NATIONAL UNIVERSITY
of Computer & Emerging Sciences

BATCH 2

# DRS Module Specifications

## Module 1:

**Mobile UI (Camera Module) + Integration of all Modules**

## Group members:

Aina Aroob-22L-8221

Ali Kamal-22L-6817

Ayesha Asmat-22L-6839

Hafsa Feroze-22L-6636

### 1. Description

This module is responsible for capturing the live video feed of the cricket pitch using the mobile device's camera. The frames are accessed via the mobile application's interface and sent in real-time to downstream processing modules, specifically the Ball and Object Tracking Module. It also serves as the integration point for all app modules, facilitating communication and data flow between components.

### 2. Functionalities

- Captures live video of the cricket pitch.
- Streams individual frames in real-time.
- Acts as a central hub to integrate with the tracking and analytics modules.
- Ensures low-latency frame delivery.

### 3. Input

- Camera feed from mobile device (live video stream).

### Output

- Video frames in real-time (usually in RGB or YUV format depending on device).
- Frames are sent to the Ball and Object Tracking Module for further analysis.

## 5. Interfaces

| Interface | Description |
|---|---|
| Camera Hardware API | Captures raw video data from the device. |
| Frame Preprocessing API (optional) | Performs resizing, filtering, or conversion of frames. |
| Data Transmission Interface | Sends the frames to the next processing module.<br><br>(usually via a socket, shared memory, or internal method call). |
| UI Interface | Allows user to start/stop the camera and view live feed. |

## 6. Dependencies

- Mobile device with functional camera.
- Mobile OS APIs (e.g., Android CameraX or iOS AVFoundation).
- Stable network/memory buffer for transmitting frames to other modules (if distributed architecture).

## 7. Implementation Flow

The steps listed below describe the module's detailed operation:

**A. Initialize Camera:**

- Use CameraX (Android) or AVFoundation (iOS) for live feed.
- Resolution: Preferably 720p/1080p @ 30 FPS.

**B. Extract Frames:**

- Convert stream to individual frames.
- Add timestamps and frame IDs for reference.

**C. Frame Dispatch:**

- Send frames to Ball/Object Tracking module via socket, API, or in-memory queue.
- Include metadata (time/frame index) for syncing with physics-based modules.

**D. UI Display:**

- Stream video live in app.
- Display edge detection overlays via integration with Stream Analysis and Overlay Module.

## 8. Conclusion

The foundation of the Decision Review System (DRS) pipeline is the Mobile User Interface (Camera Module). It establishes the basis for precise, time-synchronized analysis across the DRS system by recording high-resolution live video, extracting individual frames with timestamps, and sending them smoothly to the Ball and Object Tracking Module.

Each frame is prepared for further operations including bat's edge recognition, trajectory analysis, and visual overlay rendering. In the end, this module provides the crucial data foundation for highly accurate decision-making in cricket matches while guaranteeing a responsive and seamless user experience.

# Module 2: Ball and Object Tracking Module – Interface Specification

**Group :**

**Areeb Shahbaz 22L-6580**
**M Haseeb 22L-6577**
**Aun Noman 22L-6950**
**Saqib Ali 22L-6781**
**Awais Amir 22L-6966**

## Overview

This module is responsible for detecting and tracking key objects from video frames captured by the mobile UI. It uses computer vision and/or deep learning models to identify:
- Ball: 3D tracking over time (trajectory)
- Bat: Position and orientation
- Batsman's Legs: Bounding boxes or keypoints + leg orientation
- Stumps: 3D positions (static or dynamically adjusted if needed)

It outputs structured positional and motion data required by downstream modules (primarily Bat’s Edge Detection Module).

## Responsibilities

- Receive sequential video frames with timestamps
- Detect and assign identities to relevant objects
- Estimate 3D positions (x, y, z) using stereo vision or monocular depth estimation
- Estimate motion vectors and orientation (if applicable)
- Maintain temporal context across frames

## Input Specification

Input Source: Module 1: Mobile UI (Camera Module)

**Data Format:**

- Frame Rate: Minimum 30 FPS for smooth trajectory analysis
- Resolution: Minimum 720p; ideal 1080p or above
- Encoding: Base64-encoded frame, or reference path if video is streamed internally

**JSON Input Example:**

```
{
  "frame_id": 1042,
  "timestamp": "2025-04-07T13:42:00.123Z",
  "camera_metadata": {
    "resolution": "1920x1080",
    "fps": 30,
    "camera_position": "umpire_view",
    "focal_length": 1.8
  },
  "image_data": "base64_encoded_frame_string"
}
```

## Output Specification

Output Targets: Module 3: Bat's Edge Detection Module

**Output Description:**

- 3D Ball trajectory (x, y, z) per frame
- Object metadata (bounding boxes, confidence scores)
- Orientation vectors (where applicable)

**JSON Output Example (Per Frame):**

```
{
  "frame_id": 1042,
  "timestamp": "2025-04-07T13:42:00.123Z",
  "detections": {
    "ball": {
      "position_3d": {"x": 2.45, "y": 1.03, "z": 0.2},
    },
    "bat": {
      "position_3d": {"x": 2.50, "y": 1.10, "z": 0.20},
      "orientation": {"pitch": 12, "yaw": 40, "roll": 0}
    },
    "batsman_leg": {
      "keypoints": [{"x": 2.3, "y": 1.0}, {"x": 2.4, "y": 1.1}],
      "bounding_box": {"x1": 2.2, "y1": 0.9, "x2": 2.5, "y2": 1.3},
      "orientation_angle": 28,

    },
    "stumps": [
      {"x": 0.0, "y": 0.0, "z": 0.0},
      {"x": 0.1, "y": 0.0, "z": 0.0},
      {"x": -0.1, "y": 0.0, "z": 0.0}
    ]
  }
}
```

## Module 3

## Module Name:

Bat’s Edge Detection Module

## Group Members:

MUHAMMAD BAASIL 22L - 6674

ASJAD SIDDIQUI 22L - 6602

HAFSA WAJID BUTT 22L - 6738

QAINAT SAEED 22L - 6569

IMRANA DILSHAD   24L - 7832

## Bat's Edge Detection Module

### Introduction

The **Bat's Edge Detection Module** is a crucial component of the **Decision Review System (DRS)** in cricket. Its primary role is to determine whether the cricket ball makes contact with the **bat’s edge** during play—a critical element in decisions such as **Caught Behind** and **Leg Before Wicket (LBW)**.

Developed using **Python** and **NumPy**, the module operates in real-time as part of a larger decision pipeline. It combines **spatial analysis** of 3D tracking data with **synchronized audio signals** to detect edge contact events. Upon detection, the module either updates the ball’s trajectory or forwards unmodified data for further analysis, ensuring **high-precision and timely decisions** in match-critical scenarios.

---

### Objective

The main objectives of the Bat's Edge Detection Module are to:

- **Detect edge contact** between the ball and the bat using combined spatial proximity and audio spike analysis.

- **Update the ball's trajectory** if edge contact is detected, reflecting the change in direction due to deflection.

- **Forward unaltered trajectory data** for LBW analysis when no bat contact is identified.

---

### Module Description

The module acts as a **decision-making node** in the DRS pipeline. It receives and processes **multi-modal data**, including:

- **3D trajectory coordinates** of the ball
- **Object metadata** (bat and player positions, orientation)
- **Audio waveform signals** indicating impact events

By analyzing this data, the module determines whether an edge contact occurred. Based on the outcome, it:

- **Updates and reroutes** data to the **Stream Analysis and Overlay Module** if contact is confirmed
- **Forwards original data** with leg position info to the **Trajectory Analysis Module** for LBW evaluation if no contact is found

---

## Core Functions

### 1. Edge Detection Logic

- Computes **spatial proximity** between the ball and bat using 3D positional and orientation data.
- Monitors for **velocity anomalies or sharp deflections**, which suggest physical interaction.

- Utilizes **timestamp-synced audio waveform** to detect short, high-intensity spikes typical of ball-bat contact.

**2. Trajectory Update (on contact)**

- Alters the ball's **velocity vector** to reflect post-contact direction (`[vx', vy', vz']`).
- Identifies the **precise contact point** in 3D space.

**3. Routing and Decision Logic**

- If edge contact is detected:
  - Forwards the **updated trajectory** and **contact metadata** to the **Stream Analysis and Overlay Module**
- If no contact is detected:
  - Sends the **original trajectory**, **velocity**, and **leg position data** to the **Trajectory Analysis Module** for LBW analysis

---

## Interfaces

### Input Interfaces

**From: Ball and Object Tracking Module**

- `ball_trajectory`: 3D coordinates of the ball per frame

- `velocity_vectors`: Ball's motion vector per frame
- `object_metadata`: Bounding boxes and confidence scores for bat, batsman, etc.
- `orientation_vectors`: Directional alignment of bat and objects

**From: Audio Sensors**

- `audio_waveform`: Synchronized audio signals to detect ball-bat impact

**Output Interfaces**

**If Edge is Detected**

- **To: Stream Analysis and Overlay Module**
    - `updated_velocity`: New velocity vector `[vx', vy', vz']`
    - `contact_point`: Coordinates of bat-ball contact `(x, y, z)`
    - `decision_flag`: `(True, contact_point)`

**If No Edge is Detected**

- **To: Trajectory Analysis Module**
    - `original_trajectory`: Original 3D path and velocity

- `leg_position_data`: Batting stance/leg position (for LBW logic)

- `decision_flag`: `(False, None)`

---

## Input and Output Data Format

### Input Data

python

CopyEdit

- `ball_trajectory = [(x1, y1, z1), (x2, y2, z2), ...]`
- `velocity_vectors = [(vx1, vy1, vz1), (vx2, vy2, vz2), ...]`
- `object_metadata = {`
- `    "bat": {"bbox": [...], "confidence": ...},`
- `    "batsman": {"bbox": [...], "confidence": ...}`
- `}`
- `orientation_vectors = {`
- `    "bat": (ox, oy, oz),`
- `    ...`
- `}`
- `audio_waveform = [audio_sample1, audio_sample2, ...] # Time-aligned`

### Output Data

**Case 1: Edge Detected**

python

CopyEdit

- `updated_velocity = [vx', vy', vz']`
- `contact_point = (x, y, z)`
- `decision_flag = (True, contact_point)`
- `# Destination: Stream Analysis and Overlay Module`

**Case 2: No Edge Detected**

python

CopyEdit

- `original_trajectory = [(x1, y1, z1), (x2, y2, z2), ...]`
- `leg_position_data = (lx, ly, lz)  # From tracking module`
- `decision_flag = (False, None)`
- `# Destination: Trajectory Analysis Module`

# Module 4

# Trajectory Analysis Module

## Group Members

| | |
|---|---|
| Taha Abdullah | 22L-6566 |
| Farrukh Awan | 22L-6701 |
| Tahnan Aamir | 22L-6711 |
| Ali Shahbaz | 22L-6746 |
| Usman Khalil | 22L-6873 |

## Overview

Our module forms a vital part of an intelligent ball-tracking and decision-assistance platform tailored for cricket analytics. Its main purpose is to analyze a cricket ball’s movement after it bounces, forecasting its future path to determine if it would have struck the stumps—particularly useful in leg-before-wicket (LBW) evaluations.

This system is developed through extensive testing and refinement, aiming to deliver near real-time support for decision-making in the game of cricket.

## Research Foundation

To build a reliable and accurate prediction system, our team explored various interrelated scientific and technical domains. These research areas helped us fine-tune our approach to better reflect real-world match conditions.

### 1. Pitch Surface Modeling

Different types of pitches (dry, damp, green tops) influence ball bounce and spin differently. We integrated surface behavior into our impact prediction to handle a variety of playing environments.

### 2. Player Movement Analysis

We studied common LBW scenarios and the typical biomechanics of batters, including footwork and stance, to make sure our system can respond accurately to realistic and edge-case batting situations.

### 3. Analyzing Cricket Ball

We delved into technical resources on sports physics to analyze how different factors—such as angular velocity (spin), seam orientation, ball-surface interaction, drag forces, and wind conditions—affect a cricket ball's post-bounce behavior.

## Data Interface

### Input Parameters

```
{
  "trajectory": [
    {"pos_x": 2.3, "pos_y": 1.1, "pos_z": 0.5, "timestamp": 0.0},
    {"pos_x": 2.1, "pos_y": 0.9, "pos_z": 0.4, "timestamp": 0.1},
    ...
  ],
  "velocity_vector": [1.1, -0.7, 0.4],
  "leg_contact_position": {"x": 2.0, "y": 0.7, "z": 0.35},
  "edge_detected": false,
  "decision_flag": [false, null]
}
```

**trajectory**: Series of position-time data points from tracking system before pad impact

**velocity_vector**: Ball's velocity at or just before pad contact

**leg_contact_position:** Ball's velocity at or just before pad contact

**edge_detected:** Boolean, always false in this context

**desicion_flag:** Indicates edge status; passed as (false, null) to confirm no bat contact

## Output

```
{
  "result_id": "res1",
  "predicted_path": [
    {"pos_x": 2.3, "pos_y": 1.1, "pos_z": 0.5, "timestamp": 0.0},
    {"pos_x": 2.1, "pos_y": 0.9, "pos_z": 0.4, "timestamp": 0.1}
  ],
  "verdict": {
    "status": “Out",
    "will_hit_stumps": true,
    "impact_region": "middle",
    "confidence": 0.85
  }
}
```

## System Info

### Core Functionality

1. Intake trajectory and impact information from the data capture unit.
2. Predict post-bounce path using physical modeling and machine learning algorithms.
3. Deliver results in a clear, structured format with confidence levels.
4. Relay outputs to visualization and event-analysis modules.
5. Assess whether the ball would have collided with the stumps.

### What It has Access To

1. Control over parameter selection such as drag values and spin impact coefficients.
2. Direct access to cleaned and pre-processed trajectory data.

# 5. Decision Making Module

**Group Name: Tekila**

**• 22L-6741 Abdul Moiz**

**• 22L-6851 Hassan Musa**

**• 22L-6576 Danyal Rehman**

**• 22L-6915 Tayyab Ali**

**• 22L-6986 Muhammad Ans Jamal**

## Decision Making System

### 1. Bat's Edge Detection Module Introduction

**The Bat's Edge Detection Module is a core component of the Decision Review System (DRS) in cricket, designed to enhance umpiring accuracy by analyzing ball-bat interactions. Integrated within a modular pipeline, this module processes real-time data to determine if the cricket ball contacts the bat's edge—a pivotal factor in decisions like Leg Before Wicket (LBW). Leveraging Python and NumPy, it combines object tracking data with physics-based calculations to either update the ball's trajectory or pass data for further analysis, ensuring precise and reliable outcomes in high-stakes matches.**

### 2. Objective

**The primary objective of the Bat's Edge Detection Module is to:**

- **Detect whether the ball makes contact with the bat's edge using spatial proximity analysis.**
- **Update the ball's trajectory if contact occurs, reflecting the impact's effect.**

**• Forward original data to the next module if no contact is detected, particularly when the ball hits the batsman's leg, enabling LBW evaluation.**

## 3. Module Description

**This module receives positional and velocity data from the Ball and Object Tracking Module, processes it to identify edge contact, and routes the results to either the Stream Analysis and Overlay Module (if contact is detected) or the Trajectory Analysis Module (if no contact occurs).**

**Key Functions:**

- **Detects ball-bat contact based on positional data and impact analysis.**
- **Updates ball trajectory in case of edge contact.**
- **Sends decisions to subsequent modules for further processing.**

## 4. Interfaces

**The Bat's Edge Detection Module interacts with multiple components in the DRS system:**

### Input Interfaces

**• Ball and Object Tracking Module: Provides ball's trajectory data (x, y, z coordinates), bat position, and batsman's stance.**

**• Audio Sensors: Captures sound waves to detect ball-bat impact.**

### Output Interfaces

**• Trajectory Analysis Module: If no bat edge is detected and the ball hits the batsman's leg, data is sent for further processing.**

**• Stream Analysis and Overlay Module: Updates the live video feed with graphical representations of contact or no-contact results.**

## 5. Input and Output Data

### Input Data

**• Ball Trajectory: Array of 3D coordinates (x, y, z) tracking the ball's path.**

**• Bat Edges: Array of 3D coordinates defining the bat’s edge positions.**

**• Ball Velocity: 3D vector representing the ball’s initial velocity.**

**• Bat Normal: 3D vector indicating the bat’s surface orientation at the contact point.**

**• Source: Ball and Object Tracking Module.**

Output Data

Case 1: Edge Detected

**• Updated Ball Trajectory: New velocity vector post-reflection.**

**• Contact Point: 3D coordinates of the detected contact.**

**• Destination: Stream Analysis and Overlay Module.**

Case 2: No Edge Detected

**• Original Ball Trajectory: Unmodified coordinates and velocity.**

**• Additional Data: Leg position data if the ball hits the batsman’s leg.**

**• Destination: Trajectory Analysis Module.**

**• Decision Flag: Tuple (True, contact_pos) for edge contact, (False, None) for no contact.**

## 6. Workflow of the Bat's Edge Detection Module

- **1. Input Data Reception: Ball trajectory and bat position details are received.**
- **2. Contact Detection: Distance is calculated between the ball and bat edges.**
- **3. Confirmation via Audio: Sound wave data is optionally checked for impact evidence.**
- **4. Trajectory Adjustment: New trajectory is computed using physics if contact is detected.**
- **5. Decision Transmission: Results sent to the appropriate next module.**
- **6. Final Visualization: Overlays displayed on stream output.**
- **7. Decision Making (Trajectory Analysis) Module**

**This module receives output from the Bat's Edge Detection Module. If no bat contact is detected, it predicts the future trajectory of the ball based on spin, drag, and bounce to check whether it would**

hit the stumps.
It plays a key role in LBW decisions and uncertain ball trajectory scenarios.

**Input and Output Scenarios**

**Scenario 1: Ball Hits Bat Edge**

- Input: Ball contact confirmed by edge detection.
- Output: Not Out, trajectory not further analyzed.
- Sent to: Stream Overlay Module.

**Scenario 2: Ball Misses Bat but Hits Leg**

- Input: No bat contact, leg contact confirmed.
- Processing: Predict trajectory using physics.
- Output: Out (if trajectory hits stumps) or Not Out.
- Sent to: Stream Overlay Module.

**Scenario 3: Ball Misses Both Bat and Leg**

- Input: No contact with bat or leg.
- Processing: Predict trajectory.
- Output: Out (if trajectory hits stumps) or Not Out.
- Sent to: Stream Overlay Module.

## 8. Conclusion

The Bat's Edge Detection and Trajectory Analysis Modules work together to ensure accurate cricket decisions based on ball interaction. By leveraging advanced tracking and prediction, the system ensures reliable outcomes for LBW and edge reviews, significantly enhancing the fairness and precision of match officiating.

## Decision Making (Trajectory Analysis) Module

Inputs:{"ball_trajectory": [

    {"x": 0.12, "y": 1.52, "z": 0.85},{"x": 0.15, "y": 1.48, "z": 0.80},

    ...],

**"batsman_position": {"leg_contact": true,"leg_position": {"x": 0.14, "y": 1.50, "z": 0.82}},"stump_position": {"x": 0.0,"y": 0.0,"z": 0.71},"spin": 12.5,"bounce": 0.32,"drag_coefficient": 0.85}**

**Output:{"predicted_trajectory": [**

**{"x": 0.17, "y": 1.45, "z": 0.76},{"x": 0.20, "y": 1.40, "z": 0.71},... ],**

**"would_hit_stumps": true,"confidence_score": 0.92}**

**<u>Ball Hits Bat Edge</u>**

**Inputs:{"ball_trajectory": [**

**{"x": 0.12, "y": 1.52, "z": 0.85},{"x": 0.15, "y": 1.48, "z": 0.80}],**

**"bat_contact": true,"bat_contact_position": {"x": 0.14, "y": 1.49, "z": 0.81},"bat_orientation": {"angle": 35.5},"stump_position": {"x": 0.0, "y": 0.0, "z": 0.71}}**

**Output:{"decision": "Not Out","reason": "Ball hit bat edge","predicted_trajectory": [],"would_hit_stumps": null, "confidence_score": 0.98}**

**<u>Ball Misses Bat and Hits Leg</u>**

**Input:{"ball_trajectory": [{"x": 0.20, "y": 1.48, "z": 0.83},{"x": 0.23, "y": 1.44, "z": 0.79}],**

**"bat_contact": false,"leg_contact": true, "leg_position": {"x": 0.22, "y": 1.43, "z": 0.77},"stump_position": {"x": 0.0, "y": 0.0, "z": 0.71}, "spin": 8.5, "bounce": 0.25, "drag_coefficient": 0.76}**

**Output:{**

**"decision": "Out","reason": "Ball would have hit stumps after hitting leg","predicted_trajectory": [{"x": 0.25, "y": 1.40, "z": 0.74},{"x": 0.28, "y": 1.36, "z": 0.71}],**

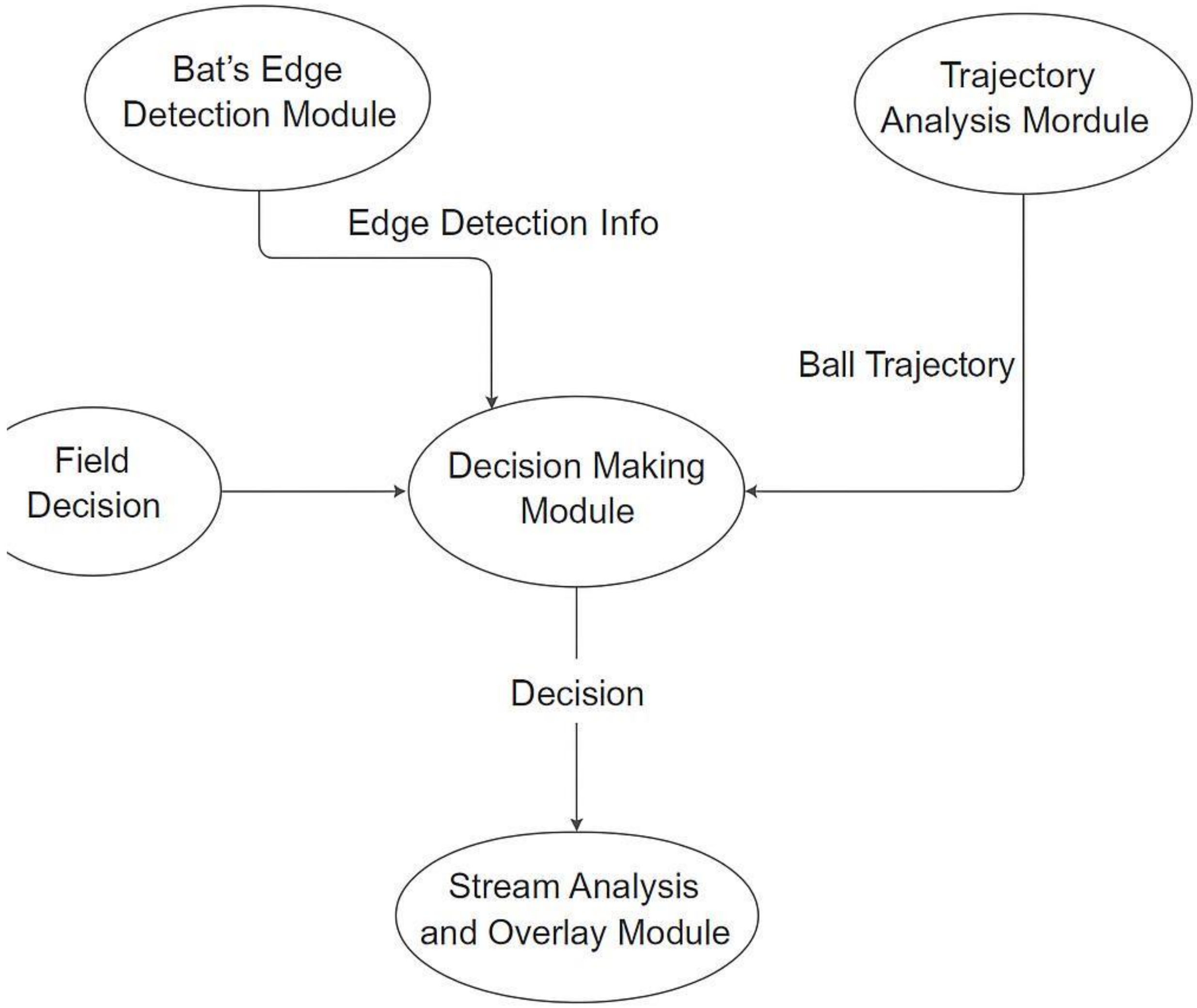
**"would_hit_stumps": true,"confidence_score": 0.89}**

**<u>Ball Misses Bat and Leg</u>**

**Input:{ "ball_trajectory": [**

**{"x": 0.10, "y": 1.50, "z": 0.85},**

```
            {"x": 0.13, "y": 1.45, "z": 0.80}],

  "bat_contact": false, "leg_contact": false,  "stump_position": {"x": 0.0, "y": 0.0, "z": 0.71},"spin":
5.5, "bounce": 0.28, "drag_coefficient": 0.70}

Output:{"decision": "Not Out","reason": "Ball did not hit bat or leg",

  "predicted_trajectory": [

            {"x": 0.16, "y": 1.40, "z": 0.74},

            {"x": 0.20, "y": 1.35, "z": 0.69}],

  "would_hit_stumps": false,"confidence_score": 0.9}
```


Bat's Edge Detection Module
Trajectory Analysis Mordule
Edge Detection Info
Ball Trajectory
Field Decision
Decision Making Module
Decision
Stream Analysis and Overlay Module

# Module 6

## Batch 2 (Group 10)

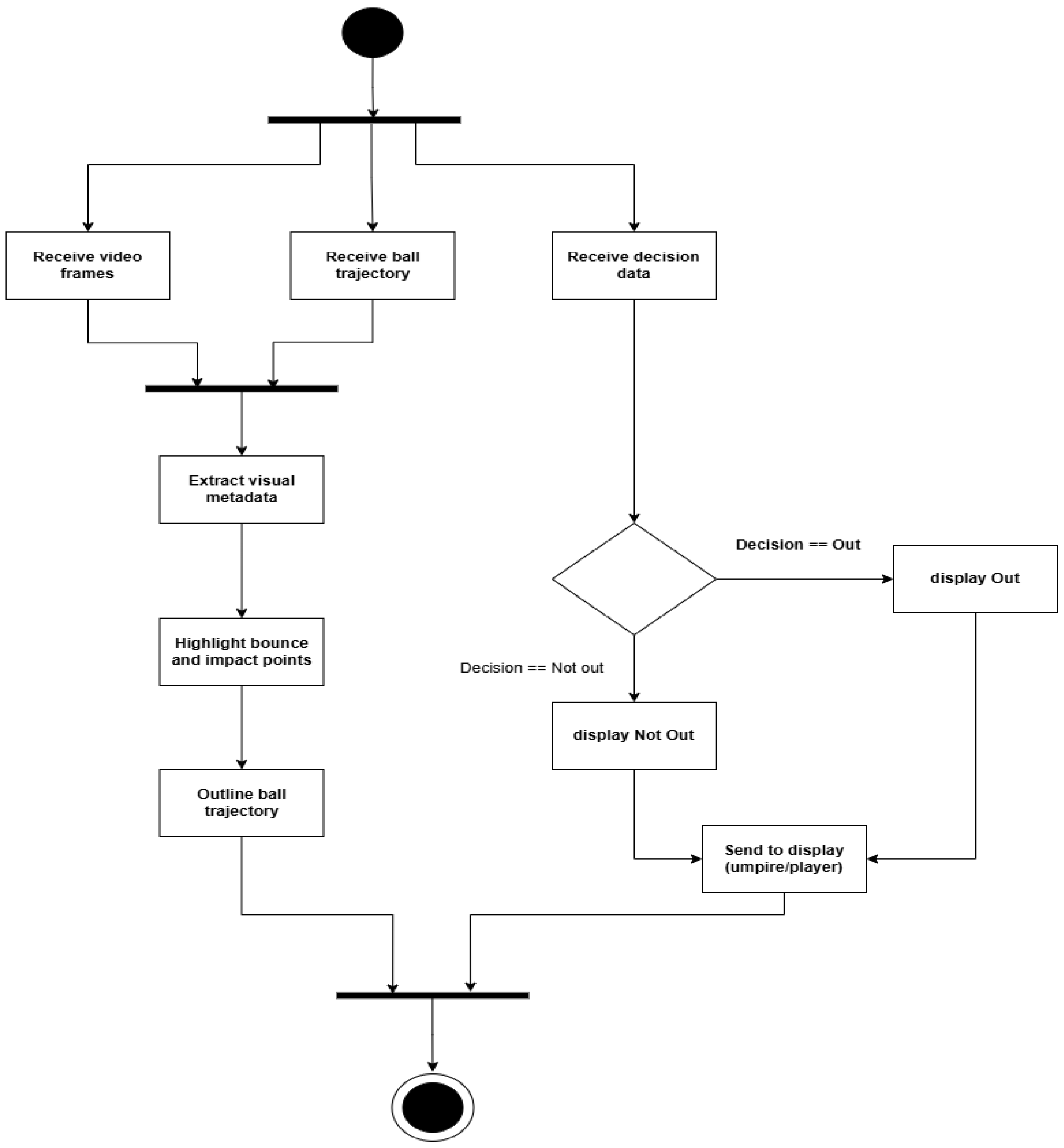
## Module: Stream Analysis and Overlay Module

**Group members:**
Arham Jahangir 22I-6613
Laiba Naseer 22I-6616
Muhammad Umer 22I-6790
Kainat Nasir 22I-6784

## Module Description:

The Stream Analysis and Overlay Module enhances the original video feed by adding graphical overlays such as ball trajectory lines, impact points, edge detection indicators, and final decision texts to produce a more user-friendly result.

## Tentative Activity Diagram:



## Module Responsibilities:

This module must accept input from previous modules and add the following visual elements on the video:

- Ball trajectory lines.
- Make the video visually appealing to the user.
- Bounce and impact markers.
- Edge detection visuals (if edge was detected).
- Final decision display (e.g., “Out”, “Not Out”, “Edge Detected”).

It must also ensure all overlays are synchronized with the correct video frames and render the final augmented video stream to be shown to the umpire or audience. Doing all this in minimal processing time is essential for this module.

## Interfaces:

### 1. Input Interface

**Source Modules**

1. Decision Making Module (Module 5)
2. Camera Module (Module 1)

**Inputs Received**

1. From Module 5
    - Final decision (e.g. “Out”, “Not Out” )
    - Trajectory data (coordinates of the predicted path)
    - Impact location
    - Bounce point and swing characteristics
    - Metadata for visual elements (colors, labels, position on screen)

2. From Module 1
    - Recorded video frames

**Example JSON Input:**

```
{
  "video_frame": "base64_encoded_image_data",
  "decision": "Out",
  "trajectory": [
    {"x": 120, "y": 250, "z": 0},
    {"x": 122, "y": 248, "z": 1},
    {"x": 125, "y": 245, "z": 2}
  ],
  "impact_point": {"x": 125, "y": 245},
  "bounce_point": {"x": 115, "y": 255},
  "swing": "in-swing",
  "overlay_metadata": {
    "trajectory_color": "red",
    "decision_text": "OUT",
    "highlight_impact": true
  }
}
```

## 2. Output Interface

The output would be displayed to players and umpire

**Output Generated:**

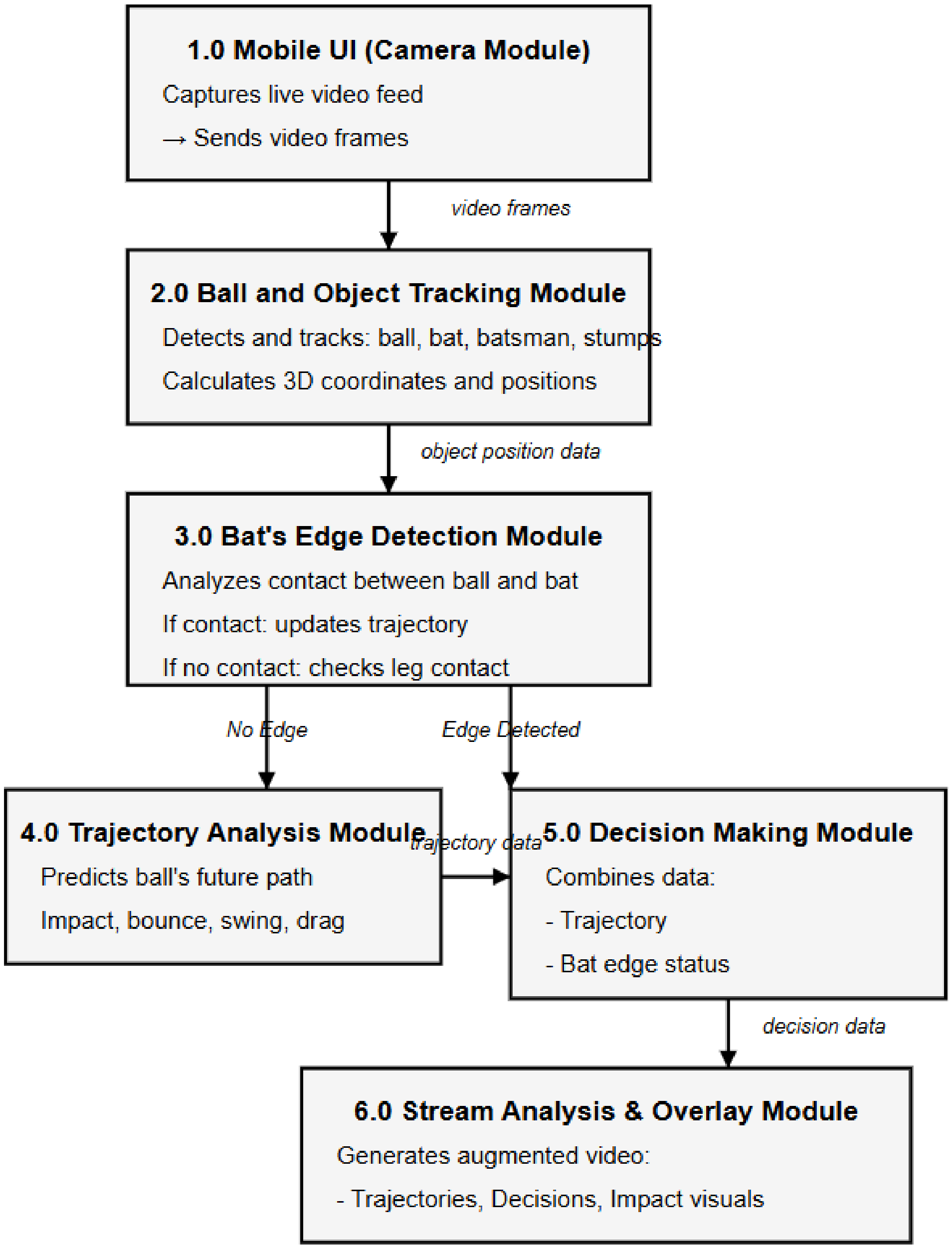
Augmented video frames with:

- Ball trajectory lines are drawn in real time
- Highlighted bounce and impact points
- Decision text (e.g., “OUT”, “NOT OUT”)
- Optional edge flash animation

**Example JSON Output:**

```
{
  "augmented_video_frame": "base64_encoded_augmented_frame",
  "overlays": [
    {
      "type": "trajectory",
      "coordinates": [[120, 250], [122, 248], [125, 245]],
      "color": "red"
    },
    {
      "type": "impact_marker",
      "position": [125, 245],
      "highlight": true
    },
    {
      "type": "decision_text",
      "text": "OUT",
      "position": [50, 50],
      "style": "bold-yellow"
    }
  ]
}
```

## High level view of system


1.0 Mobile UI (Camera Module)
Captures live video feed
→ Sends video frames
video frames
2.0 Ball and Object Tracking Module
Detects and tracks: ball, bat, batsman, stumps
Calculates 3D coordinates and positions
object position data
3.0 Bat's Edge Detection Module
Analyzes contact between ball and bat
If contact: updates trajectory
If no contact: checks leg contact
No Edge
Edge Detected
4.0 Trajectory Analysis Module
Predicts ball's future path
Impact, bounce, swing, drag
trajectory data
5.0 Decision Making Module
Combines data:
- Trajectory
- Bat edge status
decision data
6.0 Stream Analysis & Overlay Module
Generates augmented video:
- Trajectories, Decisions, Impact visuals