Blackboard based Reasoning Framework for Explainable Table Tennis Analysis

1st Pramodini Padhmanabhan
Computer Science and Engineering
PSG College of Technology
Coimbatore, India
pramodheepa@gmail.com

2nd Akash Shanmugaraj
Computer Science and Engineering
PSG College of Technology
Coimbatore, India
akashshanmugaraj@gmail.com

3rd Sanjitha Rajakumar
Computer Science and Engineering
PSG College of Technology
Coimbatore, India
sanjitha20041234@gmail.com

4th Sreeraghavan Ramamoorthy
Computer Science and Engineering
PSG College of Technology
Coimbatore, India
sreeraghavan2016@gmail.com

5th Dwarkesh
Computer Science and Engineering
PSG College of Technology
Coimbatore, India
gurudwarkesh@gmail.com

6th Mrs. K. Archana
Computer Science and Engineering
PSG College of Technology
Coimbatore, India
archanak2896@gmail.com

***Abstract*—Reasoning models presently rely a lot on LLMs and LLM agents taking various roles.This has its advantages, but it requires a large number of tokens and computation. There are also other non-LLM based ways that provide intelligence such as various ML models that are specialized.**

Working with multiple specialized models together is a better way to obtain reasoning. This architecture facilitates flexible integration of diverse analysis components and minimizes redundant computation. A shared context layer aggregates module outputs for centralized LLM based querying and downstream applications such as visualization, summarization, and performance reporting.

We demonstrate the applicability of the system in the domain of sports video analysis, using table tennis match footage as a case study. The proposed design generalizes well to other domains that require fine-grained and interpretable video understanding.

***Index Terms*—Blackboard architecture, multi-agent systems, explainable artificial intelligence, sports video analysis, trajectory estimation, depth estimation, autonomous learning**

I. Introduction

Video understanding needs extraction of various, diverse, and different insights like player positions, ball trajectory, player heat map, space point estimator etc. Current approaches try to perform the analysis in a single brute force approach using a large language model taking up a substantial amount of computing power and time, while also sacrificing on the specific task evaluation models accuracy and performance. Recent advancements in machine learning have demonstrated that specialized models have greater performance and accuracy than general machine learning models. However, integrating multiple specialized models into a single reasoning system is challenging yet rewarding with its accuracy and performance. Current systems either use general LLMs that leave out the performance and attention to detail that can be achieved by specialized systems, or use application specific rigidly integrated pipelines that don't work in generalized domains. This highlights the gap between efficient specialized model systems and the flexible, interpretable reasoning system. With this we present a modular intelligence architecture that couples specialized ML models with reasoning models. This system uses blackboard architecture in the backend to accumulate all data to a central point, from which the independent model data can be retrieved. We use a publish subscribe architecture to ensure only required models are run for any sequence. This reduces the amount of computation while maintaining explainability while allowing for flexible, multi model analysis. We check this architecture with the application of table tennis video analysis. The system uses specialized models for player detection, player heat map, ball tracking and ball trajectory estimation. These are integrated with the publish subscribe network and blackboard to generate insights

II. Problem Statement

This project will build a scalable Table Tennis Video Analysis system using deep-learning models and efficient spatio-temporal feature extractors to tally rally exchanges, pinpoint bounce locations, and produce player movement insights.

It uses the following core components - object detection, event segmentation, pose estimation, and sequence analysis. These components work together in a modular and agentic architecture, where each component functions as a semi-autonomous "agent" focused on a specialized task. Outputs from individual agents can be integrated to form higher-level insights such as:Provide point-by-point suggestions based on detected inefficiencies

The agentic architecture also allows for fast extensibility without requiring re-engineering of any core systems

III. Literature Survey

A. Trajectory Analysis

Trajectory analysis depends on multiple data and image processing along with some deep learning models. Some efforts include metadata about the video and objects such as accelerometers to create more reliable information. [16].

1) Workflow: Trajectory analysis in sports follows a systematic multi-stage workflow beginning with data acquisition through high-speed cameras operating at 60-120 fps in synchronized multi-camera setups [21].

This is followed by object detection which employs models such as those in the YOLO family or TrackNetV2 [21]or ByteTrack with Kalman filtering to maintain temporal consistency. Subsequently, the ball detection and camera calibration is used to create 3D trajectories [16].

2) Models Used: Primarily, they have used YOLOv4 and v11 for detecting the ball. Multiple cameras were also required [21]. Advanced reconstruction employs LSTM-based architectures featuring specialized networks: LSTMε for End-of-Trajectory prediction, $\text{LSTM}_{\text{height}}$ for vertical position estimation, and $\text{LSTM}_{\text{refine}}$ for trajectory smoothing.

B. Player Heatmap

Hass et al.,[], created a comprehensive player heatmap pipeline by combining deep learning models and some computer vision algorithms. [4]

1) Workflow: At first, the table was identified by using Hough transforms and some normal image processing techniques and algorithms. Next the players themselves were identified using Facebook's Detectron2. The video was segmented into rallies, however it is not clear if that was automated or manual. The heatmap itself was generated by segmenting the play area into a grid and plotting the player locations onto the grid for each frame. [4]

2) Models Used: Detectron2 for player detection. Hough Transforms for table detection [4].

C. LLM Reasoning Architectures

Earlier AI systems that separated memory, control, and knowledge were based on the Blackboard Model [5]. With the growth of Agentic AI and Large Language Models (LLMs), researchers are reusing this idea to build multi-agent systems. These systems use a mixture-of-experts (MoE) design, where different specialized agents or models work together to solve complex problems efficiently [5]

1) Workflow: A typical workflow starts with a Control Unit (CU),usually an LLM that decides which agents should handle the current task. Each agent processes part of the problem and writes its results to a shared blackboard memory. This shared memory lets every agent access the same information and build on each other's work. The process continues until the system reaches a solution, either by agreement among agents or when a stopping condition is met [5].

2) Models Used: Jacobs et al. [9] first introduced the idea of adaptive mixtures of experts, where different models handle different tasks and their outputs are combined dynamically. Later, the sparsely-gated MoE layer [**?**] improved the efficiency of very large neural networks. Recent studies [13] have shown that MoE architectures help LLMs scale better, and Han and Zhang [14] found that combining them with blackboard coordination further improves reasoning and collaboration among agents.

In short, modern blackboard-based LLM multi-agent systems (bMAS) [5] combine the clear reasoning of early blackboard systems [10], the flexibility of mixtures of experts [9] [1], and the scalability seen in recent MoE research [1] [13].

D. Blackboard

The capabilities of a classical Blackboard Architecture allows researchers to take advantage of multi-agent systems in LLM-driven architectures [14]. Agents enjoy flexible and competent workflows by making use of the information sharing and adaptive scheduling features of the blackboard. [5]. Implementations of LbMAS have shown to be scalable and efficient.

1) Workflow: The blackboard-based LLM multi-agent system (bMAS) undergoes one or more iterations during one problem solving session. Each iteration is an agent orchestration, managed by the Control Unit (CU) [14]. Each problem-solving session begins with agent generation, where an expert skills required based on some query. The CU then selects appropriate agents based on the skills and provides them with the blackboard content. Agents consume and interpret this shared memory, perform computations, contribute their outputs back to the blackboard. This whole process continues until some stopping condition is satisfied - either a consensus or a decider's output or maximum rounds. [5].

2) Models Used: Classical Blackboard Architecture has three core components: (1) Control Unit (2) Blackboard (3) Knowledge Sources / Agents. The control unit, which could also be an LLM, decides which skills and agents are required at a given stage [5], thus ensuring efficiency and reducing token consumption. The blackboard functions as shared, persistent memory, with potential to be separated into public and private spaces, where oen or more agents record, debate, and refine their outputs [14].

Advanced agents are made capable of special roles like problem decomposition, evaluation of inconsistencies, or solution proposal. The whole system, given enough iterations, converges into one single solution. [5].

IV. System Design Architecture

We aim to integrate the Blackboard Architecture (BA) into multi-agent systems (MASs) so that (1) agents with specific and unique skills / capabilities can share information during the problem-solving process (2) decisions and actions taken by agents are based on the content of the blackboard at that time.

By allowing differents knowledge sources (agents) to communicate by the means of a common information field (blackboard), we create systems capable of complex problem solving compared to traditional monolithic and sequencial systems.

We develop the first implementation of this proposal and conduct experiments on footages of Table Tennis gameplay. The overall system architecture / workflow is illustrated in Figure 1.

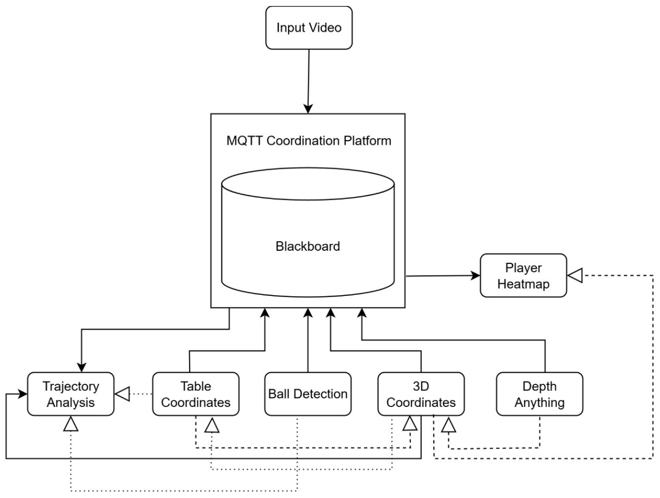

Fig. 1. Overall System Architecture Diagram

A. Overview and Motivation

The Blackboard Architecture was first proposed [6] [15] during early 1980s as a decentralized problem solving approach that mimics human expertise collaborating around an actual blackboard, each contributing their own solution until the program is solved.

Employing a blackboard architecture in a Multi-Agent System (MAS) helps to take advantage of collective intelligence and thereby further enhancing problem-solving performance of the system.

B. Core Components

The BA itself relies on four core components: (1) Blackboard (2) Knowledge Source (3) Communication Interface (4) Control Unit (CU). Information from the Knowledge Sources are stored on the Blackboard and are being subsequently reused by other Knowledge Sources to iteratively approach a solution.

1) Blackboard: The blackboard acts as a central, global data structure accessible by all modules ("knowledge sources"). Each module posts intermediate or final results to the blackboard and reacts to changes made by other modules. Communication and coordination are achieved indirectly by reading and writing to this shared resource, not through direct inter-module calls.

We have implemented the blackboard as different tables in a Relational Database (Postgres) for production grade data storage for relational data with faster retrievals and consistent read write operations.

The central database schema (see Tables I-VI) functions as the blackboard in our blackboard architecture implementation. All agents post their intermediate states, detected events, and final results solely into this schema, which defines the evolving solution space. This "single source of truth" approach ensures consistency, repeatability, and modularity: modules communicate solely through the blackboard and remain decoupled, except via well-defined database reads/writes.

This aligns with classical blackboard systems, where the shared data structure orchestrates expert contributions and manages overall task progression.

TABLE I
SCHEMA OF `TABLE_TENNIS_ANALYSIS`

Column	Type	Description
videoId	INTEGER	Unique video identifier
frameId	INTEGER	Frame number within video
frameAction	VARCHAR	Action detected in frame
tableid	INTEGER	Table reference
ballvisibility	BOOLEAN	Is ball visible?
ballid	INTEGER	Ball reference
depthMapPath	VARCHAR	Path to depth map file
ballbounce	BOOLEAN	Ball bounce event
remarks	VARCHAR	Additional comments

TABLE II
SCHEMA OF `TABLE_COORDS`

Column	Type	Description
tableid	INTEGER	Table identifier
tablex1	FLOAT	Table coordinates
tabley1	FLOAT	
tablex2	FLOAT	
tabley2	FLOAT	
tablex3	FLOAT	
tabley3	FLOAT	
tablex4	FLOAT	
tabley4	FLOAT	

TABLE III
SCHEMA OF `BALL_DATA`

Column	Type	Description
ballId	INTEGER	Ball identifier
ballx	FLOAT	Ball Position
bally	FLOAT	
ballz	FLOAT	
ballxvector	FLOAT	Ball Velocity
ballyvector	FLOAT	
ballzvector	FLOAT	

TABLE IV
SCHEMA OF `PLAYER_POSITIONS`

Column	Type	Description
playerid	INTEGER	Player identifier
playerx	FLOAT	Player position
playery	FLOAT	
playerz	FLOAT	

2) Knowledge Sources: The knowledge source is any set of functions that take some input and generate a result that brings the entire system one step closer to the solution. The input can either be raw data or an output of another knowledge source. Each knowledge source ideally has a very specific set of skills that does not overlap with another. The individual agents in our system, namely (1) Table Vertex Detection (2) Ball Position Detection (3) Player Heatmap Generation (4) Space Point Estimator and (5) Trajectory Analysis are the knowledge sources. These knowledge sources have been

TABLE V
SCHEMA OF BOUNCES

Column	Type	Description
frameId	INTEGER	Frame identifier for bounce event
ballId	INTEGER	Ball identifier for bounce event

TABLE VI
SCHEMA OF VIDEO_TABLE

Column	Type	Description
videoId	INTEGER	Video identifier
videoPath	VARCHAR	Path to video file
videoName	VARCHAR	Name of video
videoTag	VARCHAR	Custom video tag/label
fullvideoHeatmapPath	VARCHAR	Path to video heatmap
videoDotMatrixSource	VARCHAR	Path/source for dot matrix map

programmatically implemented as objects of their own classes, which are extended from an abstract class which has been illustrated in Figure 3. We believe that this approach will contribute to the versatility, diversity and generality of agent introduction at scale.

3) Communication Interface: The Communication Interface handles the communication between the knowledge sources and is based on the Publish Subscribe Model. This very model is the backbone behind BA, allowing agents share findings and results. We have implemented a Pub-Sub Model using RabbitMQ, an open-source message broker that enables reliable, asynchronous communication between distributed applications by queuing, routing, and delivering messages. Every knowledge source listens for tasks (messages) in a predefined queue and defines a *callback-function*.

CU places message on the appropriate queue, on behalf of agents, based on the skill required. Agents are triggered using the callback function when their queue gets a message. These agents perform requested operations on the blackboard and send a success message upon completion.

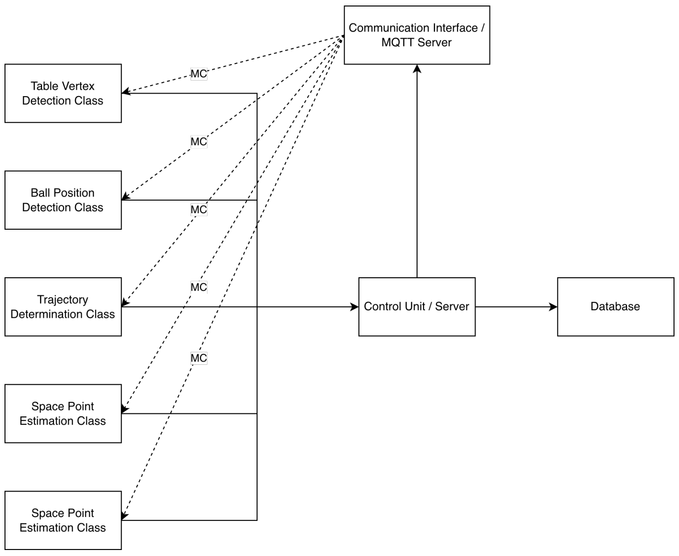
Communication Interface / MQTT Server
Table Vertex Detection Class
Ball Position Detection Class
Trajectory Determination Class
Space Point Estimation Class
Space Point Estimation Class
Control Unit / Server
Database
MC

Fig. 2. Component Interaction Diagram. Abbreviation: MC = Message Callback; MQTT = Message Queuing Telemetry Transport;

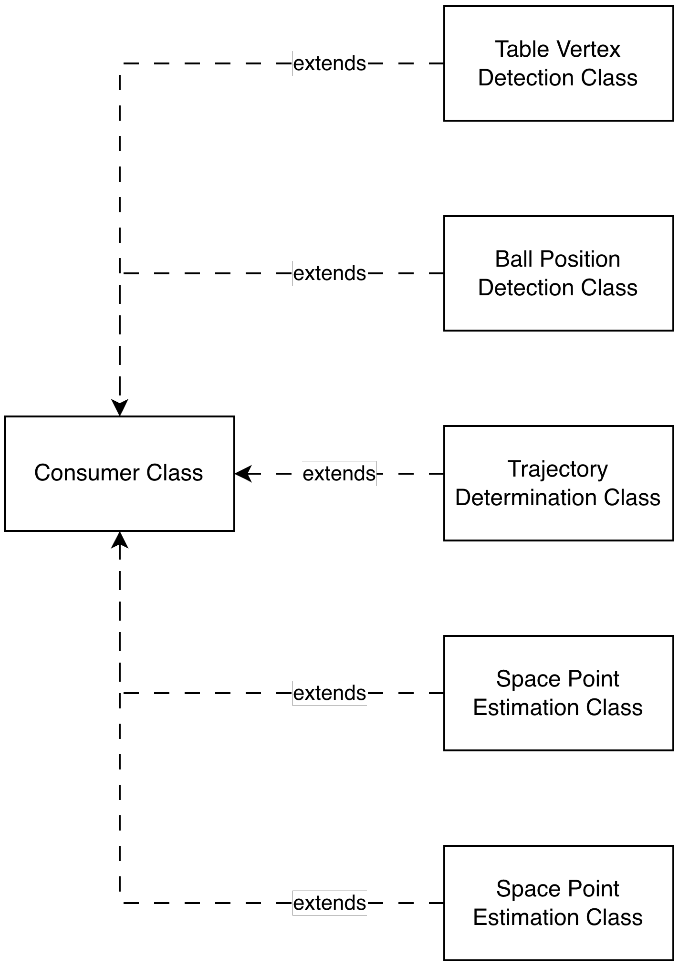
Table Vertex Detection Class
Ball Position Detection Class
Consumer Class
Trajectory Determination Class
Space Point Estimation Class
Space Point Estimation Class
extends

Fig. 3. Class Diagram

4) Control Unit: The knowledge models are loosely coupled via task delegation to the CU. When the ball tracking model requires table coordinates, it submits the request to the CU, without needing awareness of which agent holds or computes this data. Since the CU is preconfigured with the list of knowledge sources, their skill sets, the IDs of the queue that they are listening to, it dynamically resolves and forwards this request to the appropriate agent, which processes and returns the result.

This approach enables modularity and scalability by decoupling task requesters from the concrete implementers, following the orchestrator-worker pattern commonly employed in multi-agent and event-driven distributed systems

To enhance versatility and support dynamic integration of knowledge sources (agents), the control unit is designed to automatically extract and register the skill set and message queue information of each new consumer as it connects.

This mechanism enables seamless discovery and on-the-fly integration of new capabilities, without requiring manual configuration or prior knowledge of the agent's skills. As a result, the control unit serves as a dynamic registry, ensuring efficient routing of tasks based on updated agent capabilities as soon as they are available

C. Component Description

These are the knowledge sources that are incorporated into the overall system architecture

1) Table Vertex Detection: The results from this module is required by the trajectory analysis module and for answering certain questions by the answering machines.

2) Ball Position Detection: The ball detection module serves as a foundational constituent in the proposed table tennis analysis system.It enables precise localization of the ball across frames to facilitate analytics such as trajectory analysis. Accurate ball detection is essential for ensuring the reliability of subsequent modules that depend on spatial and temporal ball movement data.

3) Player Heatmap Generation: The player heatmap is a useful tool for user analysis as it shows the behaviour and preferred styles of each player. It provides the positions of each player on every frame using a simple pretrained YOLO model

4) Space Point Estimator: The space point estimator adds an extra layer of insight as it can convert points on a 2D video into 3D coordinates.

5) Trajectory Analysis: The trajectory analysis component delivers accurate ball tracking and motion analysis from high-frame-rate table tennis videos. It automatically extracts the ball's spatial positions, velocities, and bounce points with high precision, supporting both tactical coaching and detailed player performance evaluation.

V. Implementation

A. Table Vertex Detection

This is a crucial model for several other deeper analysis as it provides a sense of space to the 2D video which is useful for trajectory analysis, 3D analysis and certain types of questions. Following the work done by Haas et al., we first tried Hough Transforms and image processing to find the table coordinates. However this ran into many issues,

- For many videos the edge detection had too many gaps and smoothing was not enough to fix the images well enough for hough transformation to work
- Small disturbances could cause the table to be detected very wrongly
- The lines on the table could cause issues with the table detection
- The image processing was such that it could be made to work for some cases but only at the cost of others

These issues led to moving towards a more general deep learning pipeline using YOLO. We used a YOLOv8 pose estimation model trained on custom manually and synthetically generated data from the dataset. The pose estimation did not work well with the amount of data present in the dataset. However, the bounding boxes worked well and were used in the final system. The YOLO model was wrapped in a function

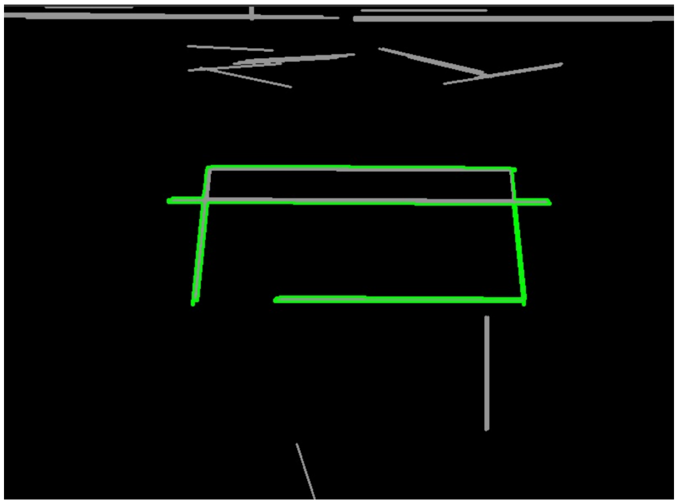

Fig. 4. Issues due to gaps in edge detection due to player interference

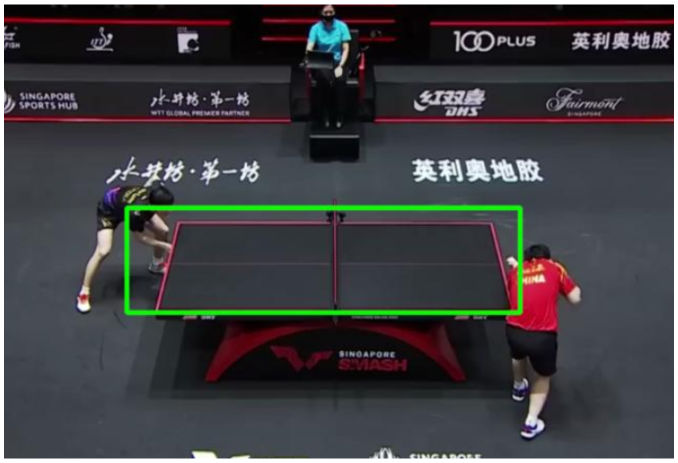

Fig. 5. Final YOLO Implementation of Table Detection

B. Ball Position Detection

The YOLOv11n model was chosen for this task because it is lightweight and good at spotting small and fast-moving objects; i,e. A table tennis ball. A large-scale dataset comprising approximately 88,599 annotated frames was used for model training in the initial stage.However, due to the high computational requirements and limited GPU resources in Google Colab, the complete dataset could not be processed efficiently. To address this, 6,721 frames containing the ball were extracted and used for training. This approach resulted in high loss values, primarily due to the lack of negative samples representing non-ball instances. To help the model learn better, a balanced dataset was made with 6,721 ball frames and 2,700 frames with no ball. This helped improve recall and made training faster.

C. Trajectory Analysis

1) Framewise Ball Position Detection: This data is collected from the blackboard and is computed by the Ball

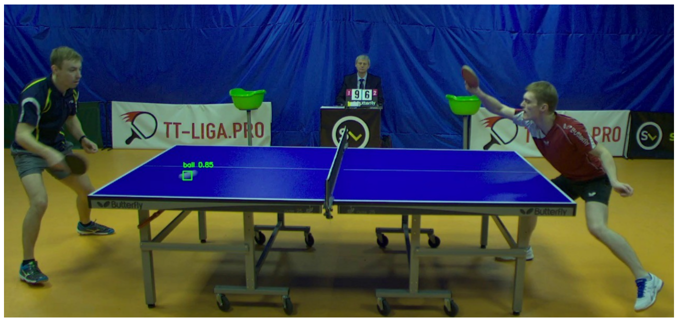

Fig. 6. Ball Position Detection

Position Detection Module. This step is the backbone sequence of ball locations.

2) Interpolation: Often, sports videos have moments where the ball is not detected, that even advanced models can lose track of it a few times. To maintain the path continuity and prevent analytical errors, the pipeline significantly identifies such missing spaces in detection. For gaps no larger than a particular threshold (e.g., 5 frames), the system applies cubic spline interpolation to fill in the missing details

- Identify all consecutive frames where the position data is missing
- For gaps that are surrounded by valid positions, fill in the missing points using a cubic spline curve, ensuring the trajectory remains smooth and continuous.

3) Kalman Filter and Smoothing: The raw or interpolated ball positions still include some noise, that is caused by imperfect detections or changes in the environment. To address this, Kalman filter is applied. The filter tracks the movement of the ball and combines the predicted movement (based on physics) with the actual which is sometimes noisy.

- The initial ball position is determined from the earliest available frames.
- The filter parameters, such as process and measurement noise, are adjusted according to the video's resolution, frame rate, and the expected motion behavior of the ball.

4) Bounce Points Detection: Bounces are key features in table tennis analysis. The system looks at the smoothed y-coordinates to detect drops near the table surface that indicate possible bounce. To distinguish real bounces from noise, it also checks for a change in the sign of the vertical velocity,which is a strong indicator that the ball has hit the surface and reversed the direction.

- For each frame, compare its y-value with neighboring frames (i-1, i, i+1) to find local minima that may represent bounces.
- Verify a change in sign confirms a reversal in motion.
- Only consider frames within a set vertical range as valid bounce candidates.

5) Velocity Computation: The ball's velocity information is crucial to analyze the ball's motion, . This system computes the velocity after smoothing the path of the ball, through the numerical gradient of its x and y position values for each frame. This gives both horizontal and vertical components of the velocity, namely $(ball_x vector)$ and $(ball_y vector)$

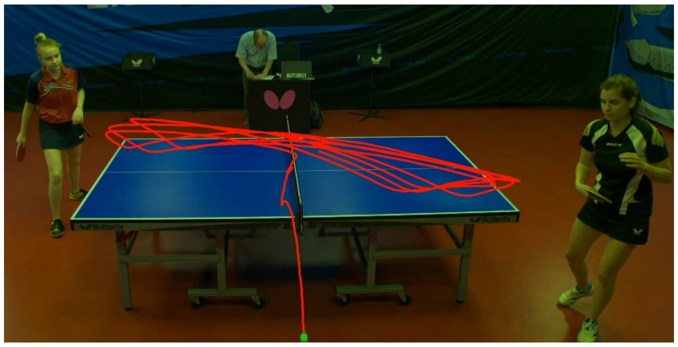

Fig. 7. Path generated by Trajectory Analysis

D. Space Point Estimator

The main objective or intention of the Space Point Estimator module is to transform 2D points captured from table tennis video frames into 3D coordinates of the world. Doing this facilitates more detailed and precise analytics throughout the system. This transformation is done using advanced depth estimation with the Depth Anything V2 model.

1) Workflow and Process: When preparing a frame (image) for depth estimation such as a video frame where table positions or ball positions are identified, the image path is submitted to the estimator.

The steps of the process are:

- **Image Input and Preprocessing:** The estimator takes the video frame path, loads the image, converts into a standard RGB format, and its dimensions are stored.
- **Depth Anything V2 Model Inference:** The heart of the system is the Depth Anything V2 model.It is a transformer based architecture, pre-trained to figure out the depth from a camera angle. Before processing, the image is converted into tensors with which the model works.The model generates a dense depth map, which illustrates per-pixel distance estimations within the scene.
- **Depth Map Normalization & Scaling:** The depth map is first normalized, based on the minimum and maximum depths within the map. After normalization, it is scaled to a specific metric range (for example, up to 1499 mm) to suit the requirements of the table tennis analysis.
- **Sub-Pixel 3D Coordinate Retrieval:** For any (x, y) pixel coordinate in the image like the position of a ball or player, the system uses bilinear interpolation on the depth map to estimate a precise Z (depth) value.
- **Output:** The resulting (x, y, z) coordinate of the requested point can now be directly translated into real-world 3D space.

VI. Experimental Results

Every module has to be evaluated differently due to the varying nature of our Knowledge Sources. Some are ML models while others are simple data and image processing.

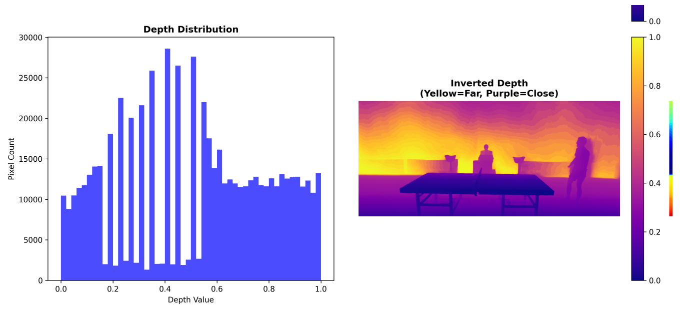

Fig. 8. Depth distribution and inverted depth map produced by the Space Point Estimator.

The performance metrics of the Knowledge Sources are given below

A. Table Vertex Detection

Here, 2 metrics were employed, Bounding Box Overlap, which checks the area of the predicted box that covers the actual object and a custom metric. Metrics such as Precision and Recall aren't informative as the final usage comes from an average over many frames so model performance over individual frames are not used

The custom metric, denoted as Shape Error is used to check the variance in the corner positions predicted by the model scaled against the size of the table in meters. We use these custom metrics as they are more suited to how we actually use the results of this model

TABLE VII
RESULTS OF TABLE_VERTEX_DETECTION

Metric	Error
Overlap	0.6761
Shape Error	3.5038

B. Ball Position Detection

Here we use normal metrics such as Precision and Recall along with certain YOLO-specific metrics such as mAP50 and mAP95

TABLE VIII
RESULTS OF BALL_POSITION_DETECTION

Metric	Error
Precision	0.9043
Recall	0.8622
mAP50	0.9457
mAP50-95	0.6136

From these metrics we can understand that this model has a very high precision value and can accurately predict where the ball is. While it does have some false negative, the trajectory analysis module can interpolate and compensate for any missing data

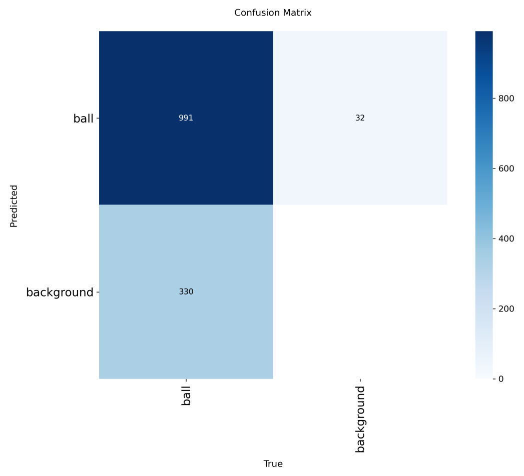

Fig. 9. Confusion Matrix for Ball Position Detection

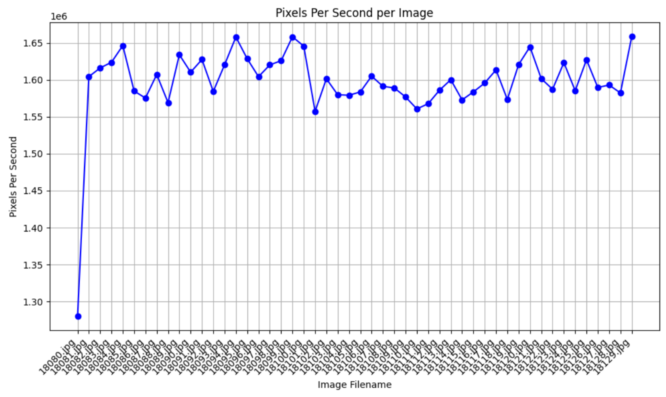

Fig. 10. Depth Anything V2 pixels processed per second

C. Depth Estimation

This metric shows the models processing throughput, scalability to higher resolution, hardware utilization and optimization on frequent access, the real time processing capability of the model. This shows that if the model is loaded it can process various image in that instance at low time making it suitable for video frames to 3d frame applications.

The detail score shows the capability of the model on how well it estimates the depth map while retaining fine structural details and edges. It is computed by the change in the spatial gradient values, with higher score meaning better detail preservation of the object boundary

The overall quality score indicates the evaluation of depth map by combing depth range, detail preservation, and consistency. This output shows the practical usefulness in the sense of maintaining higher detail and accuracy on the 3d depth map .

VII. FUTURE WORK

The Multi-Agent System proposed here operates on static, preconfigured blackboard architecture, which almost often

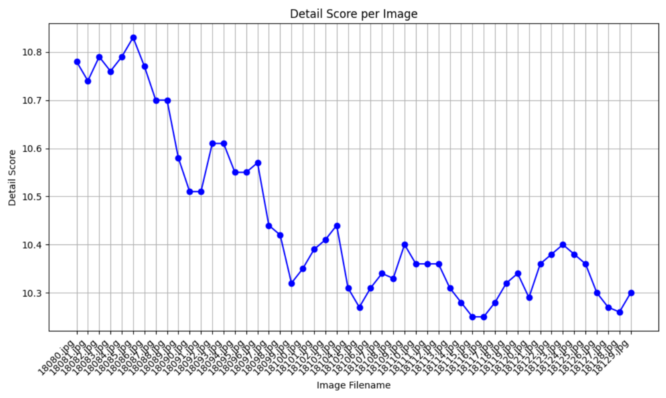

Fig. 11. Depth Anything V2 Detail score per image

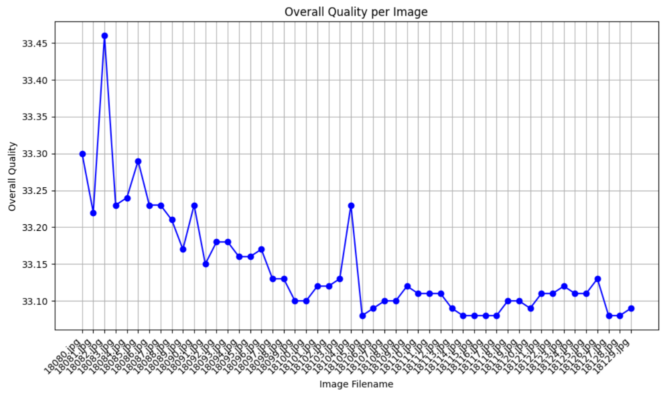

Fig. 12. Depth Anything overall quality per image

requires manual construction and thus lacks generality. Recent studies on LLM Based Multi-Agent System (LbMAS) [5] develop autonomous MASs by taking advantage of using a Large Language Model as the control unit. These LLMs can be used to dynamically configure new agents and their skills in runtime, make it much more general and requiring low to none manual setup. While, in theory, a static agent configuration is quicker and deterministic, the generality of a dynamic MAS outweighs those disadvantages. In an age of rapid Cloud and Distributed Computing evolution, our proposal could also be made to leverage remote knowledge sources/agents and remote computing resources over a distributed network. Recent studies [17] [2] show promising implementations of MASs with many agents running on different networks/systems. Our idea of a highly scalable MAS lies behind the ability to orchestrate and manage each type of knowledge source, individually, using a control unit of its own, taking inspiration from Kubernetes.

References

[1] Weilin Cai, Juyong Jiang, Fan Wang, Jing Tang, Sunghun Kim, and Jiayi Huang. A survey on mixture of experts. 06 2024.

[2] K.S. Ettabaa, I.R. Farah, B. Solaiman, and M.B. Ahmed. Distributed blackboard architecture for multi-spectral image interpretation based on multi-agent system. In *2006 2nd International Conference on Information Communication Technologies*, volume 2, pages 3070–3075, 2006.

[3] Silvio Giancola, Anthony Cioppa, Bernard Ghanem, and Marc Droogenbroeck. Deep learning for action spotting in association football videos. 10 2024.

[4] F. Haas, T. Baumgartner, T. Klein-Soetebier, F. Seifriz, and S. Klatt. Heatmap analysis to differentiate diverse player types in table tennis—a training and tactical strategy development potential. *Applied Sciences*, 13(2):1139, 2023.

[5] Bochen Han and Songmao Zhang. Exploring advanced llm multi-agent systems based on blackboard architecture. 07 2025.

[6] Barbara Hayes-Roth. A blackboard architecture for control. *Artificial Intelligence*, 26(3):251–321, 1985.

[7] C. H. Hung. A study of automatic and real-time table tennis fault serve detection system. *Sports*, 6(4):158, 2018.

[8] Chang-Hung Hung. A study of automatic and real-time table tennis fault serve detection system. *Sports*, 6:158, 11 2018.

[9] Robert Jacobs, Michael Jordan, Steven Nowlan, and Geoffrey Hinton. Adaptive mixtures of local experts. *Neural Computation*, 3:79–87, 03 1991.

[10] P. Kelly, J. D. P. M. Agapito, C. Conaire, D. Monaghan, J. Kuklyte, D. Connaghan, and N. O'Connor. A low-cost performance analysis and coaching system for tennis. In *Proceedings of ACM Multimedia*, 2010.

[11] K. M. Kulkarni, R. S. Jamadagni, J. A. Paul, and S. Shenoy. Table tennis stroke detection and recognition using ball trajectory data. *arXiv preprint arXiv:2302.09657*, 2023.

[12] Kaustubh Kulkarni, Rohan Jamadagni, Jeffrey Paul, and Sucheth Shenoy. Table tennis stroke detection and recognition using ball trajectory data. 02 2023.

[13] Siyuan Mu and Sen Lin. A comprehensive survey of mixture-of-experts: Algorithms, theory, and applications, 03 2025.

[14] H. P. Nii. The blackboard model of problem solving and the evolution of blackboard architectures. *AI magazine*, 7(2):38, 1986.

[15] H. Penny Nii. Blackboard systems at the architecture level. *Expert Systems with Applications*, 7(1):43–54, 1994.

[16] Puntawat Ponglertnapakorn and Supasorn Suwajanakorn. Where is the ball: 3d ball trajectory estimation from 2d monocular tracking. pages 6122–6131, 2025.

[17] Shiva Sai Krishna Anand Tokal, Vaibhav Jha, Anand Eswaran, Praveen Jayachandran, and Yogesh Simmhan. Agentx: Towards orchestrating robust agentic workflow patterns with faas-hosted mcp services, 2025.

[18] R. Voeikov, N. Falaleev, and R. Baikulov. Ttnet: Real-time temporal and spatial video analysis of table tennis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pages 884–885, 2020.

[19] Roman Voeikov, Nikolay Falaleev, and Ruslan Baikulov. Ttnet: Real-time temporal and spatial video analysis of table tennis. pages 3866–3874, 06 2020.

[20] G. L. Yang, M. Nguyen, W. Q. Yan, and X. J. Li. Foul detection for table tennis serves using deep learning. *Electronics*, 14(1):27, 2024.

[21] Guang Liang Yang, Minh Nguyen, Wei Qi Yan, and Xue Jun Li. Foul detection for table tennis serves using deep learning. *Electronics*, 14(1), 2025.