Silesian
University
of Technology

FINAL PROJECT

Smart 9-Ball Assistant

Sara SOBSTYL
Student identification number: ⟨306088⟩

Programme: Informatics
Specialisation: ⟨CGS⟩

SUPERVISOR
⟨Dr. hab. inż. Jakub Nalepa⟩
DEPARTMENT ⟨Katedra Algorytmiki i Oprogramowania⟩
Faculty of Automatic Control, Electronics and Computer Science

Gliwice 2025

Thesis title

Smart 9-Ball Assistant

Abstract

A project that helps players during a game of 9-ball billiards by showing them how the white ball would likely move after a shot. The system has two main parts: 1. Phone App on the Cue Stick - the app that uses the phone's motion sensors (like gyroscope and accelerometer) to track how the cue stick is held and moved. The app sends this data in real time to a computer. 2. Camera Above the Table + Computer - A camera above the billiard table sends live video to a computer. There, a program using e.g., OpenCV detects where all the balls are based on their colors. It figures out where the cue ball is and updates the layout of the table.

Key words

(2-5 keywords, separated by commas)

Tytuł pracy

Inteligentny asystent do bilarda

Streszczenie

(Streszczenie pracy – odpowiednie pole w systemie APD powinno zawierać kopię tego streszczenia.)

Słowa kluczowe

(2-5 slow (fraz) kluczowych, oddzielonych przecinkami)

Contents

Chapter 1

Introduction

Billiards, specifically the 9-ball variation, is a sport that demands a high degree of precision, spatial awareness, and consistent motor control. Unlike many other sports where physical athleticism is paramount, billiards is effectively a game of applied geometry and physics executed through fine muscle memory.

1.1 Introduction into the problem domain

The mastery of billiards relies on two distinct skill sets: the ability to visualize the correct trajectory of the balls and the physical ability to execute the stroke required to send the cue ball along that trajectory. Novice players often struggle with the concept of the ”Ghost Ball” [**bib:ghost_ball**]—the imaginary position where the cue ball must strike the object ball to pocket it.

Furthermore, the mechanics of the stroke itself involving stance, bridge stability, and wrist action significantly impact the outcome of a shot. Even a correct aim can result in a miss if the player introduces unwanted lateral acceleration or English [2] (spin) due to wrist instability during the cue delivery. With the advent of accessible Computer Vision (CV), it is now possible to digitize these physical interactions to provide real-time feedback. [**bib:cv_in_sports**]

1.2 Settling of the problem in the domain

Current training methods in billiards are predominantly manual, relying on the subjective observation of coaches or the player's own intuition. While professional tracking systems exist (such as Hawk-Eye used in tennis or snooker) [**bib:hawk_eye**], they are often prohibitively expensive and require fixed, industrial-grade hardware setups. On the consumer end, mobile applications often rely solely on 2D video analysis, which lacks the depth perception required for accurate table mapping or the high-frequency sampling needed for detailed stroke analysis.

1.3 Objective of the thesis

The primary objective of this thesis is to design, implement, and test a comprehensive "Smart Pool Assistant and Stroke Analyzer". The system aims to assist players in real-time by visualizing shot outcomes and analyzing the mechanics of their cue stroke.

The specific objectives are defined as follows:

- To develop a Computer Vision module capable of detecting billiard balls and the cue stick in a live video feed.
- To implement a physics engine that calculates and visualizes the predicted trajectory of the cue ball (Tangent and Normal lines) based on the "Ghost Ball" principle.
- To create a mobile telemetry system that captures high-frequency accelerometer and gyroscope data from the player's arm.
- To derive meaningful biomechanical metrics, such as impact force ($F = ma$) and wrist stability, to provide objective feedback on the player's technique.

1.4 Scope of the thesis

The scope of this project encompasses the software and hardware integration required to build a functional prototype for the game of 9-ball.

- **Vision System:** The visual analysis is restricted to a top-down camera view. The software utilizes Python and OpenCV libraries, integrating machine learning models (Roboflow) for object detection. The trajectory prediction assumes an ideal collision model (elastic collision) and focuses on the immediate path of the cue ball and the target ball.
- **Sensor System:** The biomechanical analysis is limited to the data provided by standard Android smartphone sensors (Linear Acceleration and Gyroscope). The scope includes the development of a TCP/IP communication protocol to transfer telemetry data to the central computer for processing.
- **Hardware:** The system is designed to run on a standard consumer PC connected to a webcam and an Android device via USB (ADB forwarding).

1.5 Short description of chapters

The thesis is organized as follows:

Chapter 2 (Problem analysis) provides the theoretical background of the project. It scrutinizes the physics of billiard collisions, the geometry of ”Ghost Ball” aiming, and the mechanics of inertial sensors. It also reviews existing solutions in sports technology to establish the context for the proposed system.

Chapter 3 (Requirements and tools) defines the functional and non-functional requirements of the system. It also presents the technological stack selected for the project, justifying the choice of Python, OpenCV, and the Android platform for their respective modules.

Chapter 4 (External specification) describes the system from the user's perspective. It details the user interface (UI) of the Android application and the Augmented Reality (AR) visualization displayed on the desktop, explaining the flow of interaction between the player and the assistant.

Chapter 5 (Internal specification) delves into the technical architecture and backend logic. It explains the algorithms used for object detection and trajectory prediction, the TCP/IP communication protocol, and the signal processing techniques (e.g., gravity removal) applied to the sensor data.

Chapter 6 (Verification and validation) presents the testing process. It evaluates the accuracy of the computer vision model, the latency of real-time data transmission, and the reliability of the stroke analysis metrics through unit tests and practical scenarios.

Chapter 7 concludes the thesis, summarizing the achieved objectives and proposing potential directions for future development.

Chapter 2

Problem analysis

This chapter provides the theoretical and mathematical foundation for the developed system. It analyzes the geometric complexities of billiard aiming implemented in the vision module and the biomechanical principles governing the sensor analysis. Furthermore, it reviews the current state of the art to contextualize the proposed hybrid solution.

2.1 Problem analysis

The game of 9-ball billiards presents a unique challenge where the player must translate a 3D physical intention into a 2D geometric execution. The core problems addressed by this thesis are classified into visual perception and biomechanical execution.

2.1.1 The Visual Paradox (Ghost Ball)

As defined by Alciatore, the ”Ghost Ball” (the required position of the cue ball at impact) is an imaginary point in empty space [1]. Players struggle to visualize this point because the line of aim (passing through the ghost ball center) does not coincide with the contact point on the object ball. This geometric offset forces the player to aim at a non-existent target, creating a ”visual paradox” that is difficult for novices to overcome without augmented feedback.

2.1.2 Biomechanical Inconsistency

A correct aiming line is useless if the stroke delivery is flawed. The mechanics of the cue stroke are critical to the outcome of the shot.

Ballistic vs. Tetanic Stroke: According to the analysis by Moore, a fundamental distinction exists between amateur and professional stroke mechanics [6]. Amateur players often utilize a ”ballistic” stroke characterized by a sudden ”quick yank” at the start of the movement. This approach makes velocity control at the moment of impact unreliable. In contrast, professionals strive for a ”tetanic” stroke—a constant state of muscle contraction—which maintains a consistent accelerating force throughout the cue delivery [6].

Lateral Deviation: Another critical factor is the stability of the swing axis. Failure to maintain a vertical ”pendulum swing” (keeping the elbow fixed in space) causes the cue tip to wander laterally off the intended aiming line [6]. Common errors include lateral wrist deviation (measured via gyroscope) and inconsistent impact force (measured via accelerometer).

2.2 State of the art

The domain of ”Smart Sports” has evolved significantly with the adoption of Convolutional Neural Networks (CNNs) and Wearable Sensors. [5]

2.2.1 Existing Solutions

- **Hawk-Eye (Computer Vision):** Widely used in snooker and tennis, this system relies on multiple calibrated high-speed cameras to triangulate 3D ball positions. While highly accurate ($<$ 3mm error), it requires a fixed, expensive infrastructure and typically does not analyze the player’s biomechanics, only the ball’s trajectory. [4]
- **Projection AR Systems:** Solutions like *PoolLiveAid* use overhead projectors to display trajectories directly onto the table cloth. These provide excellent user experience but require expensive hardware and complex calibration procedures. [3]
- **Wearable IMU Analyzers:** Devices such as *Blast Motion* (used in golf or baseball) utilize inertial sensors to track swing metrics. However, these are typically ”closed systems” that analyze the swing in isolation. They track the movement but do not interact with the game environment (e.g., they do not know the position of the balls). [7]

2.3 Mathematical Models

The implementation of the system relies on specific mathematical models derived from vector algebra and Newtonian mechanics.

2.3.1 Ghost Ball Vector Math

The core logic of the vision module relies on calculating the intersection of the aiming vector with the target ball’s collision zone.

Let P_{cue} and P_{target} be the centers of the cue ball and target ball, and $\vec{v}_{aim}$ be the normalized direction vector of the cue stick. The projection length L_{proj} of the target vector onto the aim vector is calculated using the dot product:

$$L_{proj} = (P_{target} - P_{cue}) \cdot \vec{v}_{aim} \tag{2.1}$$

The closest point on the aiming line to the target ball, P_{close}, is:

$$P_{close} = P_{cue} + \vec{v}_{aim} \cdot L_{proj} \tag{2.2}$$

A valid collision occurs only if the perpendicular distance $d_{\perp}$ is less than the ball diameter D:

$$d_{\perp} = ||P_{target} - P_{close}|| \tag{2.3}$$

If $d_{\perp} < D$, the Ghost Ball position P_{ghost} is found by retreating from P_{close} by an offset calculated via the Pythagorean theorem:

$$P_{ghost} = P_{close} - \vec{v}_{aim} \cdot \sqrt{D^2 - d_{\perp}^2} \tag{2.4}$$

This formula allows the system to draw the ”Ghost Ball” circle exactly where the white ball will be at the moment of impact.

2.3.2 Physics of the Stroke (Sensor Fusion)

The sensor module processes raw data to estimate the force of the shot.

Linear Acceleration

The Android ‘TYPE_LINEAR_ACCELERATION‘ virtual sensor is used to isolate the user’s movement from Earth’s gravity (g). The resulting acceleration vector $\vec{a} = [a_x, a_y, a_z]$ represents the proper acceleration of the cue stick. The magnitude of this acceleration is:

$$||\vec{a}|| = \sqrt{a_x^2 + a_y^2 + a_z^2} \tag{2.5}$$

Force Estimation

To provide the player with a metric of ”Shot Power” in Newtons, the system applies Newton’s Second Law. Assuming a constant estimated mass for the player’s forearm and cue ($m_{arm} \approx 4.0$ kg), the impact force F is:

$$F = m_{arm} \cdot ||\vec{a}_{impact}|| \tag{2.6}$$

2.4 Algorithms

Two primary algorithmic domains are utilized to interpret the raw data streams: Computer Vision for game state analysis and Signal Processing for biomechanical analysis.

2.4.1 Computer Vision Algorithms

The vision module utilizes the YOLO (You Only Look Once) architecture, deployed in two distinct configurations to handle different tracking requirements.

Object Detection (Balls)

To identify the game elements, a standard object detection model is employed. It classifies objects into categories such as `cue-ball` and `other` (object balls). The inference function f_{det} transforms the input image I into a set of bounding boxes B:

$$f_{det}(I) \rightarrow \{(x, y, w, h, class, conf)_i\} \tag{2.7}$$

where (x, y) is the center of the ball, used as the input for the "Ghost Ball" geometric calculation.

Pose Estimation (Cue Stick)

Determining the aiming line requires more precision than a bounding box can provide. Therefore, a **Pose Estimation** model (specifically `yolov8-pose`) is used to detect the cue stick. Instead of a box, this model predicts specific semantic keypoints: the `tip` (cue tip) and the `handle` (grip point). The output is a set of coordinates K:

$$f_{pose}(I) \rightarrow \{(x_{tip}, y_{tip}), (x_{handle}, y_{handle})\} \tag{2.8}$$

The aiming vector $\vec{v}_{aim}$ is then derived directly from these keypoints: $\vec{v}_{aim} = (x_{tip} - x_{handle}, y_{tip} - y_{handle})$.

2.4.2 Peak Detection Algorithm

To automatically detect the moment of impact without external triggers, the system implements a real-time peak detection algorithm on the linear acceleration data stream. A hit is registered at time t if the acceleration magnitude $a(t)$ satisfies both a threshold condition and a local maximum condition:

$$a(t) > T_{peak} \quad \wedge \quad a(t) \geq a(t-1) \quad \wedge \quad a(t) \geq a(t+1) \tag{2.9}$$

where T_{peak} is the noise gate threshold (set to $15.0 m/s^2$ in the implementation). This ensures that the system captures the exact moment of maximum force exertion (F_{max}) for the biomechanical analysis.

Chapter 3

Requirements and tools

This chapter details the functional and non-functional requirements of the ”Smart Pool Assistant” and presents the technological stack selected for implementation. It also describes the system’s use cases and the methodology adopted for the design and development process.

3.1 Functional and non-functional requirements

The system requirements were defined to address the problems identified in the previous chapter, specifically the ”Ghost Ball” visualization and the biomechanical analysis of the stroke.

3.1.1 Functional Requirements

The functional requirements define the specific behaviors and functions the system must support. They are categorized by module:

Vision Module (Desktop):

- **FR-01 Video Acquisition:** The system must capture a real-time video stream from a webcam positioned above the billiard table.
- **FR-02 Object Detection:** The system must detect and classify the cue ball and object balls using a Convolutional Neural Network (YOLO).
- **FR-03 Pose Estimation:** The system must identify keypoints of the cue stick (tip and handle) to determine the aiming vector.
- **FR-04 Trajectory Prediction:** The system must calculate the ”Ghost Ball” position and predict the trajectories of the cue ball and target ball based on geometric rules.

- **FR-05 AR Visualization:** The system must overlay the predicted paths and the Ghost Ball indicator onto the video feed in real-time.

Sensor Module (Mobile):

- **FR-06 Data Acquisition:** The mobile application must read data from the accelerometer (linear acceleration) and gyroscope at a frequency of at least 50 Hz.
- **FR-07 Data Transmission:** The mobile app must transmit sensor data to the desktop server via a TCP/IP socket connection (USB tethering).
- **FR-08 Stroke Analysis:** The system must detect the moment of impact (peak acceleration) and calculate the impact force ($F = ma$) and wrist rotation stability.

3.1.2 Non-functional Requirements

The non-functional requirements define the quality attributes of the system:

- **NFR-01 Real-time Performance:** The vision processing pipeline must maintain a frame rate of at least 20 FPS (frames per second) to provide smooth visual feedback.
- **NFR-02 Latency:** The latency between the physical cue movement and the AR update should be less than 150 ms to ensure the user perceives the lines as responsive.
- **NFR-03 Accuracy:** The "Ghost Ball" projection error should be less than 5% of the ball diameter to be practically useful for aiming.
- **NFR-04 Sensor Synchronization:** The time drift between the video impact detection and sensor peak detection must be handled to correctly associate the physical stroke with the visual event.
- **NFR-05 Usability:** The setup process (camera alignment, phone connection) should be performable by a single user within 5 minutes.

3.2 Use cases

The interaction between the user (Player) and the system is modeled through several key Use Cases. The primary actor is the Player, who interacts with both the physical equipment (Phone, Cue) and the software (Desktop App).

Figure 3.1: UML Use Case Diagram of the Smart Pool Assistant.

The main use cases are defined as follows:

1. **UC-1 System Setup:** The Player connects the Android phone to the PC (via USB/ADB), starts the server script, and launches the mobile app. The connection is established if the "Connected" status appears.

2. **UC-2 Aiming Assistance:** The Player addresses the cue ball. The system detects the cue stick, calculates the trajectory, and projects the "Ghost Ball" and aiming lines. The Player adjusts their stance based on this visual feedback.

3. **UC-3 Stroke Execution & Analysis:** The Player executes the shot. The system automatically detects the impact via the sensor stream, captures a snapshot of the metrics (Force, Rotation), and saves the data to a history log.

4. **UC-4 Review History:** The Player views the saved "Hit History" (CSV/Images) to analyze their consistency over the training session.

3.3 Description of tools

The project leverages a modern technology stack combining Computer Vision, AI, and Mobile Development.

3.3.1 Hardware Tools

- **Webcam:** A standard HD webcam is used for video input. It is mounted in a "top-down" configuration to minimize perspective distortion.

- **Android Smartphone:** A device equipped with an Inertial Measurement Unit (IMU). It serves as the telemetry unit attached to the player's cue.

- **Workstation (PC):** A computer with access to the internet and with installed Python 3.x environment to run the vision server and process data.

3.3.2 Software Tools

- **Python 3.x:** The core programming language for the backend server and vision processing.

- **OpenCV (Open Source Computer Vision Library):** Used for image pre-processing, drawing the AR visualization (lines, circles), and rendering the real-time telemetry graphs.

- **Ultralytics YOLO (You Only Look Once):** A state-of-the-art framework for object detection.

  - *YOLOv12 (Detection):* Used for robust detection of billiard balls under varying lighting conditions.
  - *YOLOv8-Pose (Keypoint Estimation):* Specifically used to detect the `tip` and `handle` of the cue stick, enabling precise vector calculation.

- **Roboflow:** A platform used for dataset management, image annotation, and versioning. It facilitated the preparation of the training data for the custom billiard model.

- **Android Studio & Java:** The development environment for the mobile sensor application. Java was chosen for its native support of Android Sensor APIs.

- **ADB (Android Debug Bridge):** Utilized for establishing a low-latency reverse TCP connection (`adb reverse`) between the Android device and the localhost server.

3.4 Methodology of design and implementation

The development followed an iterative **Prototyping Methodology**, which is suitable for systems involving experimental algorithms (like computer vision) and hardware integration. The process was divided into four phases:

3.4.1 Phase 1: Data Collection and Model Training

The initial phase focused on the Vision Module. A custom dataset of billiard balls and cue sticks was collected and annotated using Roboflow. The YOLO models were trained iteratively (using Google Colab GPUs) to achieve high mean Average Precision (mAP) for ball detection and keypoint estimation.

3.4.2 Phase 2: Vision Logic Implementation

Once the models were trained, the Python logic was developed to translate raw detections into game state information. This involved implementing the vector algebra for the "Ghost Ball" algorithm and using OpenCV to render the visual overlays.

3.4.3 Phase 3: Sensor Module Development

The Android application was developed to reliably extract linear acceleration and gyroscope data. The focus was on implementing the 'TYPE_LINEAR_ACCELERATION' sensor to filter out gravity and establishing a robust TCP socket protocol for data transmission.

3.4.4 Phase 4: Integration and Testing

The final phase involved fusing the two modules. The Python server was updated to handle concurrent threads: one for video processing and one for listening to incoming sensor packets. The system was tested in real-world scenarios to calibrate the impact detection thresholds and verify the synchronization between the visual stroke and the sensor peak.

Chapter 4

External specification

- hardware and software requirements
- installation procedure
- activation procedure
- types of users
- user manual
- system administration
- security issues
- example of usage
- working scenarios (with screenshots or output files)

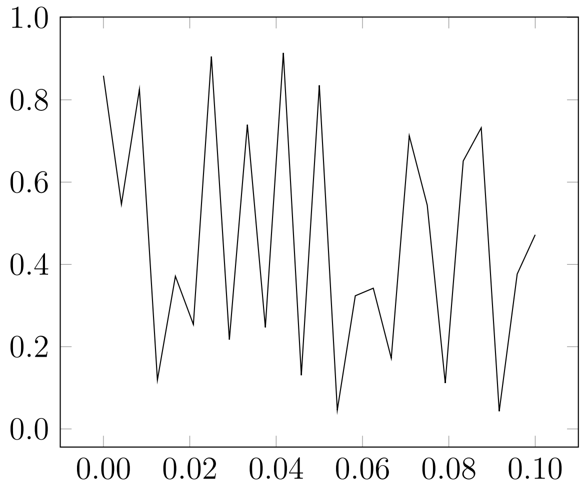

Figure 4.1: Figure caption (below the figure).

Chapter 5

Internal specification

- concept of the system
- system architecture
- description of data structures (and data bases)
- components, modules, libraries, resume of important classes (if used)
- resume of important algorithms (if used)
- details of implementation of selected parts
- applied design patterns
- UML diagrams

Use special environments for inline code, eg `int a;` (package `listings`). Longer parts of code put in the figure environment, eg. code in Fig. 5.1. Very long listings–move to an appendix.

```
class test : public basic
{
    public:
      test (int a);
      friend std::ostream operator<<(std::ostream & s,
                                          const test & t);
    protected:
      int _a;

};
```

Figure 5.1: Pseudocode in `listings`.

Chapter 6

Verification and validation

- testing paradigm (eg V model)
- test cases, testing scope (full / partial)
- detected and fixed bugs
- results of experiments (optional)

Table 6.1: A caption of a table is **above** it.

	method						
			alg. 3			alg. 4, $\gamma = 2$	
ζ	alg. 1	alg. 2	$\alpha = 1.5$	$\alpha = 2$	$\alpha = 3$	$\beta = 0.1$	$\beta = -0.1$
0	8.3250	1.45305	7.5791	14.8517	20.0028	1.16396	1.1365
5	0.6111	2.27126	6.9952	13.8560	18.6064	1.18659	1.1630
10	11.6126	2.69218	6.2520	12.5202	16.8278	1.23180	1.2045
15	0.5665	2.95046	5.7753	11.4588	15.4837	1.25131	1.2614
20	15.8728	3.07225	5.3071	10.3935	13.8738	1.25307	1.2217
25	0.9791	3.19034	5.4575	9.9533	13.0721	1.27104	1.2640
30	2.0228	3.27474	5.7461	9.7164	12.2637	1.33404	1.3209
35	13.4210	3.36086	6.6735	10.0442	12.0270	1.35385	1.3059
40	13.2226	3.36420	7.7248	10.4495	12.0379	1.34919	1.2768
45	12.8445	3.47436	8.5539	10.8552	12.2773	1.42303	1.4362
50	12.9245	3.58228	9.2702	11.2183	12.3990	1.40922	1.3724

Chapter 7

Conclusions

- achieved results with regard to objectives of the thesis and requirements
- path of further development (eg functional extension …)
- encountered difficulties and problems

Bibliography

[1] David Alciatore. *Ghost-ball Aiming*. 2025. URL: `https://drdavepoolinfo.com/resource_files/ghost_ball_aiming.pdf` (visited on 14/02/2025).

[2] David Alciatore. *Sidespin and English Terminology and Uses*. 2012. URL: `https://drdavepoolinfo.com/faq/sidespin/terminology-and-uses/` (visited on 14/02/2025).

[3] Ricardo Alves, Luís Sousa and Joao Rodrigues. 'PoolLiveAid: Augmented reality pool table to assist inexperienced players'. In: *21st International Conference on Computer Graphics, Visualization and Computer Vision*. 2013, pp. 184 –193.

[4] Manish Duggal. 'Hawk Eye Technology'. In: *Journal of Global Research Computer Science & Technology* 1.2 (2024), pp. 1 –7.

[5] Bernhard Hollaus, Sebastian Stabinger, Andreas Mehrle and Christian Raschner. 'Using Wearable Sensors and a Convolutional Neural Network for Catch Detection in American Football'. In: *Sensors* 20.23 (2020), p. 6722.

[6] A. D. Moore. 'Mechanics of Billiards, and Analysis of Willie Hoppe's Stroke'. In: University of Michigan, College of Engineering, 1947.

[7] Sean Ogle. *Breaking Eighty*. 2022. URL: `https://breakingeighty.com/blast-motion-golf-review` (visited on 13/10/2022).

Appendices

Index of abbreviations and symbols

DNA deoxyribonucleic acid

MVC model–view–controller

N cardinality of data set

μ membership function of a fuzzy set

$\mathbb{E}$ set of edges of a graph

$\mathcal{L}$ Laplace transformation

Listings

(Put long listings here.)

```
if (_nClusters < 1)
    throw std::string ("unknown number of clusters");
if (_nIterations < 1 and _epsilon < 0)
    throw std::string ("You should set a maximal number of iteration or minimal difference -- epsilon.");
if (_nIterations > 0 and _epsilon > 0)
    throw std::string ("Both number of iterations and minimal epsilon set -- you should set either number of iterations or minimal epsilon.");
```

List of additional files in electronic submission (if applicable)

Additional files uploaded to the system include:

- source code of the application,
- test data,
- a video file showing how software or hardware developed for thesis is used,
- etc.

List of Figures

List of Tables