhone 14, positioned at the bottom-left and bottom-right corners of the lane, respectively.

The Nothing Phone 2A features a 50 MP main camera with optical image stabilization, capable of capturing high-resolution videos with good low-light performance. The iPhone 14 is equipped with a 12 MP main camera with sensor-shift optical image stabilization, which provides excellent dynamic range and color accuracy.

We recorded 1920x1080 30 fps videos from both camera simultaneously to capture different perspectives of the ball's trajectory. At the end we got seven videos. The video number two has been included in the report to visually explain the results.

2. State of the art

This section reviews the current approaches in object detection and tracking, with a focus on deep neural networks and the YOLO architecture, which are relevant for tracking a bowling ball and analyzing its motion.

Object detection and tracking are fundamental tasks in computer vision with numerous applications, ranging from autonomous vehicles to sports analytics. Traditional approaches relied on handcrafted features and classical algorithms. While these methods provided baseline solutions, they often struggled with complex backgrounds, varying lighting conditions, and real-time processing requirements.

The advent of deep neural networks revolutionized object detection by enabling end-to-end learning of features directly from raw images. Convolutional Neural Networks form the backbone of most modern detectors, allowing robust detection even in challenging environments. State-of-the-art detectors, such as YOLO, offer a balance between accuracy and computational efficiency.

YOLO is particularly well-suited for tasks requiring fast and continuous tracking of objects in video sequences. Unlike region proposal-based methods, YOLO treats detection as a single regression problem, predicting bounding boxes and class probabilities simultaneously. Recent iterations, such as YOLOv8, have improved both accuracy and speed through architectural enhancements and advanced training techniques.

To analyze the rotation and spin of moving objects, optical flow techniques are widely used. Optical flow estimates the motion of features between consecutive frames by calculating pixel-level displacements, allowing the extraction of velocity vectors and rotation axes. Methods such as the Lucas-Kanade algorithm or dense optical flow enable precise tracking of small, fast-moving objects, complementing bounding box detections from YOLO. Combining object detection with optical flow provides a powerful framework for both spatial localization and temporal motion analysis.

3. Solution

This section explains all the processing steps of the pipeline created to extract both the motion and rotation information of the bowling ball. Each subsection describes in detail the methods and techniques used to obtain the required data, from video acquisition to final rotation estimation.

3.1. Intrinsic calibration

Accurate camera calibration is a foundational prerequisite for precise 3D localization. This process estimates the internal geometric and optical characteristics of each camera, which are encapsulated in a set of intrinsic parameters and distortion coefficients.

The objective is to determine the calibration matrix, $\mathbf{K}$, which models the projective transformation from 3D camera coordinates to 2D image coordinates, and a set of distortion coefficients, $\mathbf{D}$, which correct for lens-induced modifications. The calibration matrix is defined as:

$$\mathbf{K} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \tag{1}$$

Here, f_x and f_y represent the focal lengths along their respective axes, and (c_x, c_y) is the principal point.

The distortion vector is defined as:

$$\mathbf{D} = \begin{bmatrix} k_1 & k_2 & p_1 & p_2 & k_3 \end{bmatrix} \tag{2}$$

Here, we have modeled both radial (k_1, k_2, k_3) and tangential (p_1, p_2) effects.

The calibration is performed using a known planar target with a checkerboard pattern. The world coordinates of the target's corners are defined on a fixed plane ($Z = 0$), establishing a consistent coordinate frame:

$$\mathbf{X}_w = \begin{bmatrix} i \\ j \\ 0 \end{bmatrix} \qquad i \in [0, M-1] \quad j \in [0, N-1] \tag{3}$$

Here, $M \times N$ is the size of the checkerboard grid.

We have capture for each camera both still images and frames systematically sampled from video sequences. To guarantee high-quality input for the optimization, each frame is preprocessed to enhance contrast and normalize illumination. The corners of the checkerboard are then detected and refined to sub-pixel accuracy for maximum precision.

The core of the calibration is a non-linear optimization that minimizes the re-projection error. This error quantifies the discrepancy between the observed corner locations in the image and their theoretical projections based on the current parameter estimates:

$$\text{error} = \sum_{i=1}^{N} \left| \mathbf{x}_i - \hat{\mathbf{x}}_i \right|^2 \tag{4}$$

Here, $\mathbf{x}_i$ is an observed image point and $\hat{\mathbf{x}}_i$ is its projection computed using the estimated $\mathbf{K}$ and $\mathbf{D}$. Upon successful convergence of this optimization for a given camera, the resulting parameters are saved.

Results The intrinsic calibration procedure successfully estimated the camera parameters for both cameras. The resulting camera matrices ($\mathbf{K}$) and distortion coefficients ($\mathbf{D}$) are presented below:

$$\mathbf{K}_1 = \begin{bmatrix} 1129.70180 & 0 & 336.94648 \\ 0 & 1130.08987 & 638.29658 \\ 0 & 0 & 1 \end{bmatrix}$$

$$\mathbf{D}_1 = \begin{bmatrix} -0.00095 & 0.06965 & -0.00367 & -0.00401 & -0.08137 \end{bmatrix}$$

$$\mathbf{K}_2 = \begin{bmatrix} 1773.31033 & 0 & 544.90433 \\ 0 & 1773.68022 & 955.41251 \\ 0 & 0 & 1 \end{bmatrix}$$

$$\mathbf{D}_2 = \begin{bmatrix} 0.33583 & -2.11736 & -0.00277 & -0.00157 & 4.34057 \end{bmatrix}$$

The calibration was successful for both cameras since the parameters are physically plausible and well-conditioned. The key difference lies in the lens characteristics: the first uses a lens with minimal distortion, while the second one uses a wider-angle lens with strong barrel distortion.

3.2. Video synchronization

To facilitate accurate three-dimensional triangulation of the bowling ball's trajectory, the scene was captured using a stereoscopic camera setup. A critical prerequisite for this process is the precise temporal alignment of the two independent video streams.

The synchronization methodology leverages the audio tracks embedded within each video file as a common, time-varying signal. The first step involves calculating the temporal offset, Δt, between these audio streams. This is achieved by computing their cross-correlation, a standard signal processing technique for estimating the time delay between two signals. The lag corresponding to the maximum correlation value is identified and converted into a time offset, indicating whether one video leads or lags behind the other.

Following the determination of Δt, a common temporal baseline is established for both videos. The target frame rate is conservatively set to the lower of the two original frame rates to guarantee the integrity of every sampled frame. Similarly, the target duration is set to the shorter of the two video durations to ensure a valid frame exists in both streams for every point in time.

The final synchronized videos are generated by applying the calculated offset and resampling the visual data. The video that was found to be ahead is delayed by Δt, while the other begins playback from its natural start. Both streams are then sampled at the common target frame rate across the shared temporal window. This procedure ensures that for any discrete time index t within the synchronized sequence, the corresponding frames from both cameras depict the exact same moment in time.

3.3. Video undistortion

Following temporal synchronization, the video data undergoes a geometric correction process to rectify lens-induced distortions. The raw footage from each camera is subject to imperfections inherent in optical systems, primarily radial and tangential distortion, which cause straight lines in the real world to appear curved in the captured images. Correcting these modifications is a prerequisite for any subsequent quantitative analysis, as it ensures that the image geometry accurately reflects the scene's true projective transformation.

The correction is applied on a per-camera basis, utilizing the unique intrinsic calibration parameters previously determined for each lens. For every frame in a video stream, a pixel-wise transformation is computed. This transformation maps the coordinates of each pixel in the distorted image to their correct, undistorted location in a geometrically ideal image plane.

To maximize the clarity and usability of the corrected footage, an optimized intrinsic camera matrix $\mathbf{K}_{\text{new}}$ is calculated. This matrix is designed to crop the resulting image, eliminating the undefined black border regions that typically appear after distortion removal, thus producing a clean, fully-defined output frame. The entire sequence of corrected frames is then re-encoded into a new video file.

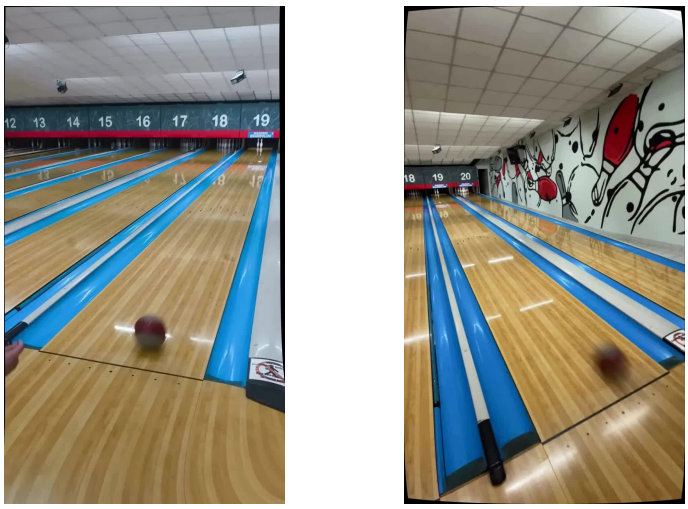
This process effectively generates a new set of videos where the imaging characteristics of each camera have been normalized to an ideal pinhole model.

Results The undistortion resulting frame is shown in 1.

3.4. Lane detection

To establish a spatial reference frame within the scene, a set of canonical lane features must be identified in the image plane of each camera. This process involves manually annotating the two-dimensional image coordinates of four key corner points that define the lane's boundaries.

The annotation is performed on a single, representative frame extracted from the midpoint of each video sequence. This frame is



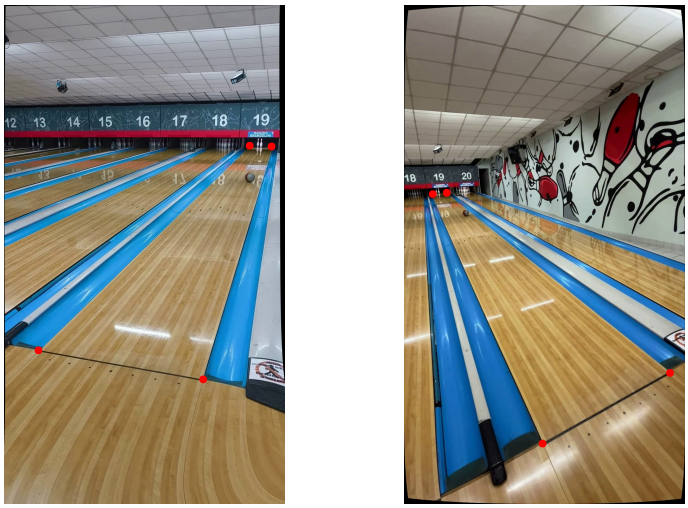
(a) Nothing undistorted **(b)** iPhone undistorted

Figure 1. Undistorted initial frame of the video two.

selected under the assumption that the camera remains static throughout the recording, ensuring the spatial consistency of the selected features. The four points to be identified are the intersections of the lane boundaries, corresponding to the bottom-left, top-left, top-right, and bottom-right corners of the lane's visible trapezoid.

The user then sequentially selects the four corner points via mouse clicks and then the image coordinate are automatically extracted. Once annotated for all camera views, the complete set of image-space lane coordinates is saved

Results The lane detection result is shown in 2.



(a) Nothing lane detection **(b)** iPhone lane detection

Figure 2. Middle frame of the video two with the four lane corners identified.

3.5. Extrinsic calibration

The extrinsic calibration process determines the position and orientation of each camera within a predefined world coordinate system. This transformation is essential for relating points in the three-dimensional world to their two-dimensional projections in each camera's image plane, enabling accurate triangulation and 3D reconstruction.

The calibration leverages a set of known correspondences between 3D points in the world frame, $\mathbf{X}_i$, and their observed 2D image coordinates, $\mathbf{x}_i$. For this application, the 3D points are the precisely measured corners of the bowling lane. The relationship between a world point and its image projection is governed by the pinhole camera model, which incorporates the previously determined intrinsic parameters, $\mathbf{K}_i$, and the unknown extrinsic parameters, the rotation $\mathbf{R}_i$ and translation $\mathbf{t}_i$ for each camera i:

$$s \begin{bmatrix} u_i \\ v_i \\ 1 \end{bmatrix} = \mathbf{K}_i \begin{bmatrix} \mathbf{R}_i & \mathbf{t}_i \end{bmatrix} \begin{bmatrix} X_i \\ Y_i \\ Z_i \\ 1 \end{bmatrix} \tag{5}$$

Here, s is a scale factor.

The Perspective-n-Point (PnP) [1] algorithm is employed to solve for these unknown parameters. Given a sufficient number of 3D-2D correspondences and the camera's intrinsic matrix, PnP computes the rotation and translation that minimize the re-projection error, ensuring the projected 3D points align with their observed 2D locations. Since we have a set of four co-planar point we have used Infinitesimal Plane-based Pose Estimation (IPPE) [2] solver to get the best possible results.

The solution is initially returned as a rotation vector, which is subsequently converted into a standard 3×3 rotation matrix, $\mathbf{R}_i$, using the Rodrigues' rotation formula [3]. The camera's position, or center, in world coordinates is then derived from the extrinsic parameters as:

$$\mathbf{C}_i = -\mathbf{R}_i^\top \mathbf{t}_i \tag{6}$$

The complete transformation from world to camera coordinates is encapsulated in the projection matrix $\mathbf{P}_i$.

$$\mathbf{P}_i = \mathbf{K}_i \begin{bmatrix} \mathbf{R}_i & \mathbf{t}_i \end{bmatrix} \tag{7}$$

Results For the position and the rotation we have obtained the following results:

$$\mathbf{t}_1 = \begin{bmatrix} -1.27867 \\ -1.60034 \\ 1.63000 \end{bmatrix} \quad \mathbf{R}_1 = \begin{bmatrix} 0.54573 & -0.11911 & 0.82945 \\ -0.79937 & 0.22293 & 0.55795 \\ -0.25137 & -0.96753 & 0.02645 \end{bmatrix}$$

$$\mathbf{t}_2 = \begin{bmatrix} -2.86383 \\ 0.01390 \\ 1.63000 \end{bmatrix} \quad \mathbf{R}_2 = \begin{bmatrix} -0.27832 & -0.19923 & 0.93960 \\ -0.89271 & 0.41462 & -0.17652 \\ -0.35440 & -0.88792 & -0.29325 \end{bmatrix}$$

The position of the camera with respect to the bowling lane is shown in Figure 3.

The positions and orientations of the cameras are computed correctly with respect to the real setup.

3.6. Ball tracking

The ball tracking process isolates and localizes the two-dimensional trajectory of the bowling ball within each video stream. This is achieved through a combination of a deep learning-based object detector and a targeted search strategy to ensure computational efficiency and robustness across the sequence.

A pretrained YOLO [4] (You Only Look Once) neural network model serves as the core detector, configured to identify objects belonging specifically to the "sports ball" class. For each frame, the model processes the image and returns a set of bounding boxes, $\mathbf{B}_v^\top = (x_{\min}, y_{\min}, x_{\max}, y_{\max})$, defining the axis-aligned rectangle enclosing any detected ball in view v at time t.

To optimize performance, a dynamic region of interest (ROI) is employed. After an initial detection on the full frame, subsequent searches are confined to a cropped region around the ball's last known position. The coordinates of any detection within this cropped region are then rescaled and translated back to the original image's coordinate system. The ball's centroid (c_x, c_y) and an approximate radius r are derived from the bounding box, providing a simplified circular representation of its position and size.

This sequence of centers and radii forms a discrete 2D trajectory for the camera view. To account for occasional detection failures, a

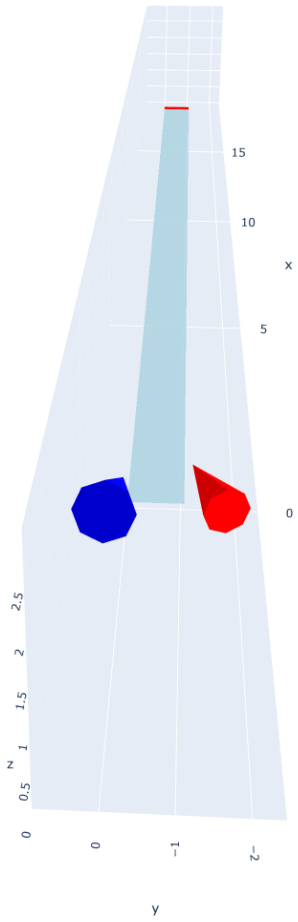


Figure 3. View of the three-dimensional position of the cameras with respect to the bowling lane for the video two.

post-tracking interpolation algorithm is applied to infer the ball's state in frames where it was not detected. The interpolation of missing data is treated as two distinct problems:

1. *Radius interpolation*: the apparent size of the ball in the image is inversely proportional to its distance from the camera due to perspective effects. To model this, a linear fit is applied to the inverse of the radius over a sliding window of the last W_r known frames.
2. *Centers interpolation*: the 2D motion of the ball's centroid is modeled as two independent linear functions of time (frame index). For a missing frame i, the x and y coordinates are estimated by performing a linear regression on the last W_c known points.

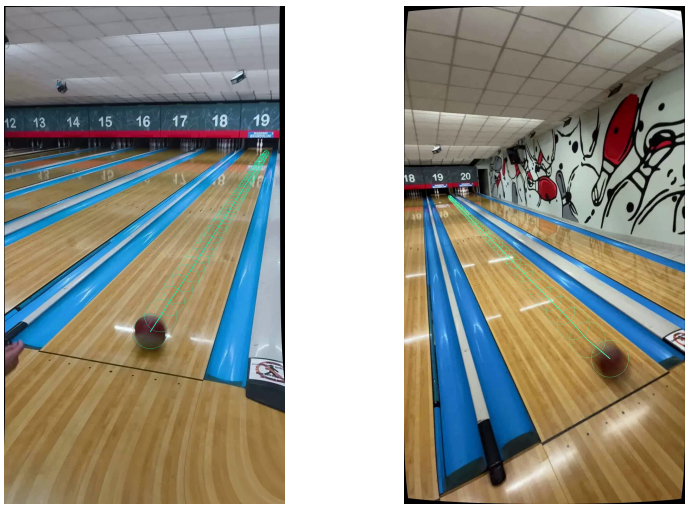
This two-stage interpolation results in a complete and continuous path. The final output for each view is a temporally ordered set of image coordinates and associated radii, representing the ball's motion through the image plane over time.

Results The result of the ball tracking is shown in 4.

3.7. World localization

The world localization process reconstructs the three-dimensional trajectory of the bowling ball by leveraging the principles of stereoscopy and the calibrated camera system. This procedure transforms the synchronized two-dimensional image-plane trajectories from each camera into a single, coherent path in world coordinates.

The foundation of this reconstruction is the triangulation of corresponding points. For each time instance t, the detected ball center in the first camera, $\mathbf{c}_1^\top = (u_1^\top, v_1^\top)$, and its counterpart in the second camera, $\mathbf{c}_2^\top = (u_2^\top, v_2^\top)$, define a pair of lines of sight. These lines are



(a) Nothing ball tracking **(b)** iPhone ball tracking

Figure 4. Initial frame with all the tracked centers and radii for the video two.

projected back into the 3D world using the inverse of each camera's projection matrix $\mathbf{P}_i$:

$$\mathbf{P}_i = \mathbf{K}_i \begin{bmatrix} \mathbf{R}_i & \mathbf{t}_i \end{bmatrix} \tag{8}$$

Here, $\mathbf{K}_i$ is the intrinsic matrix and $\begin{bmatrix} \mathbf{R}_i & \mathbf{t}_i \end{bmatrix}$ is the extrinsic matrix for camera i.

The 3D position of the ball, $\mathbf{X}^\top = (X^\top, Y^\top, Z^\top)$, is estimated as the point of closest approach between these two projected rays via triangulation with the Direct Linear Transform algorithm.

Results The three-dimensional trajectory of ball with respect to the bowling lane is shown in Figure 5.

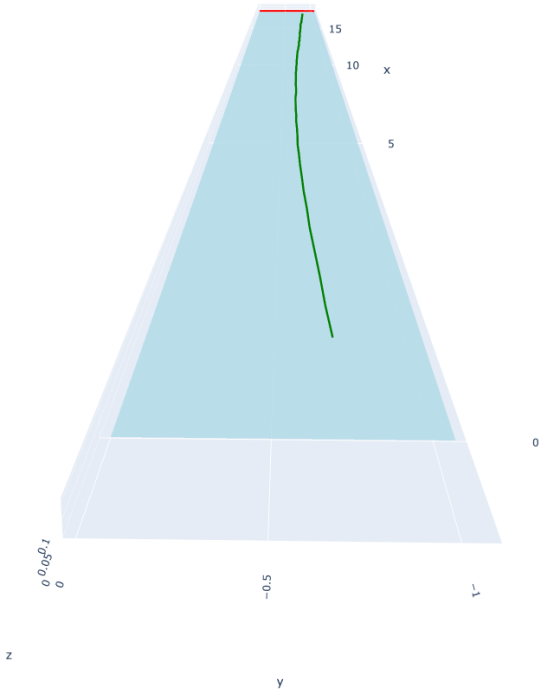


Figure 5. View of the three-dimensional trajectory of the ball on the bowling lane for the video two.

The resulting trajectory is coherent with the real one of the video two.

3.8. Ball rotation

This final step estimates the spin, or rotational velocity, of a bowling ball by analyzing the motion of visual features on its surface across consecutive video frames. The method leverages optical flow [5] to track these features and computes a weighted angular displacement in 3D space to derive the spin rate.

The process is executed for each camera view independently and consists of the following steps:

1. *Feature detection and tracking*: for each frame, the known ball position from the trajectory is used to define a region of interest. Within this region, distinctive features (corners) are detected using the Shi-Tomasi corner detector algorithm [6]. These features are then tracked to the next frame using the Lucas-Kanade optical flow method [7], which assumes constant velocity in a local neighborhood. The optical flow provides a displacement vector for each feature between frames t and $t+1$.
2. *3D motion field analysis*: the collective motion of the features is analyzed to infer the 3D rotation axis. For each successfully tracked feature, its 2D position is projected onto the 3D surface of the ball using the known ball radius:

$$d = \sqrt{(p_x - c_x)^2 + (p_y - c_y)^2}$$
$$z = \sqrt{\max(r^2 - d^2, 0)}$$

   Here, (p_x, p_y) is the feature position, (c_x, c_y) is the ball center, and r is the ball radius. The 3D position vectors $\vec{r}_t$ and $\vec{r}_{t+1}$ are constructed for each feature at times t and $t+1$.
3. *3D rotation axis estimation*: for each feature pair, the rotation axis direction is estimated using the cross product of its 3D position vectors:

$$\vec{\text{axis}}_i = \frac{\vec{r}_t \times \vec{r}_{t+1}}{\|\vec{r}_t \times \vec{r}_{t+1}\|} \tag{9}$$

   The average 3D rotation axis for the frame is computed as the normalized mean of all individual axis vectors.
4. *Angular displacement calculation*: for each feature, the angular displacement $\Delta\theta$ is calculated using the dot product of its normalized 3D position vectors:

$$\Delta\theta = \arccos\left(\frac{\vec{r}_t \cdot \vec{r}_{t+1}}{\|\vec{r}_t\| \cdot \|\vec{r}_{t+1}\|}\right) \tag{10}$$

   This yields the true angular change of the feature on the sphere's surface.
5. *Weighted spin rate estimation*: not all features contribute equally to the spin estimate. Features closer to the ball's center have a smaller linear velocity for the same angular velocity and are more susceptible to noise. Therefore, a weighted average is used, where the weight w_i for the i-th feature is proportional to its radial distance r_i from the center (normalized by the ball's radius r):

$$w_i = \frac{\|\vec{r}_i\|}{r} \tag{11}$$

   Features too close to the center ($< 0.3r$) are discarded as unreliable. The weighted average angular displacement $\overline{\Delta\theta}$ for the frame is calculated. This value is then converted to an instantaneous spin rate ω in radians per second by multiplying by the video frame rate fps:

$$\omega_{\text{i}} = \overline{\Delta\theta} \times \text{fps} \tag{12}$$

   Finally, a simple exponential smoothing filter with factor α is applied to the time series of spin rates to reduce jitter and produce a smooth, stable output:

$$\omega_{\text{s}}[t] = \alpha \cdot \omega_{\text{i}}[t] + (1 - \alpha) \cdot \omega_{\text{s}}[t-1] \tag{13}$$

The algorithm maintains a pool of tracked features, merging successfully tracked features with newly detected ones each frame. Features that move outside the ball boundary are automatically removed. When no features are tracked or the ball is not detected, the last known spin rate is maintained for continuity.

Results The estimated angular speed is presented in Figure 6. The

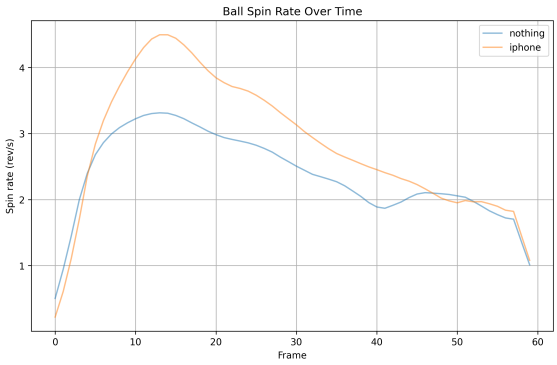


Figure 6. Angular speed of the two balls for the video two.

angular speed estimates are consistent across both videos. However, the values obtained from the iPhone video are consistently slightly higher than those from the Nothing video. This discrepancy arises because the ball in the iPhone video appears blurred in several frames, making feature tracking more challenging and slightly less accurate.

The estimated three-dimensional rotation axis over time is shown in Figure 7. The rotation axis remains largely consistent, with only

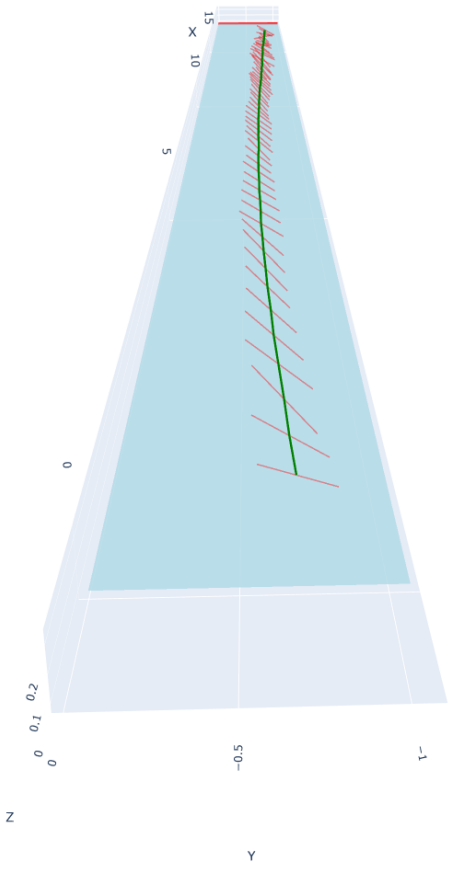


Figure 7. Rotation axis over time for the ball of the video two.

minor frame-by-frame deviations. This indicates that the axis direc-

tion is relatively robust and not significantly affected by the motion blur.

4. Conclusions

This section summarizes the main findings of the project and the possible improvements.

For ball detection, we initially employed classical computer vision techniques including background subtraction and Hough circle detection, which produced reasonably good results. However, strong reflections on both the ball and the lane surface caused detection difficulties in later frames, particularly as the ball moved farther from the camera.

To address these limitations, we implemented an object detection neural network, which significantly improved performance. The deep learning approach provided more precise frame-by-frame detections, more accurate radius estimation, and reliable ball tracking even in the final frames of the videos where traditional methods failed.

In the case in which also the object detection network failed (because a part of the lane was poorly illuminated and the resolution of the video was not the best), we applied mathematical extrapolation to estimate the ball's position, providing a reasonable approximation despite reduced precision.

The spin rate estimation demonstrates consistent performance throughout the trajectory, with reliable tracking even as the ball moves away from the camera. We observe a slight discrepancy in computed spin values between the two camera views, which can be attributed to differences in perspective, lighting conditions, and video quality between the recordings.

The 3D rotation axis estimation proves robust and physically accurate, correctly capturing the spin dynamics throughout the ball's motion. The implementation of 3D feature projection and cross-product-based axis calculation provides a geometrically correct representation of the ball's rotational behavior.

Overall, the proposed approach delivers reliable performance for the majority of the trajectory. The primary limitations stem from camera positioning at the start of the lane, which creates increasing tracking challenges as the ball moves farther away. To enhance the method further, we recommend employing higher-resolution cameras and implementing additional camera placements along the lane length. This multi-view setup would enable robust tracking and precise angular speed estimation throughout the entire trajectory, regardless of the ball's position on the lane.

Future work could also explore more sophisticated sensor fusion techniques to combine data from multiple camera views, potentially reducing the observed discrepancies in spin rate estimation and providing even more accurate 3D trajectory reconstruction.

5. Implementation and folder structure

This section describes the implementation of the project, the main technologies employed, and the organization of the source code and resources.

The project was developed in Python, primarily using the `OpenCV` and `Ultralytics` libraries for image analysis and object detection. Additional libraries include `Plotly` and `Dash` for interactive visualization, `Librosa` for audio-based synchronization, and both `Scikit-learn` and `SciPy` for interpolation and extrapolation tasks. This combination of libraries ensures an efficient pipeline for processing, analyzing, and visualizing experimental data.
The program can be executed in two main modes:

- Running `main.py` executes the full processing pipeline, from data acquisition to analysis and result generation.
- Running `dashapp.py` launches the interactive web application, enabling users to explore videos, results, and visualizations dynamically.

All videos, models, and intermediate results are stored in the `resources` folder, organized by type for reproducibility and ease of access. Meanwhile, the `src` directory contains all Python scripts required to run the project, including the modular pipeline implementation and visualization tools.

The folder structure of the project is organized as follows:

```
config ..................Configuration files for the pipeline
resources
├── calibration ......Intrinsic calibration videos and images
├── graphs ..............Generated plots from the analysis
├── images ........Snapshots and visual results of the analysis
├── models ..............Pre-trained YOLO model binaries
├── pickle ...................Serialized intermediate results
└── videos ...................Original and processed videos
src
├── pipeline ......Scripts to launch and execute the pipeline
│   └── pipes ..................Individual pipeline modules
├── dashapp.py ..Web application for interactive visualization
└── main.py .............Main entry point of the application
```

References

[1] W. contributors, *Perspective-n-point*, https://en.wikipedia.org/wiki/Perspective-n-Point.

[2] O. documentation, *Perspective-n-point solvers*, https://docs.opencv.org/4.x/d5/d1f/calib3d_solvePnP.html.

[3] W. contributors, *Rodrigues rotation formula*, https://en.wikipedia.org/wiki/Rodrigues_rotation_formula.

[4] Ultralytics, *Yolov8*, https://yolov8.com/.

[5] W. contributors, *Optical flow*, https://en.wikipedia.org/wiki/Optical_flow.

[6] W. contributors, *Shi-tomasi corner detection*, https://en.wikipedia.org/wiki/Corner_detection.

[7] W. contributors, *Lucas-kanade method*, https://en.wikipedia.org/wiki/Lucas--Kanade_method.

[8] R. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*. Cambridge: Cambridge University Press, 2004.

[9] V. Caglioti, "Single view geometry", Politecnico di Milano, 2024.