

# CSU44053 Computer Vision

## Assignment 1 Table Tennis

### Minjuan Luo

### 20313326

Part I: Locate the table tennis ball.

Using color, regions and shape locate table tennis balls in the supplied images

Purpose of the Code:

The code is intended to process a series of images (from "Ball1.jpg" to "Ball10.jpg") representing various tables(backgrounds) with ping pong balls. It applies image processing techniques to detect the balls and draw circles around them to indicate their position and size.

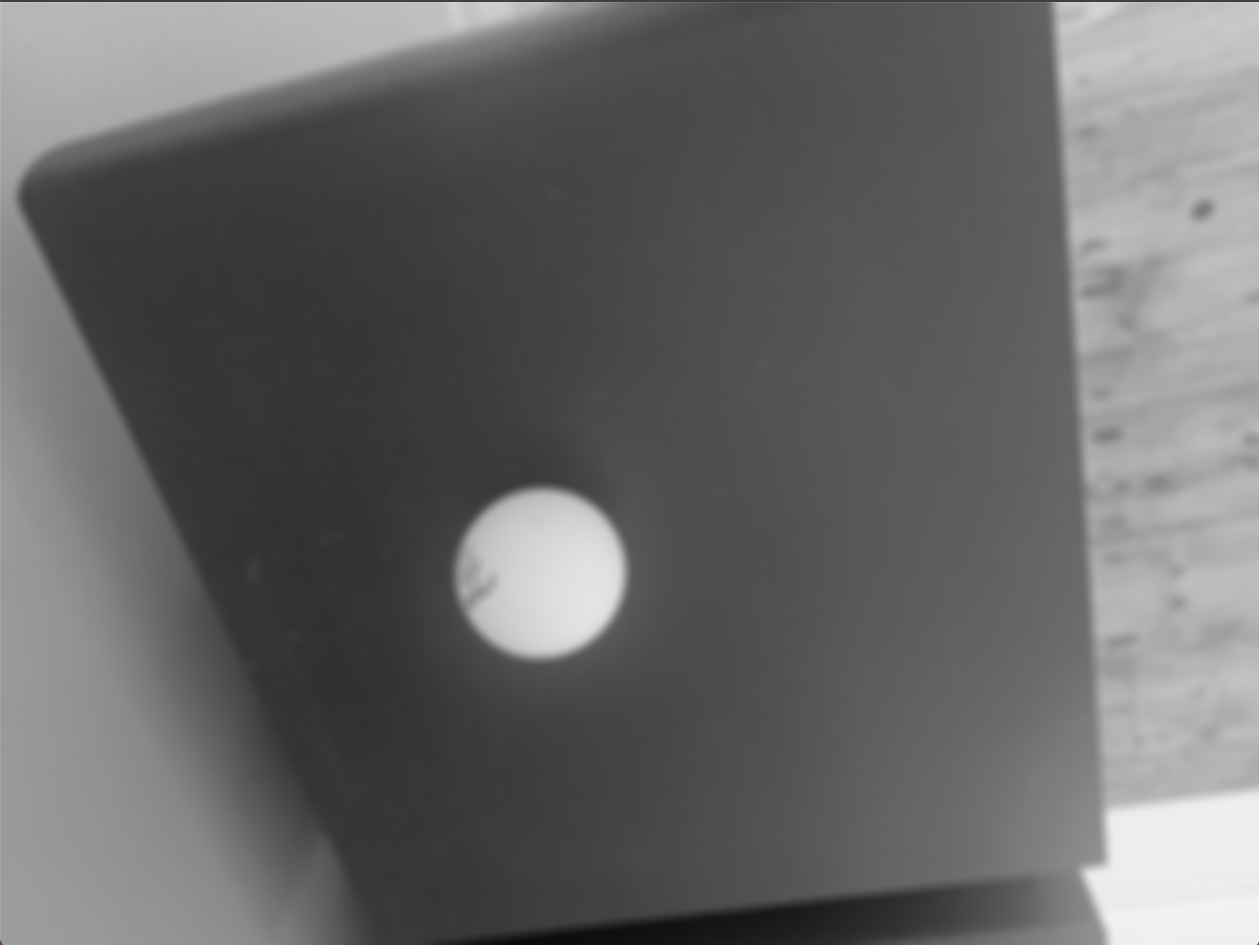
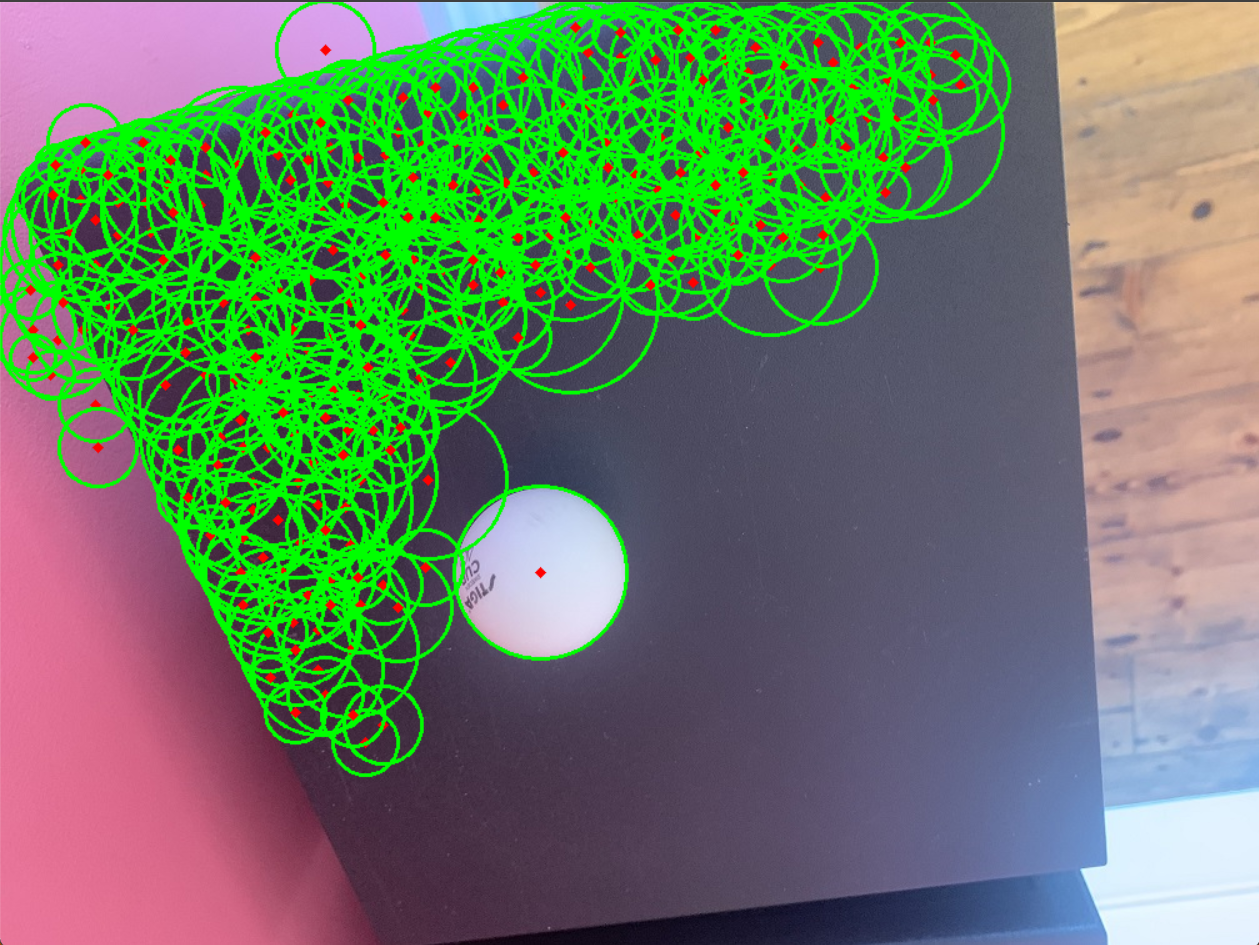
Functions and Purposes:

1. ConvertToGray Function:

Purpose: Converts the input image to grayscale and applies Gaussian blur.

cvtColor: Converts the image from BGR to grayscale.

GaussianBlur: Applies blurring to the grayscale image to reduce noise and improve circle detection (if not using this method, numerous detected circles would occur as error detection).

1. DetectCircles Function:

Purpose: Utilizes the Hough Circle Transform to detect circles in the processed image.

HoughCircles: Detects circles using the Hough Circle Transform technique, providing their centers and radii.

Fine-tuned parameters:

HOUGH\_GRADIENT is a variant of Hough transform specifically designed for circular shape detection.

1 - The inverse ratio of the accumulator resolution to the image resolution. This defines the size of the accumulator relative to the input image.

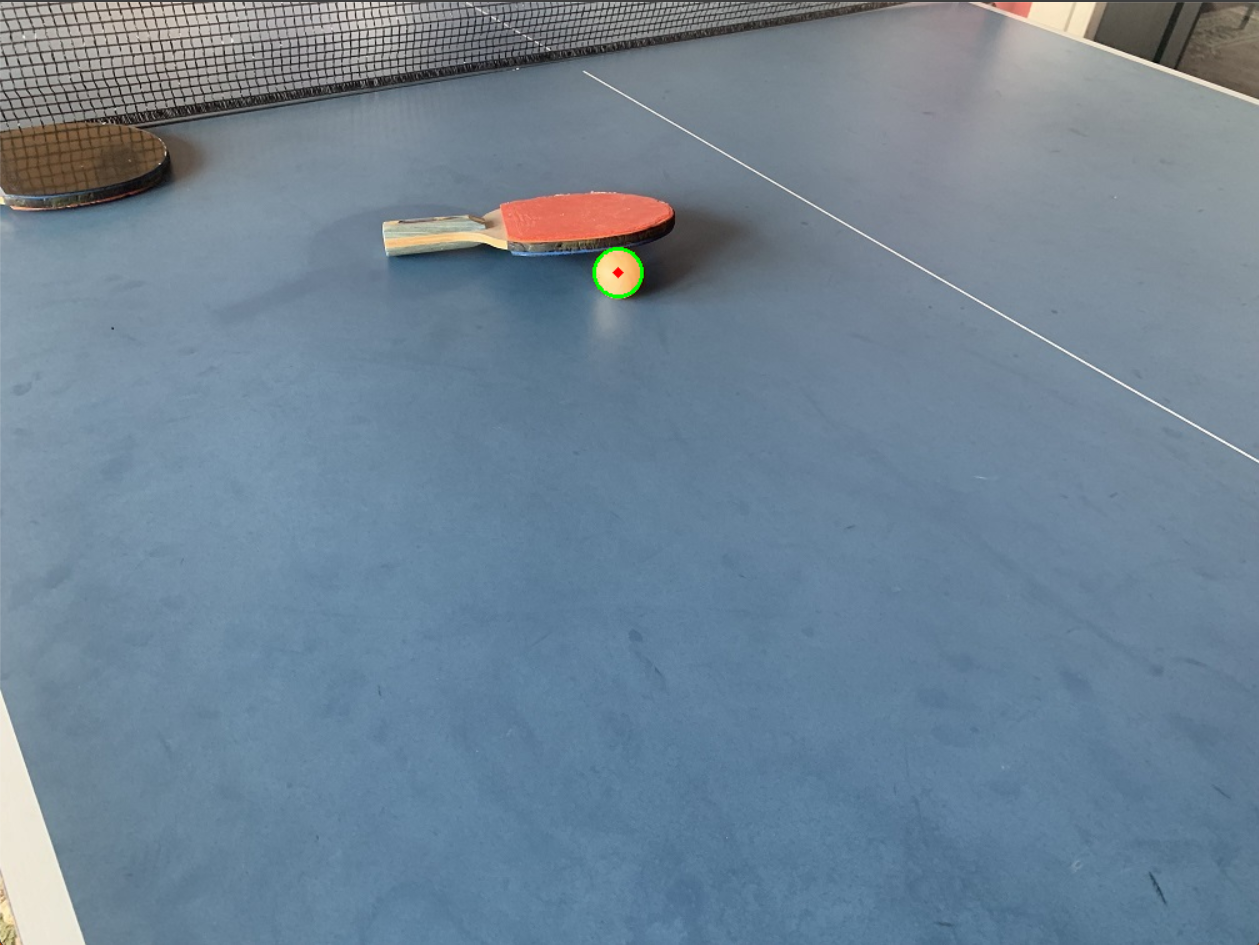
20 - Minimum distance between the centers of the detected circles. It specifies the minimum distance in pixels between the centers of the detected circles.

50 - The higher threshold for Canny edge detector. It is used in the Canny edge detection step.

30 - Threshold for circle detection. A smaller value allows more circles to be detected, potentially with more false positives.

15 - Minimum circle radius. It specifies the minimum radius of the circles to be detected.

80 - Maximum circle radius. It specifies the maximum radius of the circles to be detected.

1. Main Function:

Image Processing and Circle Detection:

Reads the image files in a loop from "Ball1.jpg" to "Ball10.jpg".

If an image fails to load, it displays an error message and continues to the next image.

If circles are detected, it marks them with green and red circles and displays their positions and diameters.

1. Image Processing Workflow:

Preprocessing:

Load image → Convert to grayscale → Apply Gaussian blur → Circle Detection:

Main idea:

Using Hough Circle Transform on the preprocessed image to detect circles.

Visualization:

Drawing detected circles on the original image.

Displaying the image with marked circles.

1. Conclusion:

The code demonstrates Hough Circle Detection along with grayscale and gaussian blur for ping pong ball detection using OpenCV. It applies image processing techniques to identify circular shapes within the images and visualize the detected circles.

The Hough Circle Detection method utilized in the code is suitable for identifying static images of Ping-Pong balls on a table. However, its effectiveness diminishes when dealing with a table tennis video in part 3 of the assignment.

In this scenario, the ball's shape changes in each frame, making it inconsistent and deviating from a perfect circular shape. As a result, a new detection method tailored for video frames will be introduced in part 3 to accurately handle the dynamic and non-circular ball shapes.

Part II: Locate the table tennis table.

You must now locate the corners of the table (the outside of the white lines) using edge detection, and then transform the image so that you have a plan view of the table.

The detection process is separated into two situations: the table with heavy light reflection and the table without light reflection

For Heavy Light Reflection:

1. Image Loading and Conversion:

The code loads Table2.jpg image of the table tennis table which is considered as heavy light reflection.

The image is converted from the BGR color space to the HSV color space, facilitating more effective color-based segmentation and processing including S channel and V channel altering.

1. Histogram Equalization for Contrast Enhancement:

The Value (V) channel in the HSV image undergoes histogram equalization. This technique aims to improve the contrast of the image by spreading out the intensity levels, compensating for uneven lighting and reducing the impact of light reflections.

1. Saturation Adjustment to Emphasize Color Intensity:

The saturation (S) channel in the HSV image is adjusted through multiplication. This process aims to emphasize the intensity of colors, compensating for potential color information loss due to extreme lighting conditions.

1. Contrast Limited Adaptive Histogram Equalization (CLAHE):

A CLAHE process is applied to the Value (V) channel multiple times. CLAHE locally enhances the contrast, preventing the saturation of pixels and addressing over-amplification in brighter regions, which often occurs in typical histogram equalization.

Several parameters can be fine-tuned to adjust the image enhancement. These parameters include:

4.1 Clip Limit (clahe->setClipLimit()):

Role: It determines the contrast limit for the histogram equalization. Higher values increase the contrast enhancement effect but might result in over-amplification, while lower values might limit the impact of the enhancement.

4.2 Grid Size (clahe->setTilesGridSize()):

Role: Specifies the grid size for dividing the image into smaller blocks for localized contrast enhancement. Larger grid sizes can affect the overall impact of enhancement within specific regions.

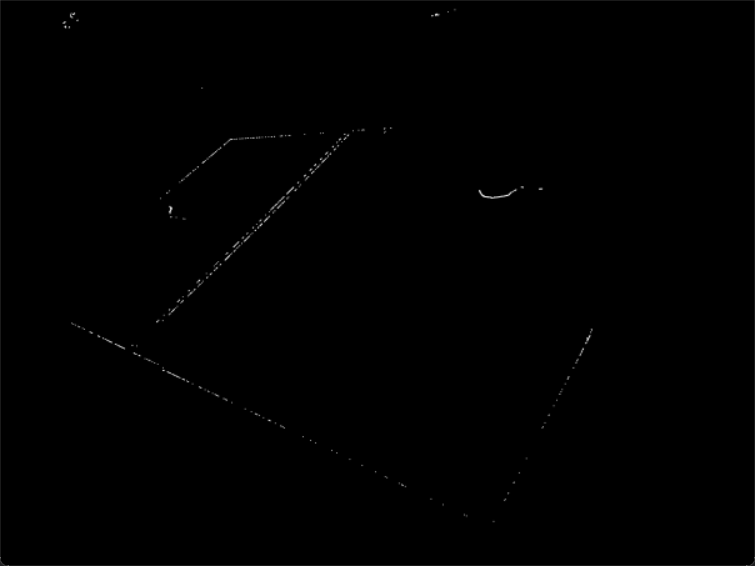
4.3 Number of Iterations:

Role: The number of iterations the CLAHE is applied to the image.

Impact: Increasing the number of iterations affects the extent of contrast improvement and helps adapt the enhancement level across the image.

1. Blue Color Range Thresholding:

Defines the range of blue color in the HSV color space, creating a mask that isolates the blue color present in the image. This step is crucial in identifying the table surface, typically characterized by a blue color.

1. Morphological Operations:

A dilation operation is performed on the blue mask. Dilation aims to expand the blue regions in the mask, potentially emphasizing and enhancing the boundaries of the table in the image.

1. Edge Detection (Canny Edge Detection):

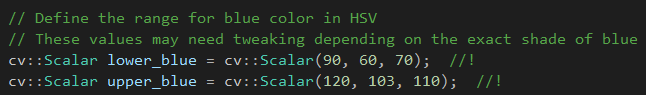
The Canny edge detection algorithm is utilized on the processed blue mask. This technique is employed to detect edges in the image, potentially aiding in delineating the table's boundaries and contours.

1. Hough Line Detection:

The Hough Transform is used for line detection in the blue-masked image. This step attempts to identify straight lines that might correspond to the edges or prominent features of the table like corners, and then contributing to its localization by simply expanding the detected result to find intersection.

1. Visualization and Display:

The code displays the processed image after multiple iterations of CLAHE and saturation adjustment, allowing observation of the enhanced image. It also exhibits the blue-filtered image, visually highlighting the detected regions with blue color—the presumed table area.



1. Potential Enhancements:

To further optimize this process, fine-tuning parameters such as CLAHE's clip limit, grid size, and saturation multiplier can be crucial for detecting the table under varying light conditions.

1. Conclusion:

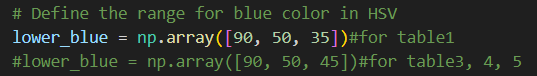
This special code for light reflection illustrates a comprehensive strategy by combining multiple image enhancement techniques and color-based segmentation including expanding result in line detection, it aims to isolate and identify the table within the image.

For No Light Reflection:

1. Isolating Blue Color (Table Surface):

The code defines a specific blue color range in the HSV color space, creating a mask that isolates the blue color associated with the table surface.

Base on the different status of the rest 4 table pictures, the blue filter’s lower blue boundary is in need of altering the parameter value:

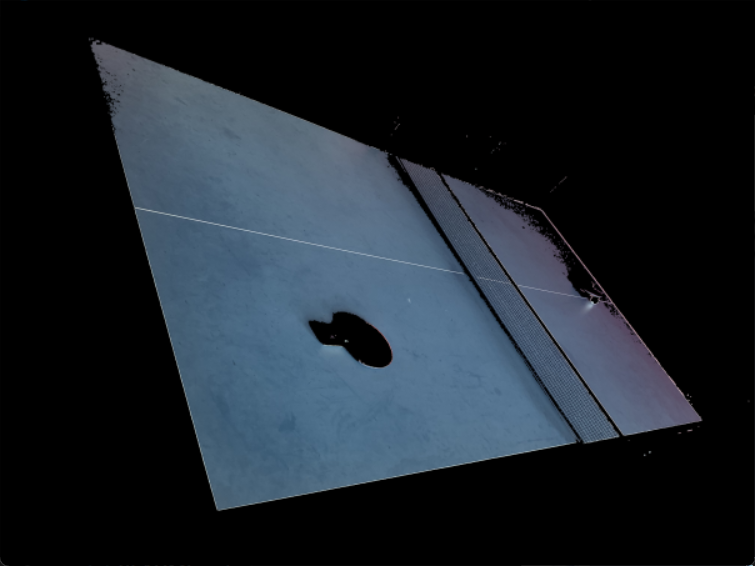
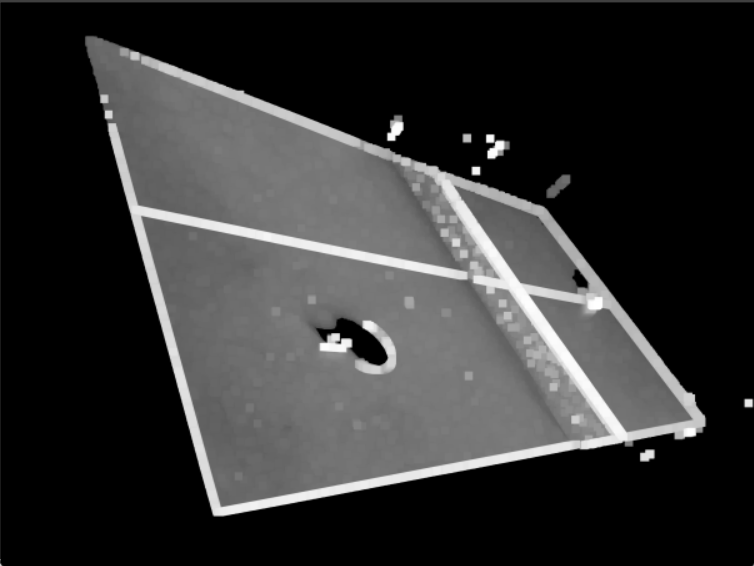
* 1. lower\_blue = np.array ([90, 50, 35]) and satuation\_multiplier = 2 for table1
  2. lower\_blue = np.array ([90, 50, 45]) and satuation\_multiplier = 2 for table3, table4, table5
  3. 

1. Enhancement and Isolation:

Dilation emphasizes blue regions in the mask, aiming to better highlight the potential table area.

Grayscale Conversion and Contour Detection:

The blue-filtered image is converted to grayscale for contour identification, facilitating accurate contour detection and reduced computational load.

1. Identification of Table Area:

The code detects contours and aims to identify the largest contour resembling a rectangle, potentially indicating the table's region.

Basically using ＂find\_largest\_rectangle\_contour(contours)＂ and then draw contours base on the result＂cv2.drawContours(blank\_image, [largest\_rectangle\_contour], -1, (0, 255, 0), 3)＂

1. Corner Detection and Area Calculation:

Utilizing the identified contours, the code performs corner detection to find potential table corners.

It analyzes combinations of these corners to identify the four corners in the largest area(table) by a loop structure, and then enable the possibility to outlining the table's boundaries

Potential Enhancements:

Parameter Fine-Tuning: Adjusting color range, dilation iterations, or corner detection thresholds could enhance the accuracy of table detection.