Tennis Ball Boundary Detection Using YOLO and OpenCV

Applicable for both Live Camera and Video modes

MANOSORN Chanutda
Bachelor of Computer Science,
Griffith University
Gold Coast, Australia
chanutda.manosorn@griffithuni.edu.au

***Abstract*— This project presents a system for detecting and tracking tennis balls within a defined boundary area using YOLO (You Only Look Once) and OpenCV. The primary objective is to determine whether tennis balls remain within the court's boundaries. Addressing challenges related to detecting small objects and environmental sensitivities, the project integrates object recognition with boundary validation techniques. The system combines YOLOv8s and pre-trained models for tennis ball detection with OpenCV's Canny edge detection and morphological operations for boundary detection. Experiments demonstrate the system's capability to accurately detect tennis balls and court lines in video feeds, highlighting its potential for future improvements and applications in tennis analytics.**

I. Introduction

In sports analytics, accurately detecting and tracking objects such as tennis balls within the confines of a playing area is crucial for performance analysis and enhancement. The objective of this project is to develop a program that can detect tennis balls, track their movement, and determine whether they remain within the tennis court's boundaries. By integrating object recognition with boundary validation, this system aims to enhance functionality beyond existing models.

The primary challenge is to create a system capable of detecting small objects like tennis balls within a large area, such as a tennis court, and accurately determining their position relative to the court boundaries. Additionally, the system should be able to track the tennis balls' movement and speed to assess boundary violations. While existing open-source models like various versions of YOLO are proficient in object detection and recognition [1][2], YOLO has been chosen for the tennis ball detection task. This project seeks to extend these capabilities by integrating YOLO with OpenCV's edge detection and morphological operations to focus on boundary detection.

Due to time constraints and computational limitations, the project focuses on implementing a single, suitable version of YOLO in combination with OpenCV's edge detection techniques. Detecting small objects like tennis balls poses significant challenges, particularly when using standard cameras in highly controlled environments, which can greatly affect detection accuracy. Therefore, the project emphasizes optimizing performance within these constraints, with the detection of object speed considered beyond the project's current scope.

II. Previous Work

Object detection and boundary identification are essential aspects of computer vision, with extensive research and applications across various domains. This section reviews previous work related to OpenCV techniques for boundary detection and YOLO models for object detection and tracking.

A. OpenCV Techniques for Boundary Detection

OpenCV has been widely used for image processing tasks, particularly in edge and boundary detection. The library provides various tools and algorithms to facilitate these tasks.

- Canny Edge Detector: The Canny Edge Detector is a popular and widely used algorithm for edge detection in images. It uses a multi-stage process involving noise reduction, gradient calculation, non-maximum suppression, and hysteresis thresholding to detect a wide range of edges with high accuracy [3]. The Canny Edge Detector is known for its ability to minimize error rates and produce well-defined edges, making it suitable for applications like lane detection and object boundary identification.
- Morphological Transformations: Morphological transformations are a set of operations that process images based on shapes. They are particularly useful for removing noise, filling gaps, and emphasizing structures within an image. Operations like dilation, erosion, opening, and closing help refine edge detection results by enhancing or suppressing certain features [4]. In the context of boundary detection, morphological transformations can improve the accuracy of detected edges and contours.

- Structural Analysis and Shape Descriptors: Structural analysis and shape descriptors in OpenCV provide methods to analyze and interpret the shapes and structures within an image. Contour detection and approximation are fundamental techniques used to extract and represent the boundaries of objects [5]. By approximating contours to simpler shapes like polygons, circles, or ellipses, it becomes possible to identify and classify objects based on their geometric properties. This is particularly useful in detecting court boundaries by identifying rectangular shapes corresponding to the tennis court lines, especially applying for imperfect square detection on frames.

There are numerous examples of using OpenCV for line detection, including the Hough Transform method [6][7], for example detecting street lane on video feeds. The Hough Transform is effective for detecting lines and curves by transforming points in the image space to a parameter space and identifying lines through a voting process. This method has been applied in applications like road lane detection and object boundary identification, which parallels the requirements of this project.

B. YOLO Models for Object Detection and Tracking

YOLO (You Only Look Once) is one of the most successful and widely used models for real-time object detection and tracking. It formulates object detection as a regression problem and provides pretrained models across multiple versions, each improving upon the last in terms of speed and accuracy. YOLO has been effectively applied in numerous applications such as

- General Object Detection: YOLO has been employed for standard object detection tasks, identifying objects like cars, pedestrians, and everyday items with high accuracy [8].
- Sports Analytics: In the context of sports, YOLO has been used to detect and track tennis balls, demonstrating its potential in handling small objects when properly trained and configured [9]. Customized training on specific datasets can enhance YOLO's ability to detect objects that are not well-represented in its pretrained weights. Customized training on specific datasets can enhance YOLO's ability to detect objects that are not well-represented in its pretrained weights. However, YOLO may exhibit limitations in detecting and tracking small objects, necessitating the use of pre-trained models or fine-tuning to achieve better accuracy.

The adaptability of YOLO makes it a suitable choice for this project, where the goal is to detect tennis balls within a tennis court accurately.

III. Technical Approach

The project is structured into several key phases:

A. Setup and Installation

- Environment Configuration: Installed and configured OpenCV and YOLO on Jupyter Notebook.
- Model Selection: Performed initial tests with different versions of YOLO to select the most suitable one for detecting tennis balls.

B. Development of Detection Modules

- Object Detection with YOLO:
  - Used a pre-trained YOLOv8s model for initial tests.
  - Due to limitations in detecting tennis balls, trained a custom model using a tennis ball dataset [10] on Google Colab for 8 epochs.
  - Implemented the detection code to identify tennis balls in both live camera feeds and video inputs.
- Boundary Detection with OpenCV:
  - Applied Gaussian Blur to reduce noise in frames.
  - Used Canny Edge Detection to identify edges in the ROI.
  - Employed morphological operations (dilation and closing) to fill gaps in edges.
  - Detected contours and approximated them to identify square shapes representing the tennis court boundaries.
- Integration of Detection Methods:
  - Combined the ball detection and boundary detection modules.
  - Introduced a mechanism to track previous ball positions to improve detection consistency.
  - Adjusted overlay functions to accurately display detections without frame resets.

C. Testing and Refinement

- Edge Detection Optimization:
  - Adjusted parameters like kernel sizes and iteration counts to improve edge detection under varying environmental conditions.
  - Shifted from Hough Transform to contour detection due to the former's sensitivity to environmental factors.
- ROI Implementation: Applied a Region of Interest to focus processing on the tennis court area, reducing false positives from irrelevant regions.

- Overlay Adjustments:
    - Modified the overlay function to highlight court boundaries in red only when the ball hits the ground outside specified ranges.
    - Ensured that the detection visuals were accurate and informative.

D. Final Implementation

The final code integrates all modules and is capable of processing both live camera feeds and video files (.mp4). It accurately detects tennis balls and court boundaries, providing visual feedback on boundary violations.

IV. Experiments

The experiments conducted to develop and refine the tennis ball and court boundary detection system applicable in both live camera feeds and video inputs [3]. The experiments are divided into 2 main categories, including court line detection and ball detection. Each category includes multiple iterations and methods tested to optimize the system's performance.

A. Court Line Detection

The detection of court lines is critical for determining whether the tennis ball remains within the court boundaries. Several methods were explored to achieve robust and accurate line detection.

1) Canny Edge Detection and Hough Transform.

The initial approach involved using the Canny Edge Detector combined with the Hough Transform to detect straight lines representing the court boundaries

a) Implementation: Applied Gaussian Blur to the grayscale image to reduce noise, followed by the Canny Edge Detector to identify edges. The Hough Line Transform was then used to detect lines from the edges [4][7].

b) Results: The method detected some lines but was highly sensitive to environmental conditions such as lighting and shadows. The lines were often fragmented, and many false positives were detected due to textures and patterns in the background.

c) Conclusion: The sensitivity to noise and environmental factors made this method unreliable for consistent court line detection as Figure 1.

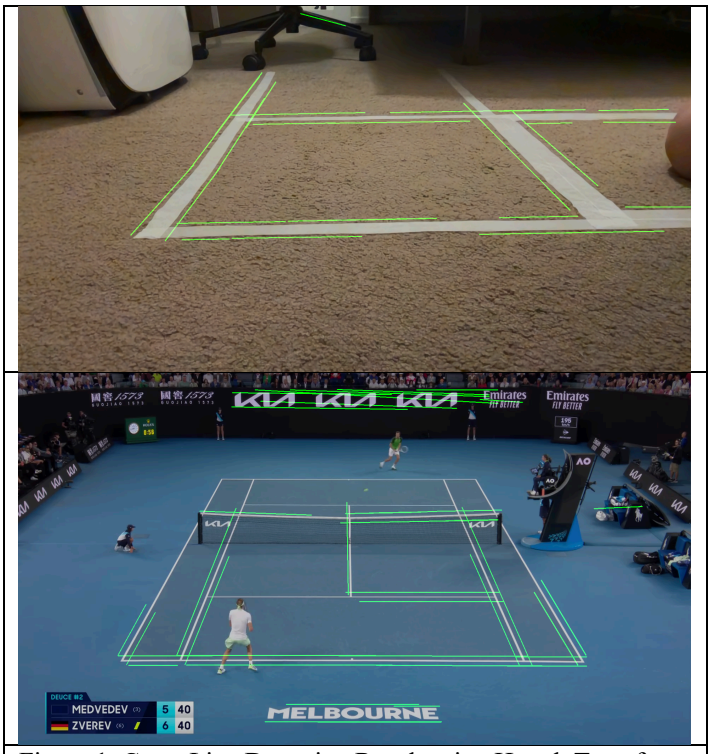


Figure 1. Court Line Detection Result using Hough Transform

2) Canny Edge Detection with Dilation.

To improve the results from the initial method, morphological operations were introduced

a) Implementation: After edge detection using the Canny method, dilation was applied using various kernel sizes and iteration counts to fill gaps in the detected edges [4].

b) Parameters Tested: Kernel sizes (3x3), (5x5), (7x7), (9x9) and Iterations: 1 to 5.

c) Results: Dilation helped to some extent in connecting fragmented edges, but the optimal parameters varied greatly depending on the frame and lighting conditions. Over-dilation led to merging of unrelated edges, while under-dilation failed to bridge gaps.

d) Conclusion: Canny Edge detection can give best fit edge result for training vdo feed and live camera environment setting where dilation improved edge continuity (iterations=2) and after GaussianBlur (kernel=(5,5)), it is sufficiently enhance line detection for reliable boundary identification.

3) Transition to Contours and Polygon Operations.

Recognizing the limitations of the previous methods (using Hough Transformation after Canny Edge Detection), a shift was made to use findContours() and approxPolyDP() to find all shapes and simplify each contour into a polygon for more extensively also add morphological closing after dilation to fill small holes and gaps in the edges.

a) Implementation: Applied morphological closing after dilation to fill small holes and gaps in the edges. Contour detection was then used to identify shapes resembling the court boundaries [4][6].

b) Challenges: The method struggled with detecting edges close to the image borders and was sensitive to noise, leading to inaccurate contour approximations.

c) Results: While some improvement was observed in detecting continuous lines enhancing by morphological, the method still failed to consistently identify the court boundaries accurately. However, using Contour detection helps filtering the shapes including square shapes, assuming having 4 vertices.

d) Conclusion: Morphological operations followed by Contour detection can enhance the program to find the correct square shape, but not only tennis court.

4) Canny Edge Detection with Morphological Operations and ROI.

To enhance detection accuracy, a Region of Interest (ROI) was introduced to focus on the area where the court is expected to be located.

a) Implementation:

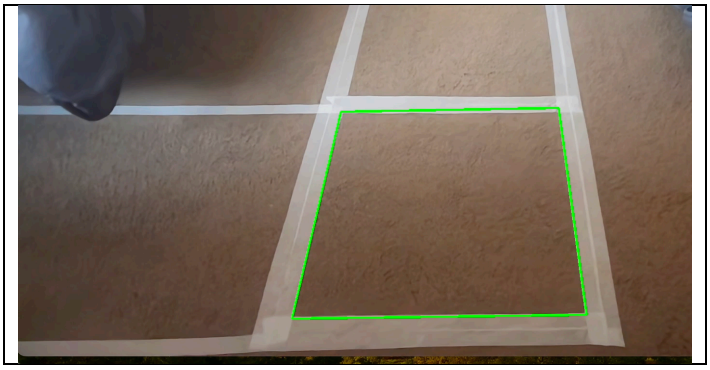
- *Applied ROI to the input frame to isolate the tennis court area for 4 sides.*
- *Performed Gaussian Blur and Canny Edge Detection within the ROI.*
- *Used dilation and closing morphological operations to improve edge continuity.*
- *Detected contours and approximated them to polygons, specifically looking for quadrilaterals representing the court [4][5][6].*

b) Parameters:

- *ROI coordinates were adjusted based on the video frame dimensions.*
- *Kernel size for morphological operations set to (5x5).*

c) Results: The introduction of ROI significantly reduced false positives from irrelevant areas. The method successfully detected the court boundaries in several frames, with improved consistency for court detection.

d) Conclusion: Combining Canny Edge Detection, morphological operations, and ROI provided the most reliable court line detection. This method and ste values are adopted for the final implementation as Figure 2.



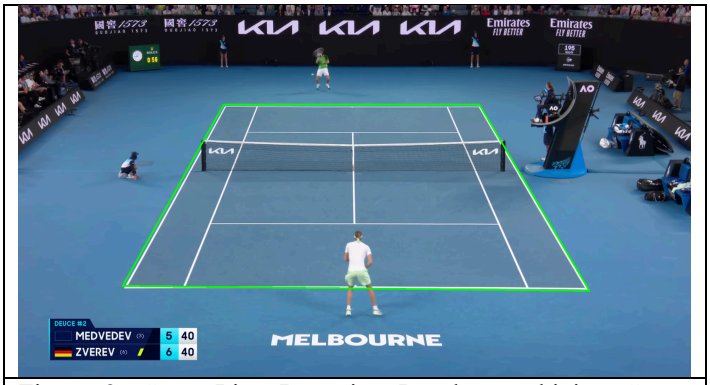


Figure 2. Court Line Detection Results combining Canny Edge Detection, Morphological operation with Contour Detection, and ROI.

B. Ball Detection

Accurate detection of the tennis ball is essential for tracking its movement and determining boundary interactions. Various methods and models were tested for optimal performance.

1) HSV Color Space for Tennis Ball Detection.

The initial approach for ball detection utilized color segmentation in the HSV color space.

a) Implementation: Converted frames to HSV color space and applied a color mask to isolate the yellow color of the tennis ball [10].

b) Results: This method worked reasonably well in controlled lighting conditions but was hardly detected when other closed-color objects appeared nearby in the frame.

c) Conclusion: HSV color segmentation alone was insufficient for reliable tennis ball detection, prompting the need for more advanced methods.

2) Comparison of YOLOv5 and YOLOv8 Models.

YOLOv5 and YOLOv8 were evaluated to be chosen for improving tennis ball detection accuracy rather than HSV alone.

a) Implementation:

- *Tested both YOLOv5 and YOLOv8 using their pre-trained weights.*
- *Evaluated performance on live camera feeds and example video footage.*

b) Results:

- *YOLOv5: Failed to detect the tennis ball consistently in both live camera feeds and video footage.*
- *YOLOv8: Demonstrated better performance in live camera detection but still struggled with video giving rarely ball detection.*

c) Conclusion: YOLOv8 showed superior performance over YOLOv5 for this application.

Due to its better detection capabilities in live feeds, YOLOv8 was selected for further development.

3) Training a Custom YOLOv8 Model

To address the limitations of the pre-trained models, a custom YOLOv8 model was trained specifically for tennis ball detection.

a) Implementation:

- Used a tennis ball dataset from Roboflow [11].
- Initially planned to train the model for 50 epochs, but due to time constraints (training time could exceeding 10 hours), then the reduced epochs are 8.
- Training was conducted on Google Colab to leverage GPU acceleration.

b) Results:

- The custom-trained model showed significantly improved detection of tennis balls in video feeds.
- Despite the limited number of epochs, the model was adequate for detecting and tracking the tennis ball.

c) Conclusion: Training a custom YOLOv8 model significantly enhanced the detection performance in video inputs. Time constraints limited the training duration, but the results were sufficient for the project's objectives.

C. Integration of Court Line Detection and Tennis Ball Detection

After developing the individual modules for court line and ball detection, they were integrated into a single system.

1) Challenges:
   a) Initial integration resulted in overlapping frames and inconsistent detections.
   b) The display of detection results from both modules interfered with each other due to frame resets.
2) Solutions:
   c) Introduced the prev_ball_positions variable to maintain continuity of ball positions across frames.
   d) Adjusted the overlay functions to properly combine the outputs without resetting frames.
3) Results: The integrated system successfully displayed both court boundaries and tennis ball detections simultaneously. The tracking and boundary validation mechanisms worked cohesively.
4) Conclusion: Resolving the integration issues allowed for a functional system that met the project's objectives.

D. Final Refinements

With the core functionality in place, final adjustments were made to optimize the system's performance within the project's time constraints.

1) Adjustments:
   a) Tweaked the ROI parameters for video inputs to achieve the best possible detection results after extensive testing.
   b) Refined the overlay function to improve the visual representation of boundary violations (red squares when the ball hits the ground) considering from comparing the ball movements between frames to going down.
2) Limitations:
   a) Detection accuracy was not perfect where sometimes the ball hit the ground, but the square shape is still green, and sometimes the tennis player hits the ball but square shape changes color to red; however, the system correctly identified boundary activities in approximately more than 50% of instances.
   b) Further improvements would require more time for parameter tuning and potentially more extensive training of the custom YOLOv8 model.
3) Conclusion: Given the project's scale and time constraints, the achieved results were satisfactory. The system demonstrated the capability to detect tennis balls and court boundaries effectively, fulfilling the primary objectives as Figure 3.

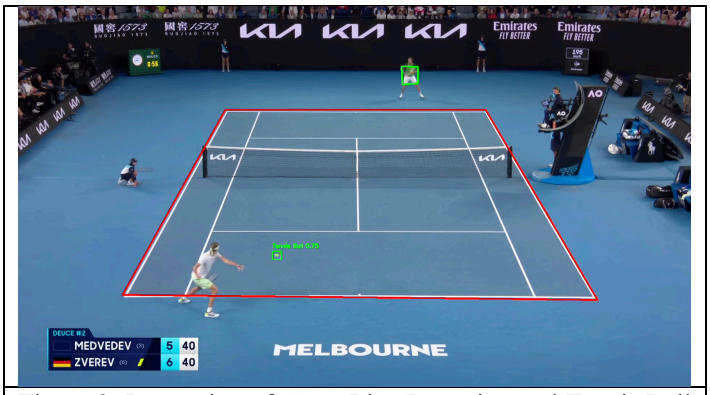


Figure 3. Integration of Court Line Detection and Tennis Ball Detection using YOLOv8 results on live cameras (iPhone and MacBook, respectively) and video input

V. Conclusion

This project successfully developed a system capable of detecting tennis balls and determining their positions relative to tennis court boundaries. By integrating YOLO for object detection with OpenCV's advanced image processing techniques, the system demonstrates significant potential for applications in tennis ball analysis and sports analytics.

Key challenges, such as environmental sensitivity and the detection of small objects, were addressed through custom model training and the implementation of focused Regions of Interest (ROI). Despite these efforts, the final model does not completely mitigate environmental sensitivity issues, making the program most effective in low-noise settings when using live camera feeds.

Future work may involve enhancing the speed of the detection component and improving the system's robustness under varying conditions. Additionally, expanding the model to better handle diverse environments and incorporating real-time processing optimizations could further increase its applicability and effectiveness in practical sports analytics scenarios.

Acknowledgment

A huge thanks to Oliver for helping me rescope this project and bringing back some ideas I had lost along the way. His support in setting up the frame to focus on the typical spots on a tennis court and defining the ROI made the detection process so much easier.

.

References

[1] Ultralytics, “Models supported by ultralytics,” https://docs.ultralytics.com/models/#featured-models, 2024, accessed: September 1, 2024.

[2] Soumyadip, “Mastering all yolo models from yolov1 to yolov9: Papers explained (2024),” https://learnopencv.com/mastering-all-yolo-models/, 2023, accessed: August 31, 2024.

[3] Australian Open TV, “Zverev Wins Incredible 51-Shot Rally Against Medvedev | Australian Open 2024,” https://www.youtube.com/watch?v=80_yDGIz_Rk, 2024, accessed: September 1, 2024.

[4] OpenCV Documentation, "Canny Edge Detector," https://docs.opencv.org/4.x/da/d5c/tutorial_canny_detector.html, 2024, accessed: September 15, 2024.

[5] OpenCV Documentation, "Morphological Transformations," https://docs.opencv.org/4.x/d9/d61/tutorial_py_morphological_ops.html, 2024, accessed: September 26, 2024.

[6] OpenCV Documentation, "Structural Analysis and Shape Descriptors," https://docs.opencv.org/4.x/d3/dc0/group__imgproc__shape.html, 2024, accessed: September 26, 2024.

[7] OpenCV Documentation, "Hough Line Transform," https://docs.opencv.org/3.4/d9/db0/tutorial_hough_lines.html, 2024, accessed: September 18, 2024.

[8] N. Anantha, "OpenCV Hough Transform Line Detection," YouTube, https://www.youtube.com/watch?v=KEYzUP7-kkU, 2020, accessed: September 18, 2024.

[9] Murtaza's Workshop, "Object Detection using YOLOv5 and OpenCV DNN in Python," https://www.youtube.com/watch?v=fu2tfOV9vbY, 2021, accessed: September 18, 2024.

[10] H. Saxena, "Real-Time Object Detection with YOLO, OpenCV, and Python," https://www.youtube.com/watch?v=L23oIHZE14w, 2021, accessed: September 27, 2024.

[11] V. Dhanwani, "Tennis Ball Detection Dataset," Roboflow Universe, https://universe.roboflow.com/viren-dhanwani/tennis-ball-detection/dataset/6, 2023, accessed: September 27, 2024.