Annasaheb Dange College of Engineering and Technology, Ashta

(An Autonomous Institute affiliated to Shivaji University, Kolhapur)
(NAAC A++ Grade Accredited Institute, NBA Accredited Program, ISO 9001: 2015 Certified)

Report on Shuttlecock Detection and Real-Time Parking Space Counter Projects

Department Of Computer Science & Engineering

Under The Guidance Of

Mr. Sabarishwaran R

ISE ACTIVITY

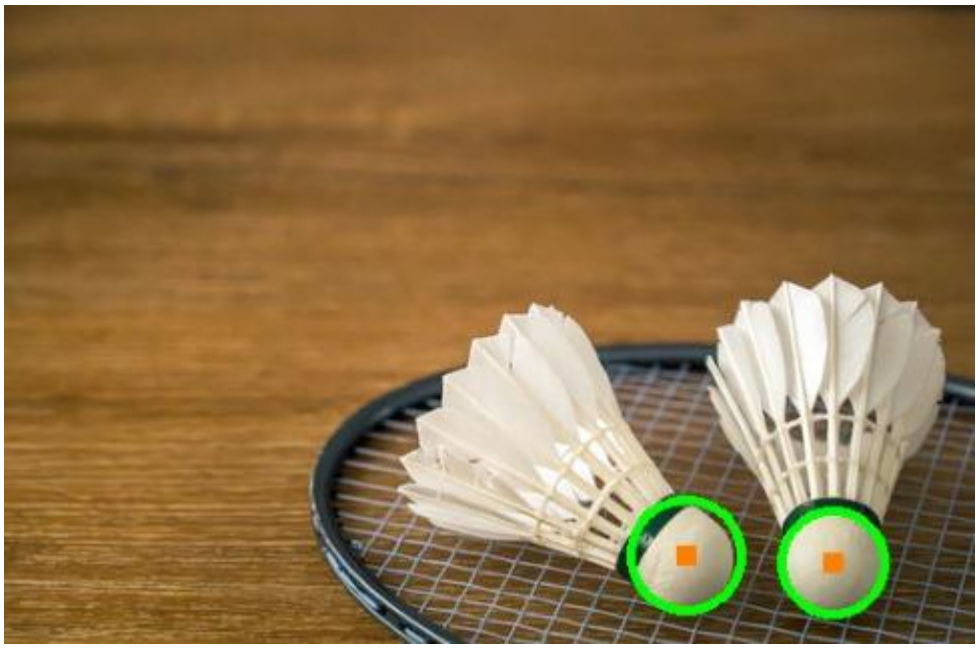
TY Tech Sem -V

Group Members Name

Sr no.	Student Name	Roll No	URN No	Sign
1	Prathamesh Patil	3139	1022031066	
2	Sarthak N Patil	3061	1022031105	

Signature

Shuttlecock Detection using Hough Circle Transform



Abstract

The "Shuttlecock Detection using Hough Circle Transform" project utilizes OpenCV, a powerful Python-based library, to detect shuttlecocks in images. The approach begins with preprocessing, such as converting images to grayscale and applying Gaussian Blur to minimize noise. The Hough Circle Transform is then employed to identify the shuttlecock's circular shape. The system is highly customizable, enabling users to adjust parameters like radius range, sensitivity, and the minimum distance between detected circles. This solution is efficient, cost-effective, and suitable for sports analytics, automated training, and badminton object detection. Additionally, it serves as the foundation for future real-time tracking systems.

Introduction

Shuttlecock detection plays a crucial role in sports analysis and computer vision, particularly in badminton. Tracking shuttlecock movement provides valuable insights into player performance and gameplay dynamics. This project integrates computer vision techniques, including grayscale conversion, Gaussian Blur, and Hough Circle Transform, to detect shuttlecocks in images. The ultimate goal is to develop a tool for real-time applications such as automated training systems and game analysis.

Objectives

The primary objectives of this project are:

1. **Accurate Detection**: To reliably identify shuttlecocks' circular shapes in images through edge and gradient analysis.
2. **Noise Reduction**: By applying grayscale conversion and Gaussian Blur, the detection accuracy is improved.
3. **Customizable Parameters**: To allow users to adjust detection parameters like radius, sensitivity, and distance for a variety of scenarios.
4. **Practical Application**: The tool aims to be useful for sports analytics, automated training, and object detection in badminton.
5. **Foundation for Real-time Systems**: Laying the groundwork for future real-time shuttlecock tracking in videos.

Step-by-Step Explanation

1. **Import Libraries**:
   - **cv2**: For image processing functions.
   - **numpy**: For numerical operations.
   - **matplotlib.pyplot**: For displaying images.

2. **Define Function**:
   A function detect_shuttlecock() is created to process the input image and detect shuttlecocks.
3. **Preprocessing**:
   The image is first converted to grayscale, followed by Gaussian Blur to smooth out noise, making the detection process more reliable.
4. **Hough Circle Transform**:
   The cv2.HoughCircles() function is applied to detect circular shapes. Various parameters such as minimum and maximum radius, sensitivity thresholds, and minimum distance between circles are used to fine-tune the detection process.
5. **Circle Drawing**:
   Detected shuttlecocks are highlighted with circles on the image, with their centers marked for easy identification.
6. **Display Result**:
   The processed image, showing detected shuttlecocks, is displayed using Matplotlib.
7. **Handling No Detection**:
   If no shuttlecock is detected, the program prints a message informing the user.

Code Overview

```
import cv2
import numpy as np
import matplotlib.pyplot as plt

def detect_shuttlecock(img):
    img = cv2.imread(img)
    gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
    blurred = cv2.GaussianBlur(gray, (15, 15), 0)
    circles = cv2.HoughCircles(blurred, cv2.HOUGH_GRADIENT, dp=1, minDist=50,
param1=50, param2=28, minRadius=10, maxRadius=50)
    if circles is not None:
        circles = np.round(circles[0, :]).astype("int")
        for (x, y, r) in circles:
            cv2.circle(img, (x, y), r, (0, 255, 0), 4)
            cv2.rectangle(img, (x - 5, y - 5), (x + 5, y + 5), (0, 128, 255), -1)
        plt.imshow(cv2.cvtColor(img, cv2.COLOR_BGR2RGB))
        plt.axis('off')
        plt.show()
    else:
        print("No shuttlecock detected in the image.")
```

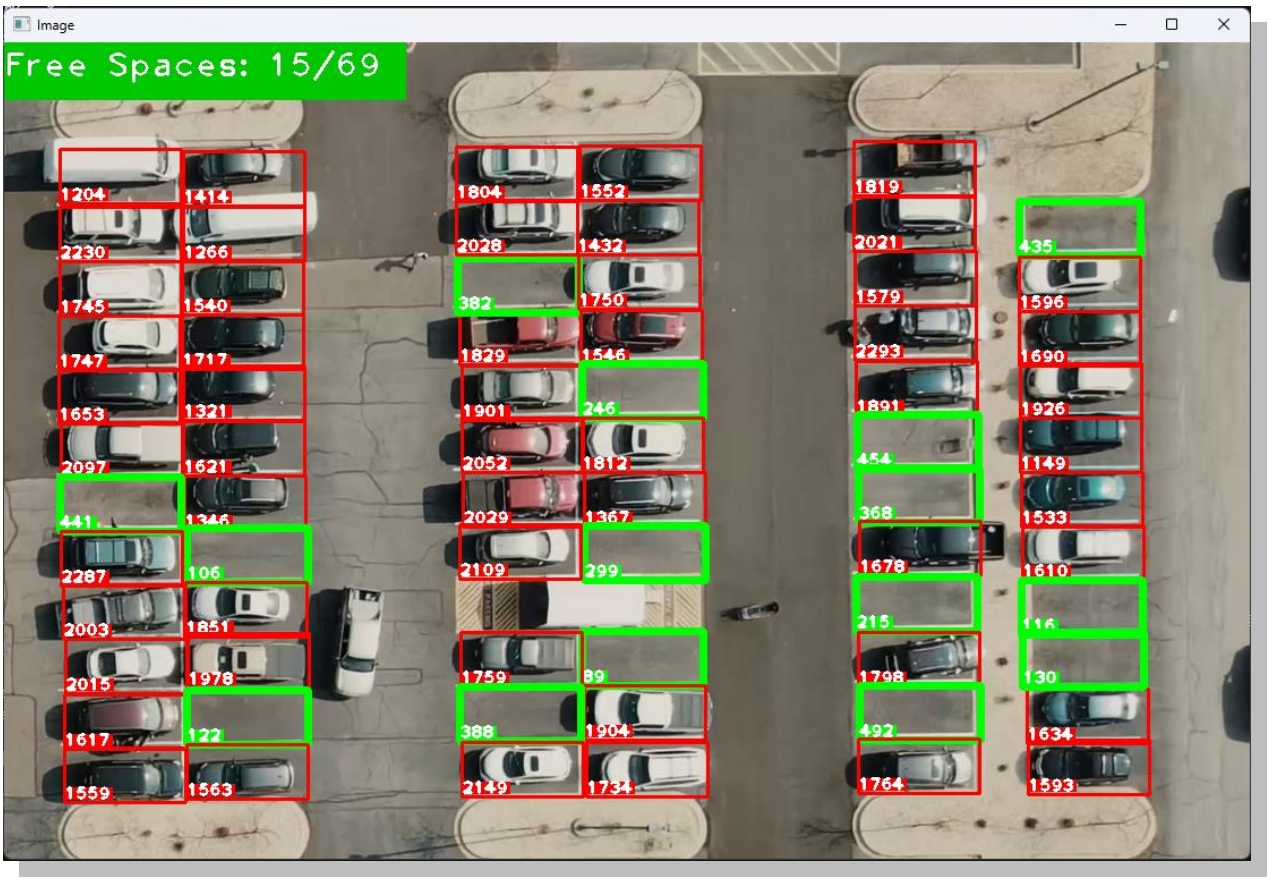
Conclusion

1. **Summary**:
   This project demonstrates the successful detection of shuttlecocks in images using the Hough Circle Transform. Detected shuttlecocks are highlighted, providing a useful tool for sports applications such as automated training and game analysis.
2. **Future Work**:
   - **Real-Time Detection**: Extend the tool to handle live video streams for real-time tracking of shuttlecocks.
   - **Enhanced Accuracy**: Incorporate machine learning models to improve detection under complex backgrounds or varying lighting.
   - **Multiple Object Detection**: Enhance the system to detect multiple shuttlecocks simultaneously.
   - **Motion Analysis**: Integrate motion tracking for analyzing shuttlecock trajectory and speed.
   - **Cross-Platform Support**: Optimize the system for mobile and embedded devices.

Real-Time Parking Space Counter Using OpenCV

Project Introduction

The "Real-Time Parking Space Counter Using OpenCV" project aims to create a Python-based solution that detects and counts vacant parking spaces in real time from a video feed. The system processes video frames to detect parking space occupancy and provides real-time updates to assist drivers in locating available spaces. The solution employs image processing techniques such as grayscale conversion, Gaussian Blurring, and thresholding, along with a dynamic system that continuously tracks parking space occupancy.



Objectives

The key objectives of this project are:

1. **Parking Space Detection**: Detect vacant and occupied parking spaces in real time from a video feed.
2. **Efficient Image Processing**: Enhance video frames for clarity and reduced noise, enabling more accurate parking space analysis.
3. **Real-Time Feedback**: Dynamically update parking space availability and display it on the video feed.
4. **Interactive Spot Definition**: Allow manual definition of parking spaces via a mouse tool, providing a customized layout for parking lots.

Workflow

1. **Spot Definition** (ParkingSpacePicker.py):
   A static parking lot image is used to manually define parking spot positions using mouse clicks. These positions are saved and later used in video feed processing.
2. **Video Feed Processing** (main.py):
   The system processes the video feed frame by frame, converting each frame to grayscale, followed by blurring and thresholding to enhance the image quality for better analysis.
3. **Parking Space Analysis**:
   The system analyzes each parking spot's pixel density to determine its occupancy. If a spot is vacant, it is highlighted in green; if occupied, it is marked red.
4. **Result Display**:
   The parking spaces' status (vacant or occupied) is dynamically updated on the video feed, making it easier for drivers to locate available spots.

Code Overview

```
import cv2 as cv
import pickle
import cvzone
import numpy as np

cap = cv.VideoCapture("carPark.mp4")

with open('CarParkPos', 'rb') as f:
    posList = pickle.load(f)

width, height = 107, 48
```

```
def checkParkingSpace(imgProcessed):
    spaceCounter = 0
    for pos in posList:
        x, y = pos
        imgCrop = imgProcessed[y:y + height, x:x + width]
        count = cv.countNonZero(imgCrop)
        if count < 950:
            color = (0, 255, 0)
            thickness = 5
            spaceCounter += 1
        else:
            color = (0, 0, 255)
            thickness = 2

        cv.rectangle(img, pos, (pos[0] + width, pos[1] + height), color,
thickness)
        cvzone.putTextRect(img, str(count), (x, y + height - 3), scale=1,
thickness=2, offset=0, colorR=color)

    cvzone.putTextRect(img, f'Free Spaces: {spaceCounter}/{len(posList)}', (0,
30), scale=2, thickness=2, offset=20, colorR=(0, 200, 0))

while True:
    success, img = cap.read()
    if not success:
        break

    imgGray = cv.cvtColor(img, cv.COLOR_BGR2GRAY)
    imgBlur = cv.GaussianBlur(imgGray, (3, 3), 1)
    imgThreshold = cv.adaptiveThreshold(imgBlur, 255,
cv.ADAPTIVE_THRESH_GAUSSIAN_C, cv.THRESH_BINARY_INV, 25, 16)
    imgMedian = cv.medianBlur(imgThreshold, 5)
    kernal = np.ones((3, 3), np.uint8)
    imgDilate = cv.dilate(imgMedian, kernal, iterations=1)

    checkParkingSpace(imgDilate)
    cv.imshow("Image", img)
    if cv.waitKey(10) & 0xFF == ord('q'):
        break

cap.release()
cv.destroyAllWindows()
```

Conclusion

1. **Summary**:
   The Real-Time Parking Space Counter successfully detects vacant and occupied parking spots by analyzing video frames. The system dynamically updates the parking availability status on the video feed, helping drivers locate free spots efficiently. This project showcases the potential applications of computer vision in smart parking management systems.
2. **Future Work**:
   - **AI-based Spot Detection**: Implement machine learning models to improve detection accuracy.
   - **Live Streaming Integration**: Adapt the system for use with live CCTV camera feeds for real-time parking lot monitoring.

- **User Alerts**: Add notifications to alert users when parking spots become available.

Appendix

Installation Instructions

To set up the environment and dependencies for both projects, follow the steps below:

1. **Install Required Libraries**: Make sure you have Python 3.x installed, and then use the following command to install the required packages:

```
pip install opencv-python
pip install numpy
pip install matplotlib
pip install cvzone
```

2. **Download the Project Files**:
    - For the **Shuttlecock Detection** project, download the image you wish to process and place it in the same directory as the Python script.
    - For the **Real-Time Parking Space Counter**, ensure you have a parking lot video feed (e.g., carPark.mp4) and a defined list of parking spot positions saved in a .pkl file.
3. **Running the Projects**:
    - For Shuttlecock Detection, simply run the Python script after setting the correct image path.
    - For the Real-Time Parking Space Counter, ensure your video file (carPark.mp4) and saved parking spot positions (CarParkPos.pkl) are in place before executing the script.

```
#for shuttle cock detection
python shuttlecock_detection.py
#for parking space counter
python main.py
```

Sample Outputs

- **Shuttlecock Detection**: When the shuttlecock is detected, the program will highlight the circular shape with green circles and small rectangles at the center, with the image displayed for the user.
- **Parking Space Counter**: The video will show the parking spaces overlaid with green for vacant spots and red for occupied spots. The real-time count of free spaces is displayed on the top-left corner of the video feed.

References

1. OpenCV Documentation - https://docs.opencv.org/
2. NumPy Documentation - https://numpy.org/doc/stable/
3. Matplotlib Documentation - https://matplotlib.org/stable/contents.html
4. cvzone Library - https://github.com/cvzone/cvzone

Team Information

Performed By:

1. **Prathmesh Patil**

- **College:** Annasaheb Dange College of Engineering and Technology Ashta
- **URN:** 1022031066
- **Department:** Computer Science and Engineering
- **LinkedIn ID:** https://www.linkedin.com/in/prathmesh-patil-03407427b/
- **Email:** pprathmesh7777@gmail.com

2. **Sarthak Patil**

- **College:** Annasaheb Dange College of Engineering and Technology Ashta
- **URN:** 1022031105
- **Department:** Computer Science and Engineering
- **LinkedIn ID:** https://www.linkedin.com/in/sarthak-patil-882475296
- **Email:** patilsarthak336@gmail.com