Ball Tracking Robot using Image Processing Techniques

Omar Abdelwaly
Mechatronics Engineering Department
German University in Cairo
Cairo, Egypt
omar.abdelwaly@student.guc.edu.eg

Waleed Abdelrahman
Mechatronics Engineering Department
German University in Cairo
Cairo, Egypt
waleed.abdelrahman@student.guc.edu.eg

Abdallah Falah
Mechatronics Engineering Department
German University in Cairo
Cairo, Egypt
abdallah.falah@student.guc.edu.eg

Mohamed ElShall
Mechatronics Engineering Department
German University in Cairo
Cairo, Egypt
Mohammad.eshall@student.guc.edu.eg

Fouad Elsheikh
Mechatronics Engineering Department
German University in Cairo
Cairo, Egypt
fouad.elsheikh@student.guc.edu.eg

***Abstract*—This paper presents the development of an autonomous Arduino-powered car capable of detecting, tracking, and approaching a ball using image processing techniques. The system utilizes a camera module to identify the ball in real time, leveraging a combination of traditional computer vision methods and deep learning approaches for accurate detection under varying conditions. The car is equipped with an embedded control system that processes the ball's position and adjusts its movement accordingly. A PID controller ensures smooth navigation, allowing the car to follow the ball efficiently while minimizing sudden movements. Once the car reaches a predefined proximity to the ball, it stops to prevent collisions. The integration of efficient motor control, real-time image processing, and adaptive tracking algorithms enables reliable performance in dynamic environments. This work demonstrates a practical approach to autonomous ball tracking, with potential applications in robotics, sports analysis, and interactive robotic systems.**

I. Introduction

Ball tracking cars integrate image processing and robotics to autonomously follow a moving object, commonly a ball. This technology has applications in robotics competitions, autonomous vehicles, and sports analytics. Recent advancements in image processing techniques and embedded systems have improved the efficiency and accuracy of ball-tracking mechanisms.

In recent years, the development of adaptive real-time object detection systems has become crucial for enhancing the reliability and safety of autonomous driving systems (ADS). Traditional object detection algorithms, such as those based on convolutional neural networks (CNNs), often struggle to balance detection accuracy with computational efficiency, particularly under dynamic and resource-constrained environments. Many conventional approaches rely on high-power GPUs to process images and detect objects efficiently, making them impractical for edge devices and embedded systems used in autonomous vehicles. To address these challenges, Hemmati et al. [1] proposed an adaptive real-time object detection system that enhances pedestrian and vehicle detection across various environmental conditions, particularly different lighting scenarios. Their approach integrates a hardware-software co-design on Zynq UltraScale+ MPSoC, where hardware accelerators perform detection tasks while dynamically adjusting computational resources based on environmental factors.

The proposed system demonstrates significant improvements in both performance and power efficiency, making it a viable solution for real-time autonomous driving applications. Unlike previous works that primarily depend on fixed deep learning models for object detection, this system utilizes an adaptive framework that selects different detection methodologies depending on lighting conditions, ensuring reliable detection at all times. Furthermore, the integration of FPGA-based acceleration significantly reduces latency while maintaining high detection accuracy. The study highlights the importance of balancing hardware and software optimization to achieve real-time processing in embedded autonomous systems. These findings align with other research efforts focused on enhancing deep learning-based object detection using reconfigurable hardware and lightweight neural networks, demonstrating the growing trend of adaptive solutions in real-time vision applications for autonomous driving [1].

Building upon these advancements, further research has explored the integration of deep learning and computer vision to enhance the accuracy and efficiency of such systems. While hardware-software co-design approaches improve real-time processing and adaptability in constrained environments, recent studies highlight the need for optimizing deep learning models to achieve better generalization across different driving conditions. One such contribution comes from Pramudito [2], who investigates novel methodologies for improving object detection performance in autonomous systems through deep learning and advanced computer vision techniques.

Pramudito's research emphasizes the role of neural network optimization, feature extraction enhancement, and dataset augmentation in improving detection accuracy while maintaining computational efficiency. The study explores various architectures that balance real-time performance with minimal energy consumption, making them suitable for deployment in edge computing scenarios. Unlike traditional object detection approaches that rely solely on pre-trained

models, this work incorporates adaptive learning strategies, enabling the system to dynamically adjust to diverse environments, such as urban landscapes, highways, and low-light conditions. The findings contribute to the growing body of knowledge on deep learning-driven perception systems for autonomous driving, further reinforcing the importance of integrating adaptive methodologies into real-world applications.

Real-time object detection is a critical component in various domains, including autonomous systems and robotics. While prior research has primarily focused on pedestrian and vehicle detection for autonomous driving [1], [2], specialized applications such as ball detection in sports robotics require tailored approaches due to unique challenges such as motion blur, occlusions, and varying lighting conditions. Addressing these issues, Teimouri et al. [3] proposed a real-time ball detection approach leveraging Convolutional Neural Networks (CNNs) to enhance the accuracy and efficiency of ball recognition in dynamic environments.

The methodology introduced by Teimouri et al. incorporates a two-stage detection framework. Initially, an iterative algorithm identifies candidate regions within an image that are likely to contain a ball. These regions are then processed by a lightweight CNN, which refines the classification and localization of the ball with high precision. The proposed model achieves an impressive accuracy of 97.17 percent, making it a viable solution for real-time applications in robotic soccer and similar domains. Additionally, the study emphasizes the importance of optimizing computational resources, ensuring that the detection system runs efficiently on embedded platforms with limited processing power. By integrating machine learning-based object recognition with efficient region proposal methods, this research contributes significantly to advancing real-time vision systems in sports robotics and autonomous agents.

Expanding on the advancements in real-time object detection using deep learning, as demonstrated by Hemmati et al. [1] and Teimouri et al. [3], the broader field of object tracking has also seen significant progress through the application of deep learning techniques. In their comprehensive review, Uke et al. [4] analyze various deep learning-based object tracking methods, highlighting how these approaches have enhanced the accuracy and robustness of tracking systems in dynamic environments. The authors discuss the evolution from traditional tracking algorithms to modern deep learning frameworks, emphasizing the improvements in handling occlusions, varying lighting conditions, and complex backgrounds. This transition underscores the pivotal role of deep learning in advancing not only object detection but also continuous tracking, which is essential for applications such as autonomous driving, surveillance, and robotics.

Building upon the integration of deep learning in object detection and tracking, recent advancements have extended these techniques to visual servoing systems. Machkour et al. [5] provide a comprehensive survey of both classical and deep learning-based visual servoing methods, highlighting the evolution from traditional feature-based approaches to modern deep neural network applications. Their analysis emphasizes how deep learning has enhanced the adaptability and robustness of visual servoing systems in complex and dynamic environments, facilitating more precise and flexible robotic control. This progression underscores the transformative impact of deep learning across various facets of robotics, from perception to action.

The development of a ball-tracking car relies on efficient image processing techniques and robust hardware integration. Traditional image processing methods such as color segmentation and edge detection are effective under controlled conditions but struggle in dynamic environments. Deep learning approaches like YOLO and CNNs offer better adaptability but require powerful hardware. The use of embedded systems like Raspberry Pi and Arduino, coupled with PID controllers and DC motors, ensures smooth navigation and responsiveness. Future advancements may focus on optimizing computational efficiency and improving real-time adaptability in varied environments.

II. Methodology

A. Hardware Fabrication and Setup

The ball-tracking robot system was developed using a Raspberry Pi and an Arduino microcontroller. The Raspberry Pi was responsible for image acquisition and processing, while the Arduino handled motor control based on commands received from the Raspberry Pi.

Hardware Components:

- Raspberry Pi 4 (image acquisition and processing)
- Arduino Uno (motor control execution)
- USB Webcam
- L298N Motor Driver
- DC Motors
- Chassis with wheels
- Power Bank (mobile power supply)

Connections Overview:

- The USB Webcam is connected to the Raspberry Pi for real-time video input.
- Serial communication via USB is established between the Raspberry Pi and Arduino for command transmission.
- The Arduino generates PWM signals to control motor speed and direction via the L298N motor driver.

B. Image Processing Pipeline

The image processing pipeline was designed and implemented using OpenCV in Python, optimized for real-time performance on the Raspberry Pi platform.

1) Geometric Transformations:

- **Resizing:** Input frames are scaled to half their original size to reduce computation time.
- **Flipping:** Frames are horizontally flipped to match the camera's orientation with the physical motion of the robot.

2) Adaptive Contrast Enhancement: Instead of a static brightness and contrast adjustment, the Contrast Limited Adaptive Histogram Equalization (CLAHE) method was employed. CLAHE adaptively enhances local contrast in different regions of the image, making object detection more reliable under variable lighting conditions.

3) Smoothing: A Gaussian Blur is applied to the enhanced image to suppress noise and stabilize subsequent feature detection steps.

4) Morphological Operations: Morphological closing followed by opening is applied to the binary segmentation mask to eliminate small holes and remove isolated noise, improving the shape integrity of the detected object.

5) Contour Detection and Centroid Estimation: Contours are extracted from the edge map. The largest contour is assumed to correspond to the ball. Image moments are computed to determine the centroid, which is later used for robot motion control.

C. Closed-Loop Control Implementation

The robot's motion is controlled based on visual feedback using a closed-loop control system. Two main features are extracted:

- **Feature 1: Horizontal Offset** — The centroid of the detected ball is compared to the frame center. A proportional control approach adjusts the motor speed and direction to steer the robot.
- **Feature 2: Contour Area** — The size of the detected ball contour is used to control the forward movement. If the contour area is below a target threshold, the robot moves forward until the area increases, indicating the robot is close enough to stop.

Additionally, the average brightness of the frame is used to adjust the robot's forward speed dynamically to handle varying lighting conditions.

D. Ball Tracking and Motion Control Algorithm

Algorithm 1 Ball Tracking and Motion Control Algorithm

```
Initialize Serial Communication with Arduino
Initialize CLAHE for contrast enhancement
Initialize Camera and set resolution
while Robot is Active do
  Capture a frame from the camera
  Horizontally flip the frame
  Resize the frame to reduce computational load
  Apply CLAHE to equalize brightness (Y channel)
  Convert frame to HSV color space
  Create a binary mask for green color detection
  Apply mask to extract green object
  Convert segmented image to grayscale
  Apply binary threshold to the grayscale image
  Apply Morphological Opening and Closing to refine the mask
  Detect contours in the binary mask
  if Valid contours are found then
    Select the largest valid contour
    Compute the area of the contour
    Compute the centroid of the contour
    Calculate horizontal offset from frame center
    if Offset > Threshold then
      Compute proportional PWM based on offset
      Set direction LEFT or RIGHT based on offset sign
      Send movement command to Arduino
    else
      Send Stop command to Arduino
    end if
    if Contour Area is within target range & Offset < Threshold then
      Compute proportional PWM based on area
      Set direction FORWARD or STOP based on area
      Send movement command to Arduino
    else
      Send Stop command to Arduino
    end if
    if No valid contours then
      Send Stop command to Arduino
    end if
  else
    Send Stop command to Arduino
  end if
  Display FPS and processed frames
end while
Release Camera and close Serial Connection
```

E. Performance Comparison: Raspberry Pi vs Laptop

A performance comparison was conducted to evaluate the real-time efficiency of the system across two platforms: a Raspberry Pi and a standard laptop.

- The Raspberry Pi achieved a higher frame rate due to optimized camera integration and efficient pipeline design.
- Both platforms demonstrated comparable image processing performance, confirming the system's real-time suitability.

III. Results

IV. Conclusion

References

[1] M. Hemmati, M. Biglari-Abhari, and S. Niar, "Adaptive real-time object detection for autonomous driving systems," *Journal of Imaging*, vol. 8, no. 4, p. 106, 2022.

[2] D. K. Pramudito, "Enhancing real-time object detection in autonomous systems using deep learning and computer vision techniques," *The Journal of Academic Science*, vol. 1, no. 6, pp. 788–804, 2024.

[3] M. Teimouri, M. H. Delavaran, and M. Rezaei, "A real-time ball detection approach using convolutional neural networks," in *RoboCup 2019: Robot World Cup XXIII*, pp. 323–336, 2019.

[4] N. Uke, P. Futane, N. Deshpande, and S. Uke, "A review on deep learning-based object tracking methods," *Multiagent and Grid Systems*, vol. 20, no. 1, pp. 1–20, 2023.

[5] Z. Machkour, D. Ortiz-Arroyo, and P. Durdevic, "Classical and deep learning based visual servoing systems: A survey on state of the art," *Journal of Intelligent & Robotic Systems*, vol. 102, no. 1, pp. 1–20, 2022.