Documentation

Frame setup and Virtual Screen:

- Used a 16:9 aspect ratio.
- Frame size: 38.22 $_{in}$(16*2.388) X 21.5 $_{in}$(9*2.388) .
- The system is designed in style that it works with any frame size with aspect ratio of 16:9.
- Virtual Screen is also at aspect ratio of 16:9.
- Proper Scaling and real-time feedback is maintained in the virtual screen.

Camera Setup:

- Using a Webcam directed towards the framed area.
- For the demonstration video purpose, I've used an already recorded video (because of the laptop instability).
- The system tested with both video and webcam compatibility.

Lighting Robustness:

- Ensure the system works under various lighting conditions without compromising hit detection accuracy,
- This being maintained by using HSV color filtering wide and accurate range ensuring the object detection does not get affected by lighting conditions.

Strategies to Improve Accuracy in Tennis Ball Detection System

This document outlines the strategies implemented to M**inimize noise**, **Reduce false positives**, **Accurately identify the exact point of impact**, and **Ensure high accuracy in distinguishing actual hits** from near misses or other movements. The code utilizes **OpenCV** and Python to capture video, detect impacts within a defined boundary, and map these impacts onto a virtual screen.

1. Minimizing Noise and False Positives

To ensure that the system reliably detects the ball and ignores irrelevant movements or objects, several noise reduction and filtering techniques are employed:

1. **Color Filtering with HSV Color Space**:
    - The system uses HSV color filtering to detect both the ball and the taped border. This is because HSV allows for more robust color filtering compared to the RGB color space, especially under varying lighting conditions.

    - Specific HSV ranges are set for the ball and the taped border. Adjusting these ranges helps filter out objects that do not match the desired colors.
2. **Contour Area Filtering**:
    - To avoid false positives caused by small, irrelevant contours, a minimum area threshold is set for detecting the ball. Only contours larger than a specific size are considered, helping to ignore noise such as shadows or small objects.
3. **Minimum Movement Threshold**:
    - The system tracks the ball's position and velocity between frames to differentiate real movements from jitter. A minimum velocity threshold ensures that only significant movements, like a ball hit, are considered, reducing noise from minor, irrelevant movements.

2. Accurate Identification of Impact Points

1. **Velocity Change Detection for Hits**:
    - A hit is detected when there is a significant change in the ball's velocity, typically indicating an impact. The system calculates the ball's velocity in each frame and compares it to the previous velocity to detect sudden changes.
2. **Boundary Validation Using Contours**:
    - The system uses the taped border's contour to validate whether the detected impact point lies within the defined boundary. This validation ensures that only hits within the taped area are considered, ignoring impacts outside the intended zone.
    - Contour area filtering
3. **Scaling Impact Points to the Virtual Screen**:
    - Impact points detected on the physical frame are mapped onto a virtual screen that maintains the same aspect ratio of 16:9. This mapping ensures that the virtual representation accurately reflects the real-world impact location.
4. **For correct Ball Detection**:

    - Instead of using HoughCircle() method which gives lesser accuracy in providing the circles and identifying the ball. I've been using HSV color filtering for tennis ball color range. This method provide a better accuracy.

3. Ensuring High Accuracy in Distinguishing Actual Hits from Near Misses or Other Movements

To improve accuracy, the following methods are used to differentiate genuine ball hits from near misses or irrelevant movements:

1. **Consistent Hit Validation**:
    - Only impacts that show consistent motion changes over multiple frames are considered. This approach reduces false positives from momentary noise or sudden movements unrelated to a real hit.

2. **False Hit Filtering Inside the Boundary**:
    - Hits are only registered if they occur inside the detected taped boundary. This ensures that any impact outside the target area is ignored, focusing the detection solely on valid zones.
3. **Real-Time Feedback and Debugging**:
    - The code includes real-time feedback on detected impacts, displaying the position of hits on both the physical and virtual screens. This aids in debugging and allows for immediate adjustment of detection parameters if inaccuracies are observed.
4. **Ensuring the ball hits inside the frame:**
    - Here I used a method called **cv2.pointPolygonTest()** which ensures weather a given point/ball lies inside or outside of a polygon.

Background compatibility

Considering background compatibility, I've used two different methods:

1. **To detect the Tapped boundary:**
    - It ensures to check if any colored tapped (Black specifically) for the setting the boundary of the screen, by help of hsv color.
    - This ensure the boundary and check if the ball hitting outside or inside the boundary.
2. **To detect the White Background:**
    - It ensures to check if any white colored background (ensuring two different tone of color) for the setting the frame of the screen, by help of hsv color.
    - This ensure the frame and check if the ball hitting outside or inside the framed area.
    - **If you use colored tape on white background, then it still going to work.**

Conclusion

The implemented strategies work in conjunction to minimize noise, reduce false positives, and accurately detect ball impacts within a defined boundary. By using advanced filtering, validation checks, and real-time feedback, the system ensures reliable detection of actual hits while ignoring irrelevant movements. This approach enhances the accuracy of the virtual mapping, creating a highly interactive and precise representation of the physical interactions on the virtual screen.

Created by: **Ankush Gupta, DSEU Okhla-2, CSE.**