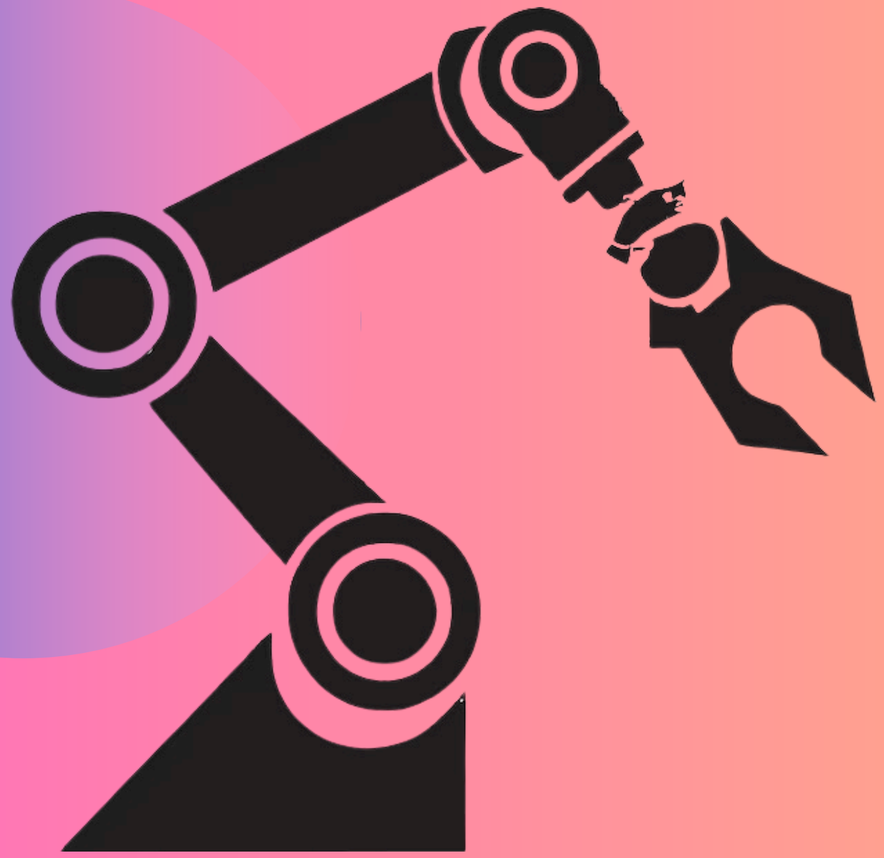


Autonomous Chinese Checkers Playing Robot Arm

FYP24057

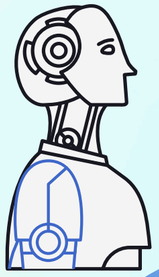
Leung Ho Ning 3035801453
Shiu Chun Nam Alex 3035800849

01

Objecties and Background

Background

- Rise of AI and Robotics in Chess Games



- Increased Interest in Robotics

- Cultural Significance

Motivation

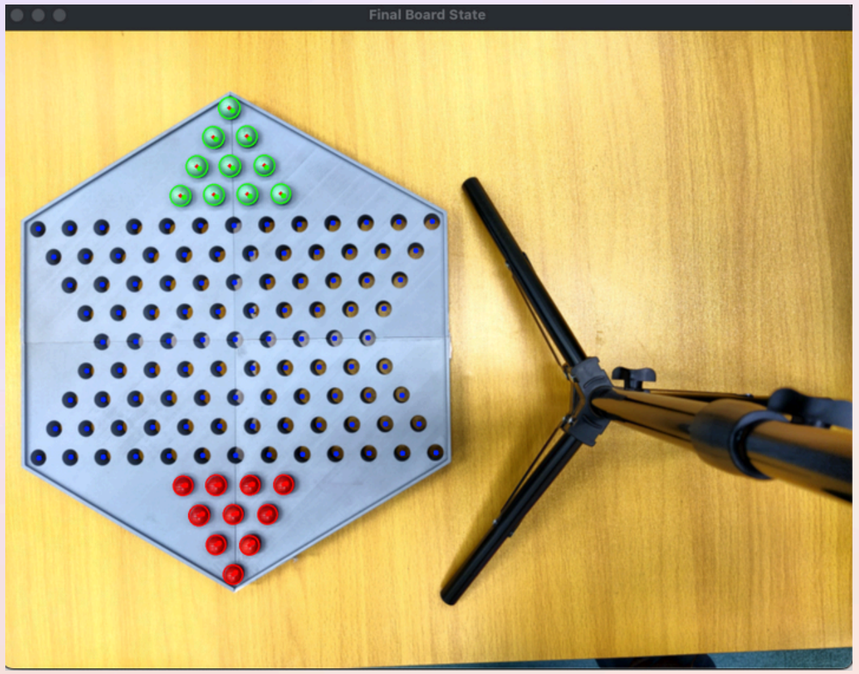
- Nostalgia Meets Innovation
- Bridging the Gap For Chinese Checker compared to other Chess Games

Objectives

- Innovative Integration
- Enhancing the Gaming Experience

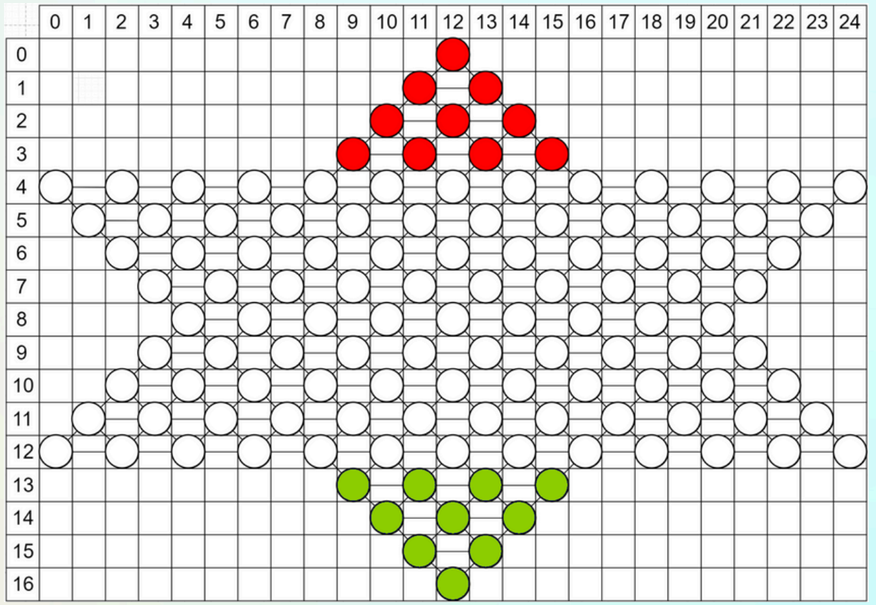
Key Deliverables

- **Checker Board Recognition**



- **Remote Control of Robotic Arm**

- **Artificial Chinese Checker**

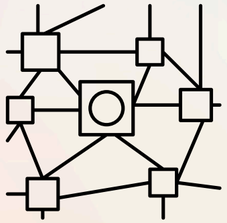


Uniqueness

- No existing solutions focus on Chinese Checker detections

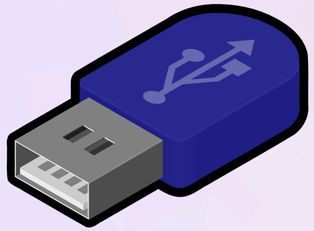
- First Automated Player:
  No product exists that can autonomously play Chinese Checkers

- We focus on building a practical system to provide a real-world interactive product.



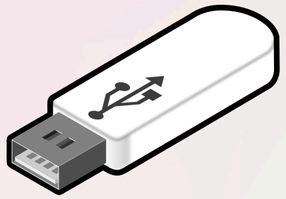
Preserving its cultural value.

Technology Research



Checker Board Recognition

- opencv
- image processing technique
- cells, marbles, boards detection
- mapping logic
- flask
- ...



Robotic Arm

- ESP32
- Servo Motor Control
- Inverse-kinematics
- Wi-Fi Communication
- Mobile Application Control
- ...

Artificial Intelligence Algorithm

- rule-based strategy
- graph-based pathfinding
- Minimax Algorithm
- Alpha-Beta Pruning
- Endgame Optimization
- ...

02

Chinese Checker Recognition

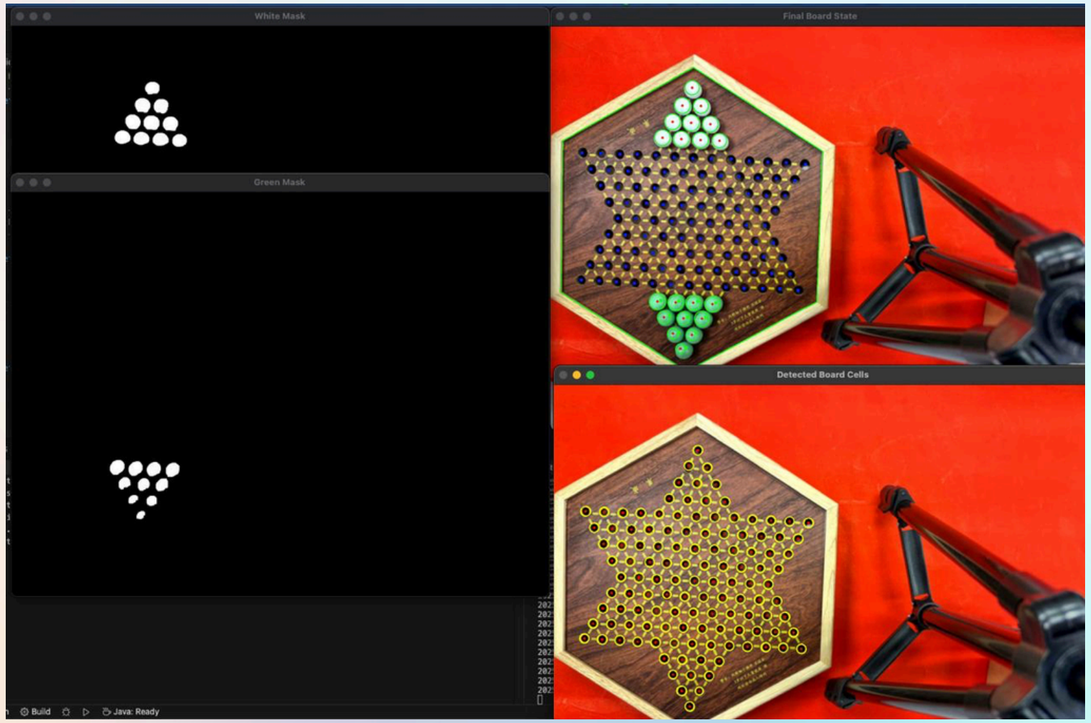
Challenges - Detection

- **Totally algorithmic, Self-design**

  Not using ML, don’t have data, time, resources...
  Not a project on ML but a product
  Completely modularized, could enhance in future.

- **Low Tolarance**

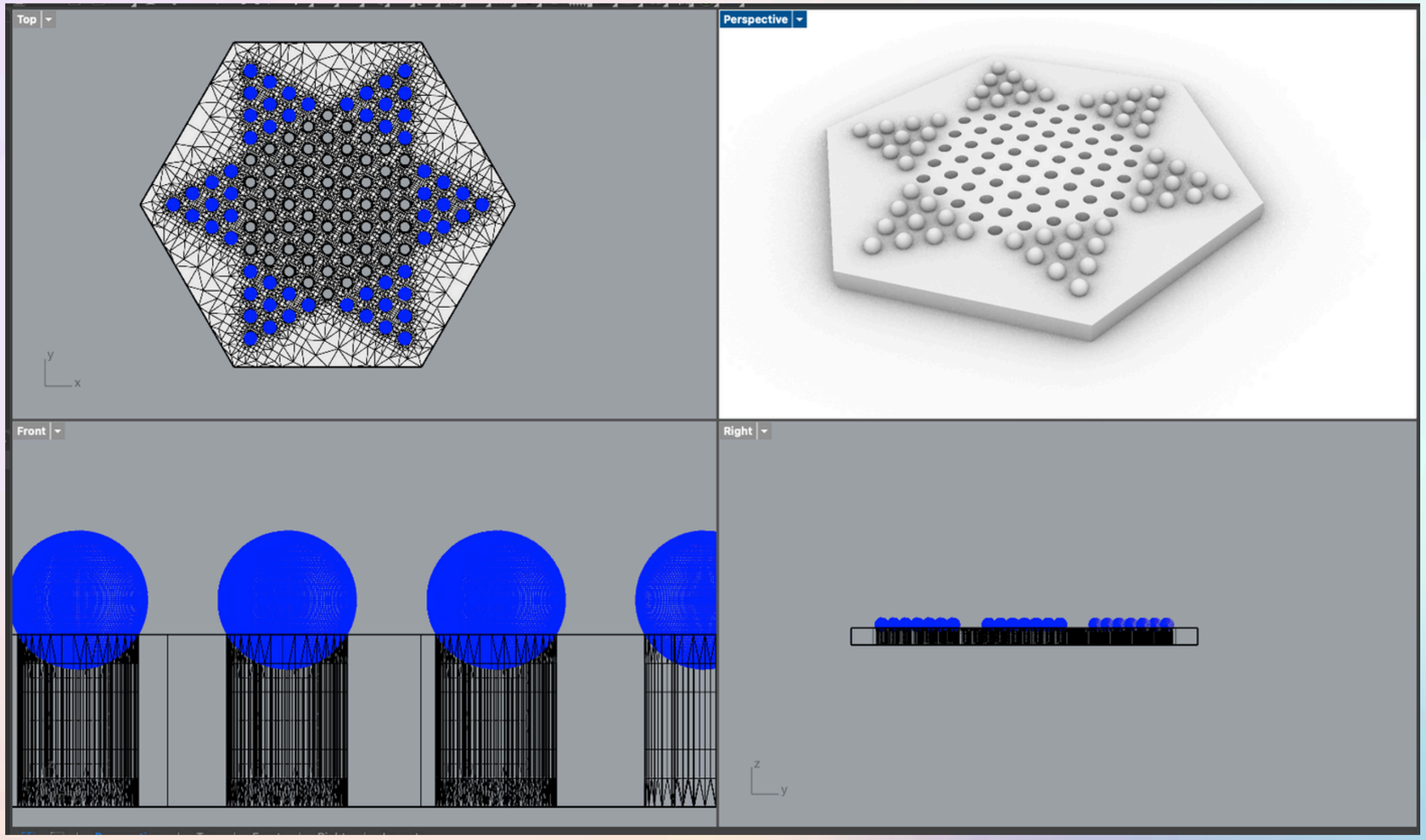
  Distances between marbles are too narrow



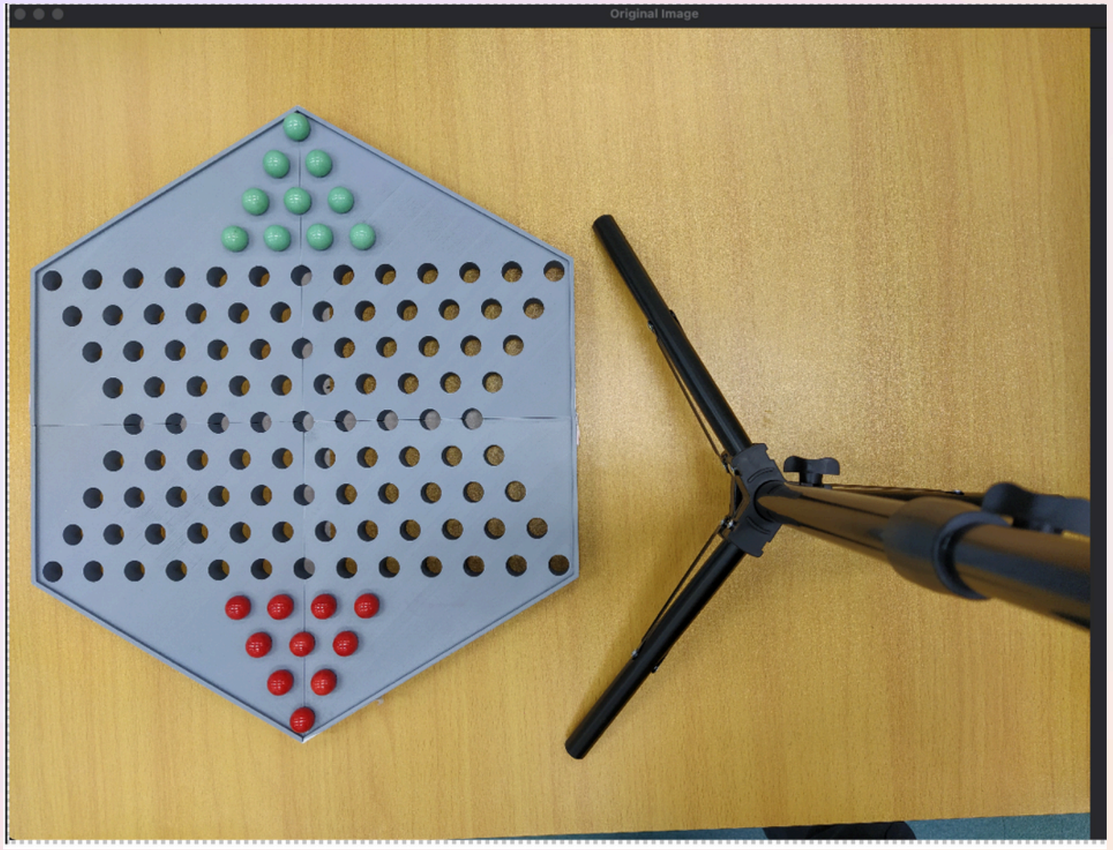
3D-modeling

- **Rhino self-design checker baord**

  Increase distance between marbles



Computer Vision - *Image Processing*

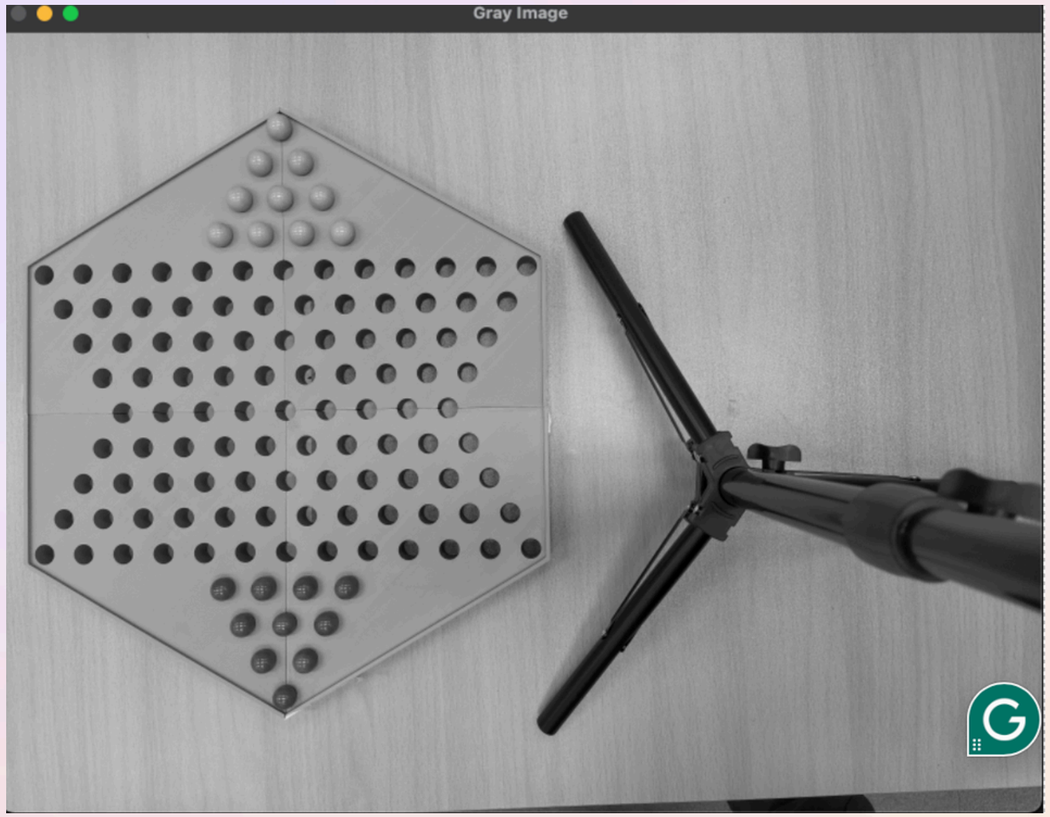


Adaptive Resizing

- Calculate scaling factor based on max dim (1600 pixel now)
- Resize the image using cv2.resize with cv2.INTER_AREA for smoother downscaling

Image Processing - Brightness and Contrast

Components



Convert Image to Grayscale

Compute the Histogram and Cumulative Histogram

- Calculated using cv2.calcHist
- Shows the cumulative count of pixels with intensities less than or equal to a given value

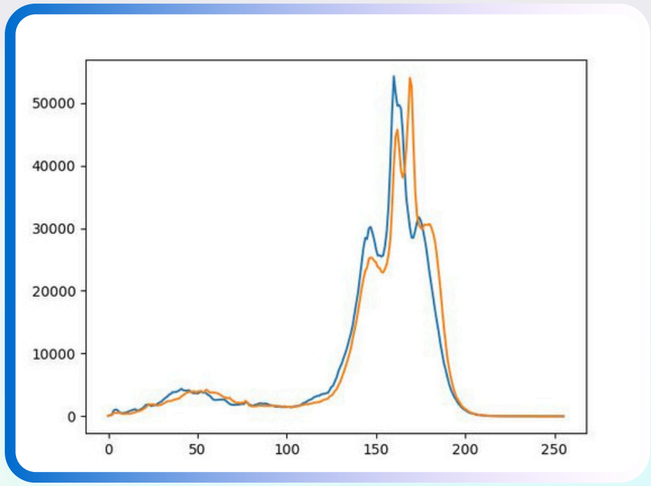
Determine Histogram Clipping Limits

Find Minimum and Maximum Gray Level After Clipping

Image Processing - Brightness and Contrast

Computation of the Scaling Factor (Alpha) and Bias (Beta)

- **Alpha determines the contrast scaling**
- Calculate the effective range of pixel intensities after clipping
- Alpha is calculated as 255.0 / (maximum_gray - minimum_gray)
- Make the clipped range fill the full 0-255 range



Histogram

- **Beta determines the brightness adjustment**
- It shifts the intensity range such that minimum_gray starts from 0, calculated as : -minimum_gray * **Alpha**

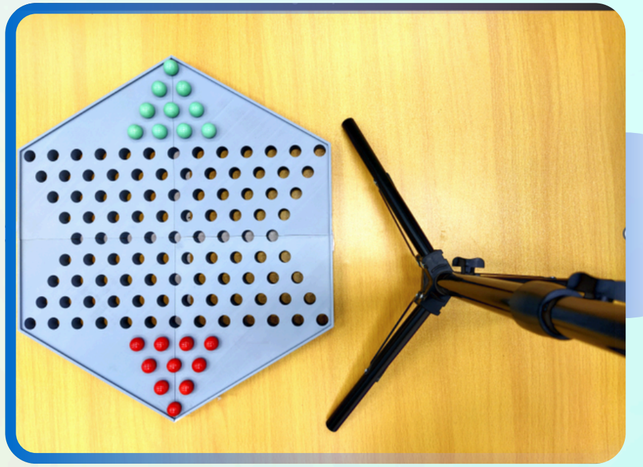


Image Processing - Color Space Transformation

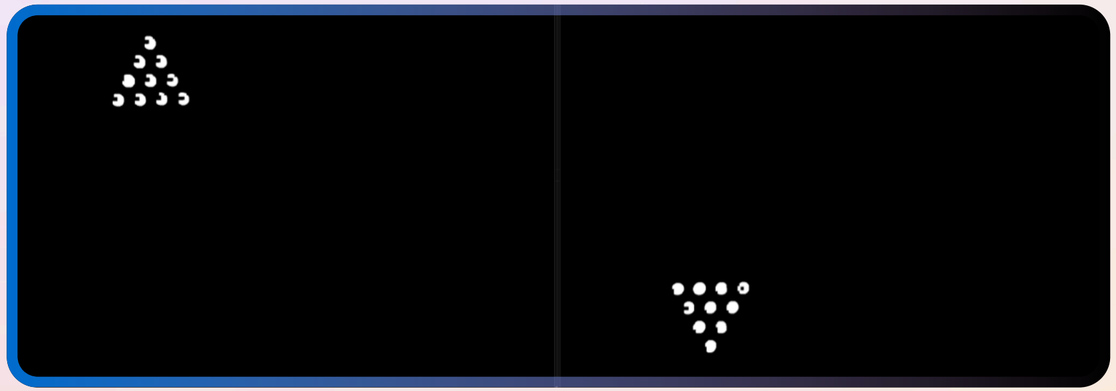
Gaussian Blur

How ?

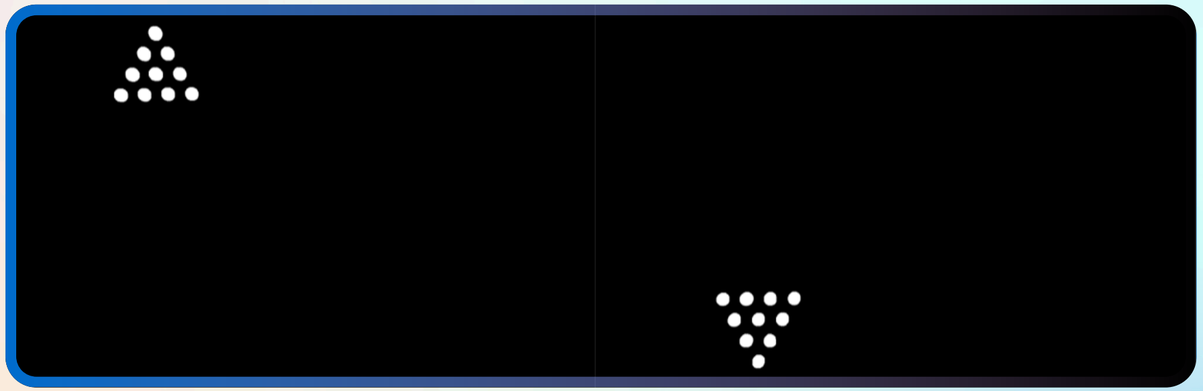
- **Smooths the image by reducing high-frequency noise**
- Kernel size (5, 5) determines the level of blurring
- Sigma is set to 0 for automatic s.d.

Why ?

- **Simplifies feature detection by removing small, irrelevant details**
- Makes Hough Circle detection more stable
- Helps merge small breaks in contours



Without Blur

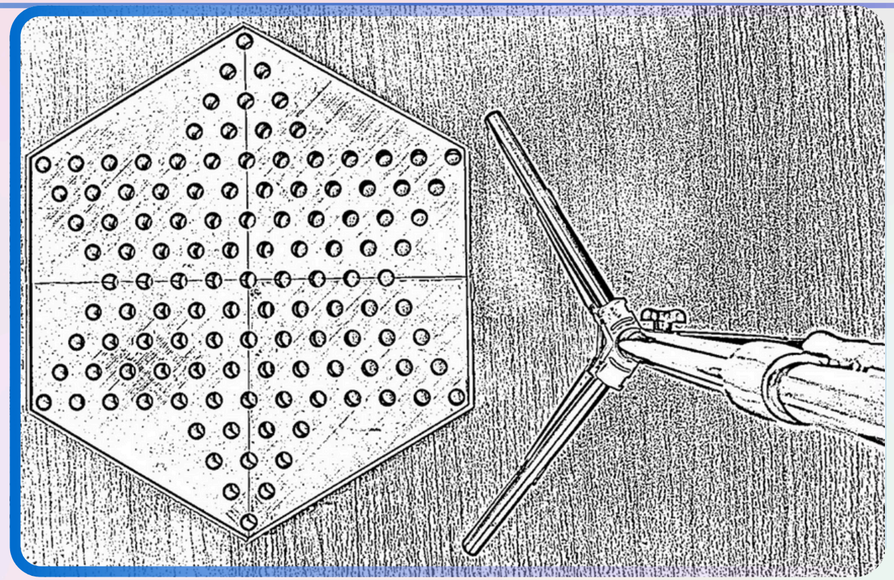


With Blur

2.1

Board Detection

Computer Vision - *Board Detection*



Convert to Grayscale >>>>> **Gaussian Blur AGAIN for board detection** >>>>> **Adaptive Thresholding**

- enhancing **edge detection** accuracy
- (Trial, not in used at last)

Computer Vision - *Board Detection*

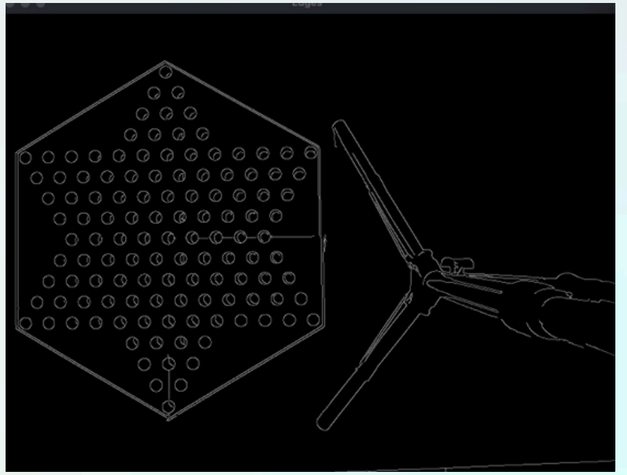
Canny Edge Detection

Canny recommended a upper:lower ratio between 2:1 and 3:1

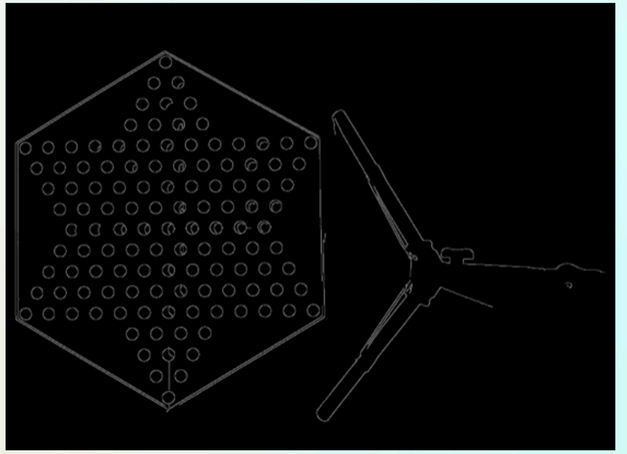
Purpose: Detects **strong edges** in the image

- Parameters
- Lower Threshold
- Upper Threshold

Why: Highlights the **boundaries of shapes** like the board's edges.

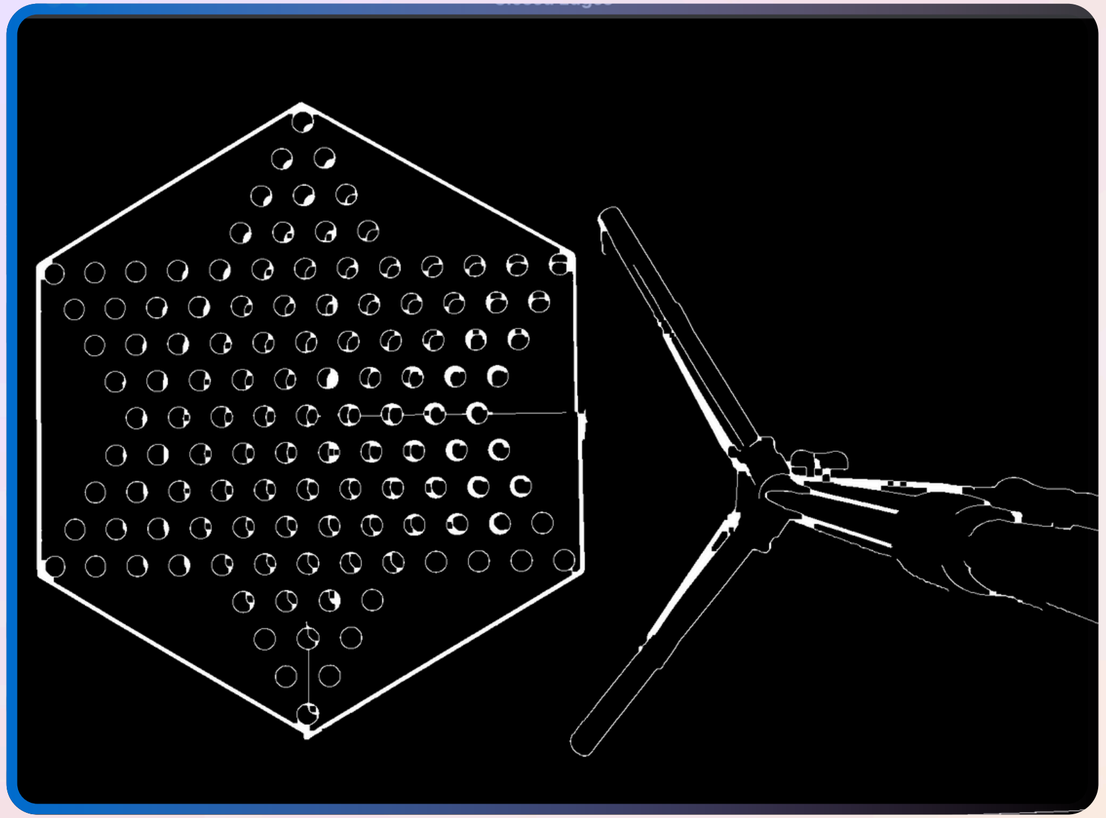


Lower Threshold: 50 Upper Threshold: 150



Lower Threshold: 85 Upper Threshold: 255

Computer Vision - *Board Detection*



Morphological Closing

Purpose:

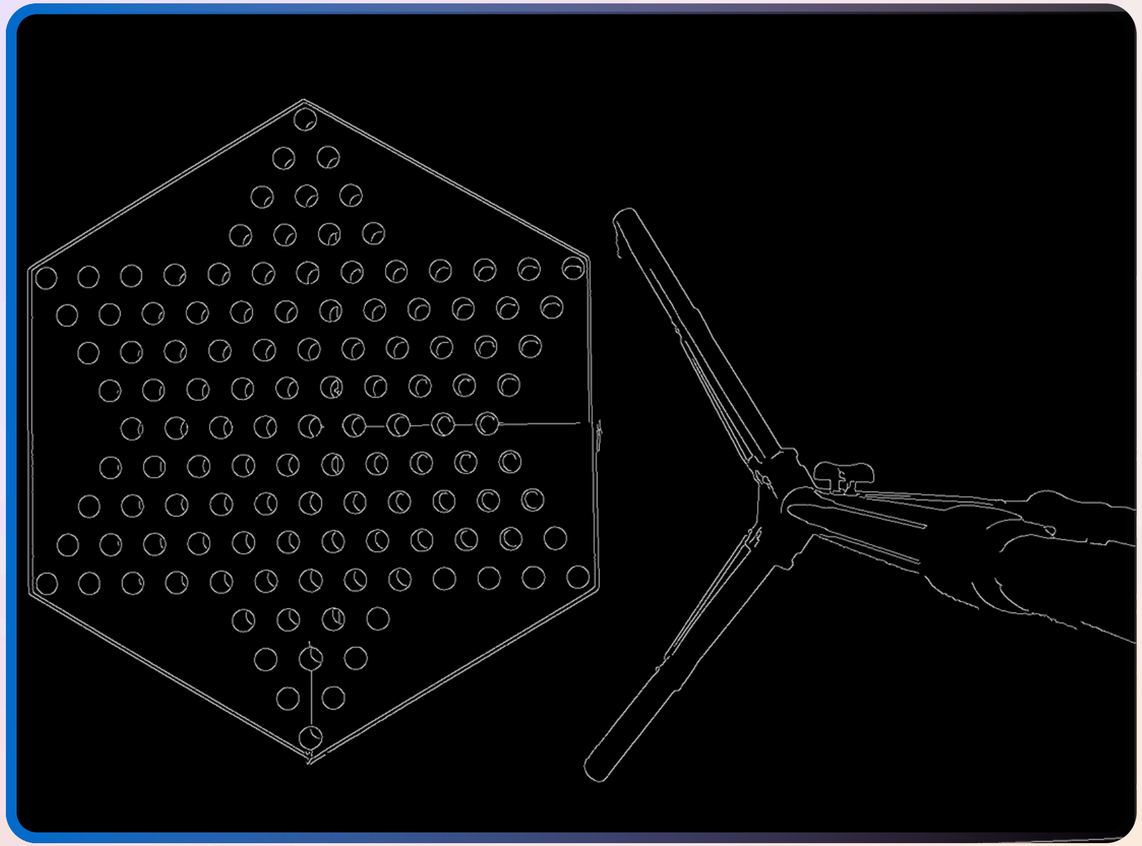
Fills small gaps in the edges to form a continuous boundary

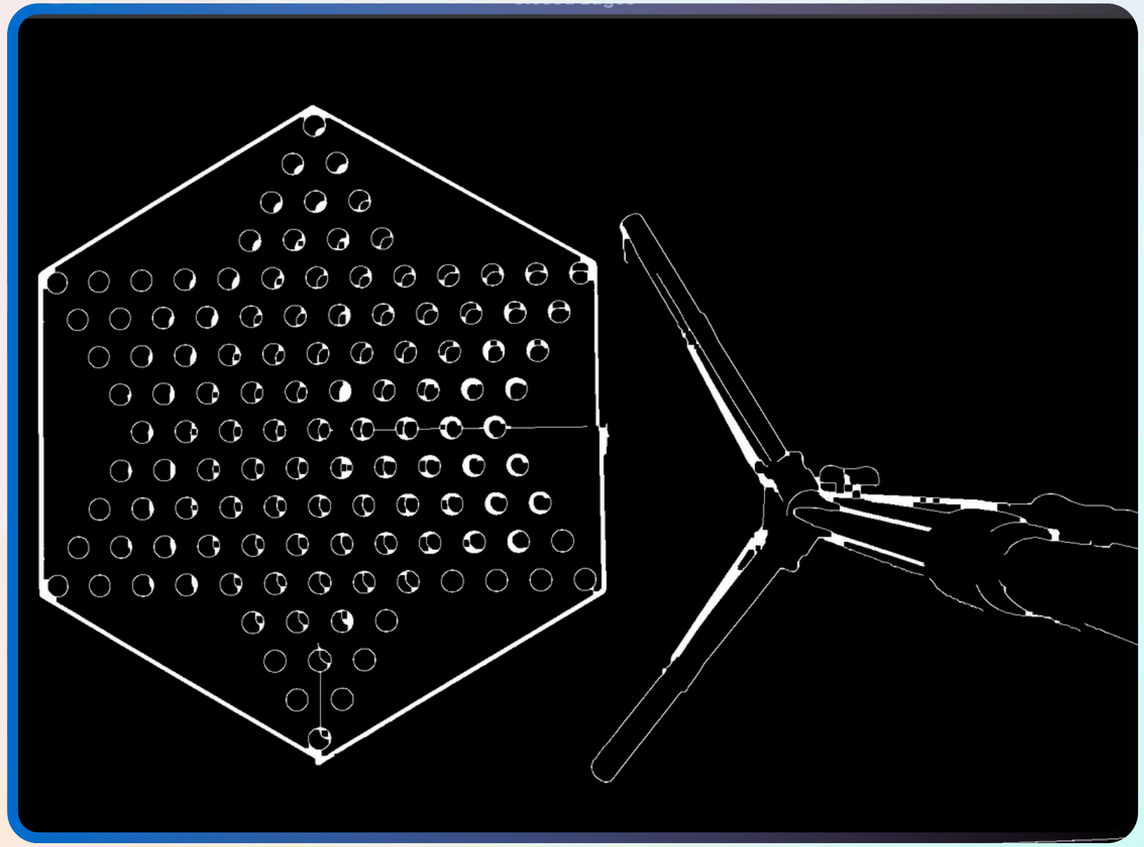
Kernel Size: (7, 7) determines the size of the gaps to close

Why:

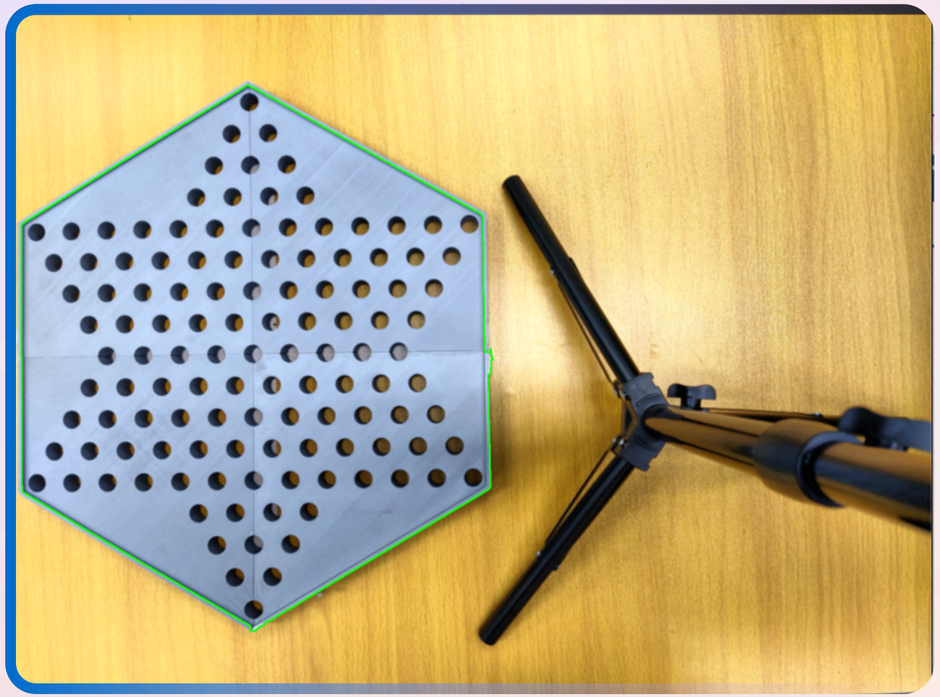
Ensures contours are well-defined for detection

Computer Vision - *Board Detection*





Computer Vision - *Board Detection*



Find Contours

Identifies all closed shapes (contours) in the processed image

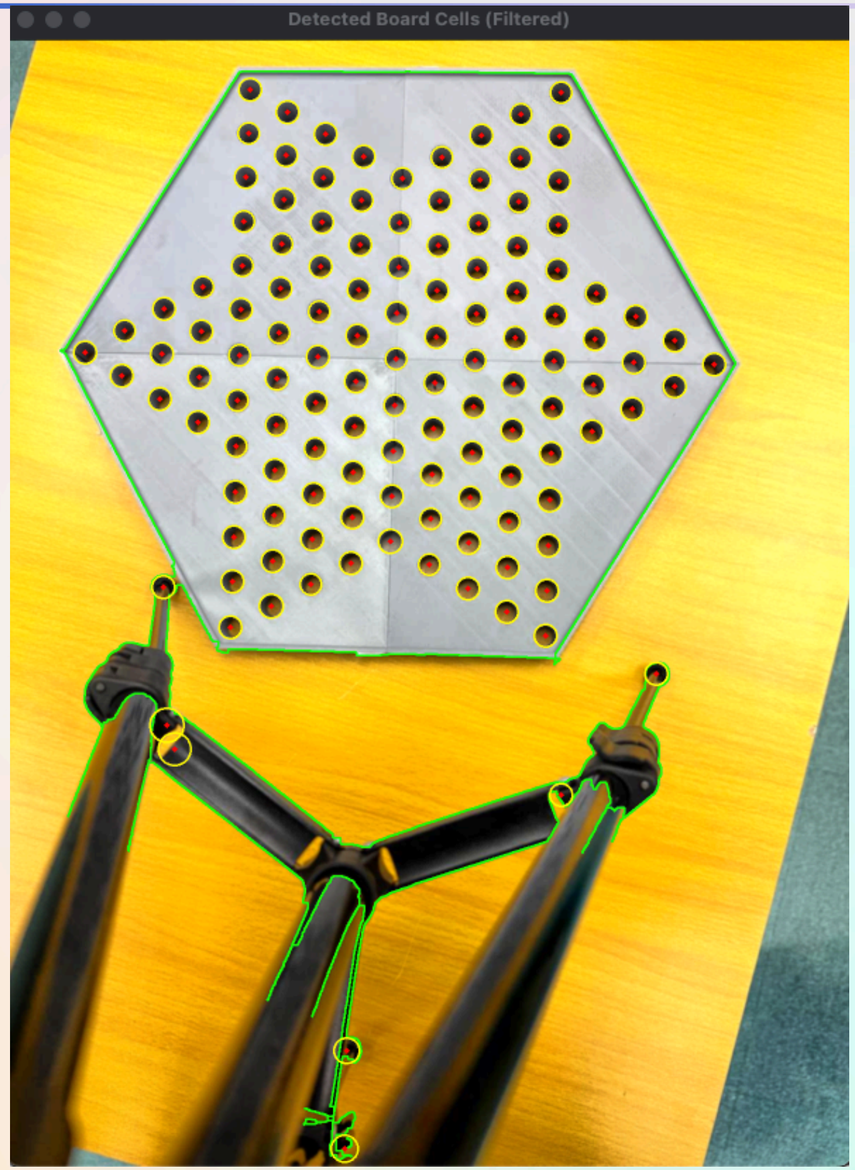
- RETR_EXTERNAL: Finds only the outermost contours in the image.
- CHAIN_APPROX_SIMPLE: Removes all redundant points along a straight line in a contour and keeps only the endpoints of the line.

Our Practice:

- Use **6 – 18 corners** for increased error tolerance.
- Filter out if the contour is not occupying 5% above the image area.
- Only the largest such contours fulfill the requirement are taken

Computer Vision - *Board Detection*

Failed Case



2.2

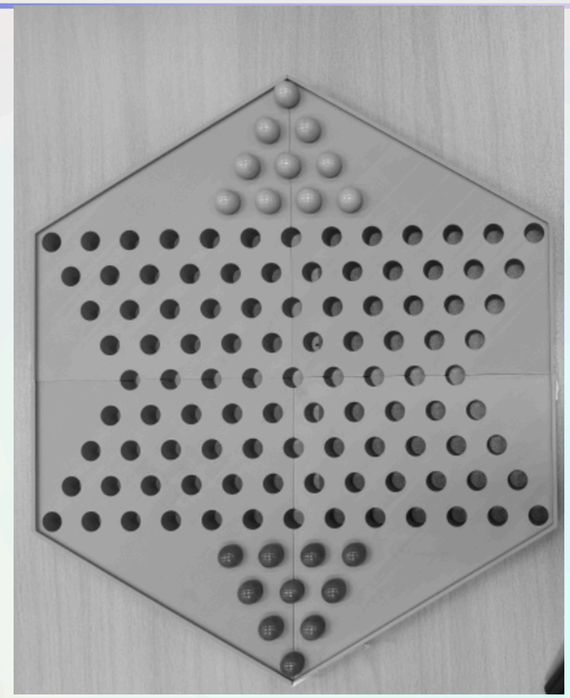
Cell Detection

Computer Vision - Cell *Detection*

Step 1: Convert to Grayscale

Converts the board image to grayscale to **simplify processing**.

Why: Hough Circle Transform requires single-channel intensity images.



Step 2: Apply Median Blur

Reduces noise while preserving edges

- Kernel Size: (5) determines the amount of blurring

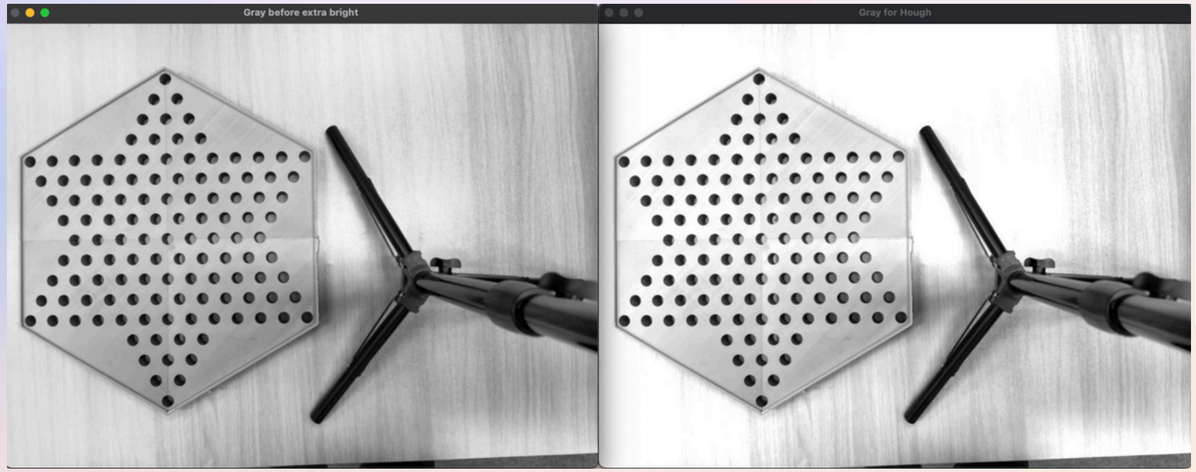
Why: Ensures cleaner input for circle detection by removing irrelevant small details.

Computer Vision - Cell *Detection*

Step 3: Enhance Brightness Again

alpha: Scales pixel intensities by 1.2

Why: Helps the Hough Circle Transform detect faint edges



Step 4: Hough Circle Transform

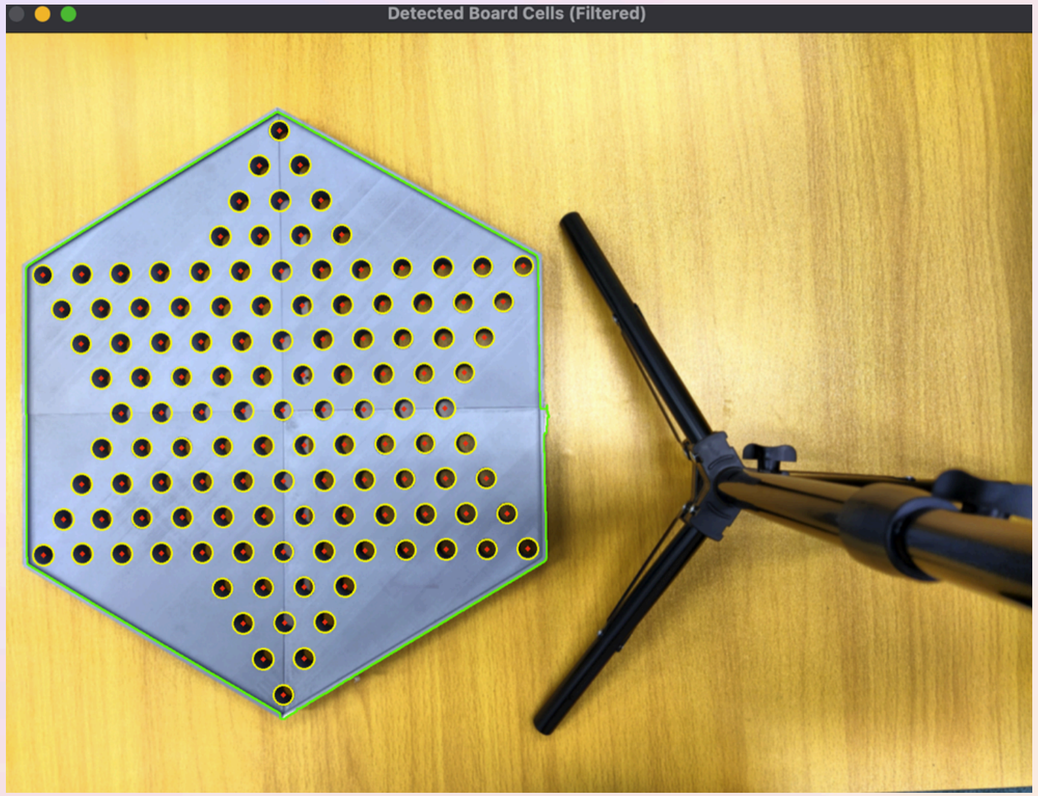
Identifies all circular shapes that could represent board cells

```
# Predefined Hough Circle Detection Parameters for Cells
HOUGH_DP = 1.1          # Inverse ratio of accumulator resolution
HOUGH_MIN_DIST = 30
HOUGH_PARAM1=500        # Canny high threshold, decrease will detect more edges
HOUGH_PARAM2 = 10   # Accumulator threshold, decrease will detect more circles
HOUGH_MIN_RADIUS = 15
HOUGH_MAX_RADIUS = 25
```

Circular shapes with **Radius 15-25**

Computer Vision - Cell *Detection*

Filter Circles by Board Contour



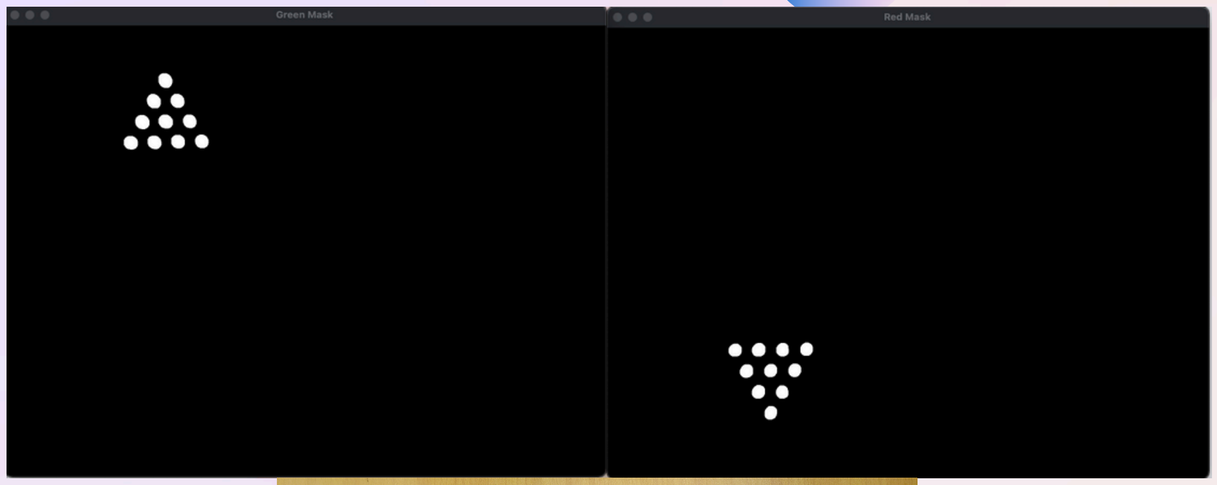
Converts detected circle coordinates to integers using np.int16

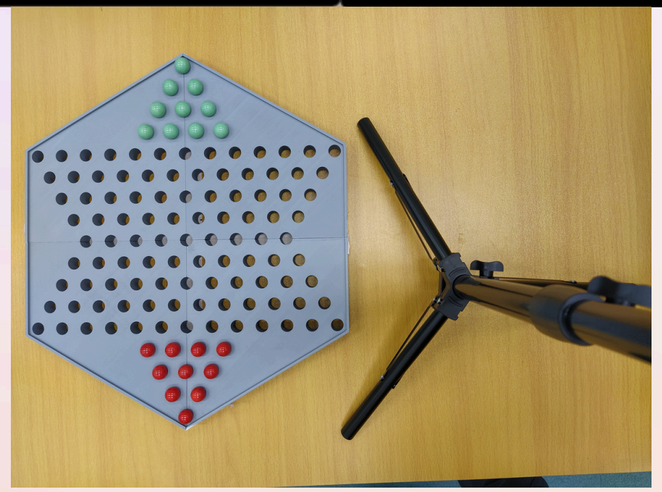
- Check if each circle’s center lies inside the board contour using **cv2.pointPolygonTest**.

2.3

Marble Detection

Computer Vision - *Marble Detection*





Find Marbles

Identifies Colours (Red & Green) in the image

Morphological cleaning

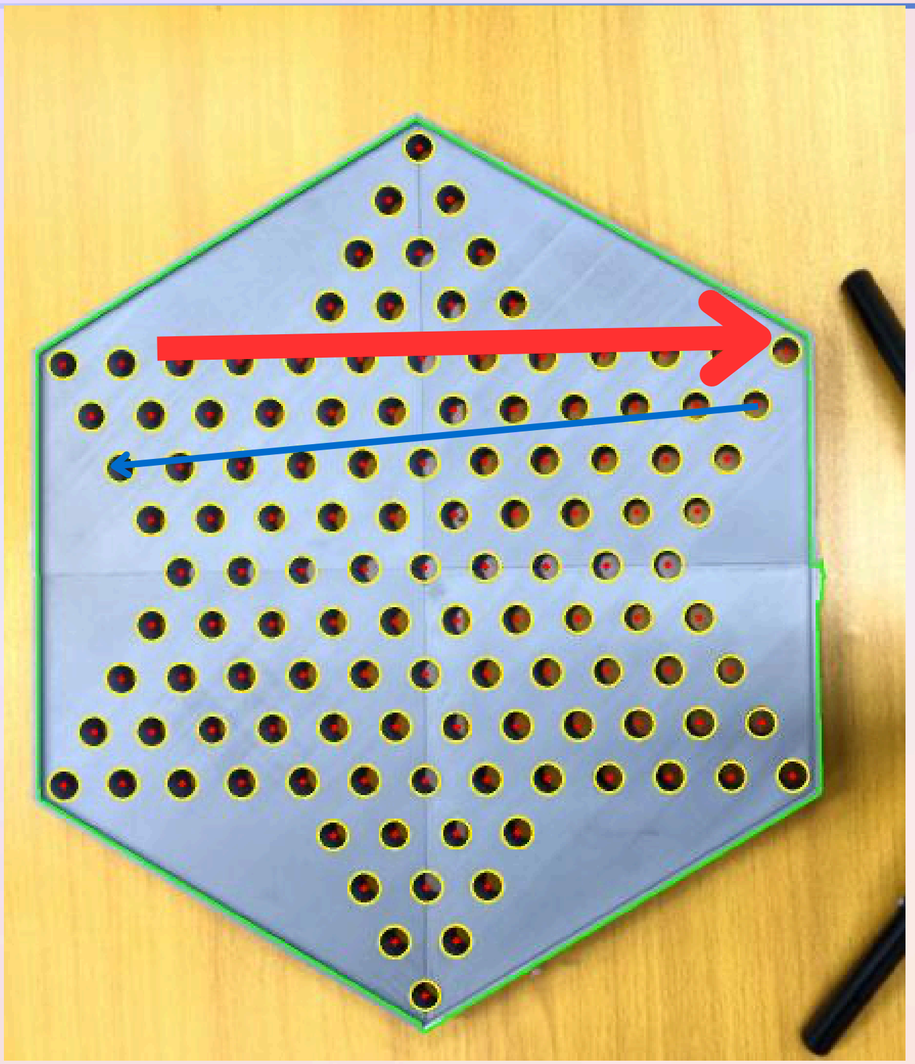
Circle Detection from Masks:

- 10 < Radius < 30
- circularity > 0.6, $4\pi \times (\text{area} / \text{perimeter}^2) > 0.6$
- Lies inside the board

2.4

Board State

Cells Assignment



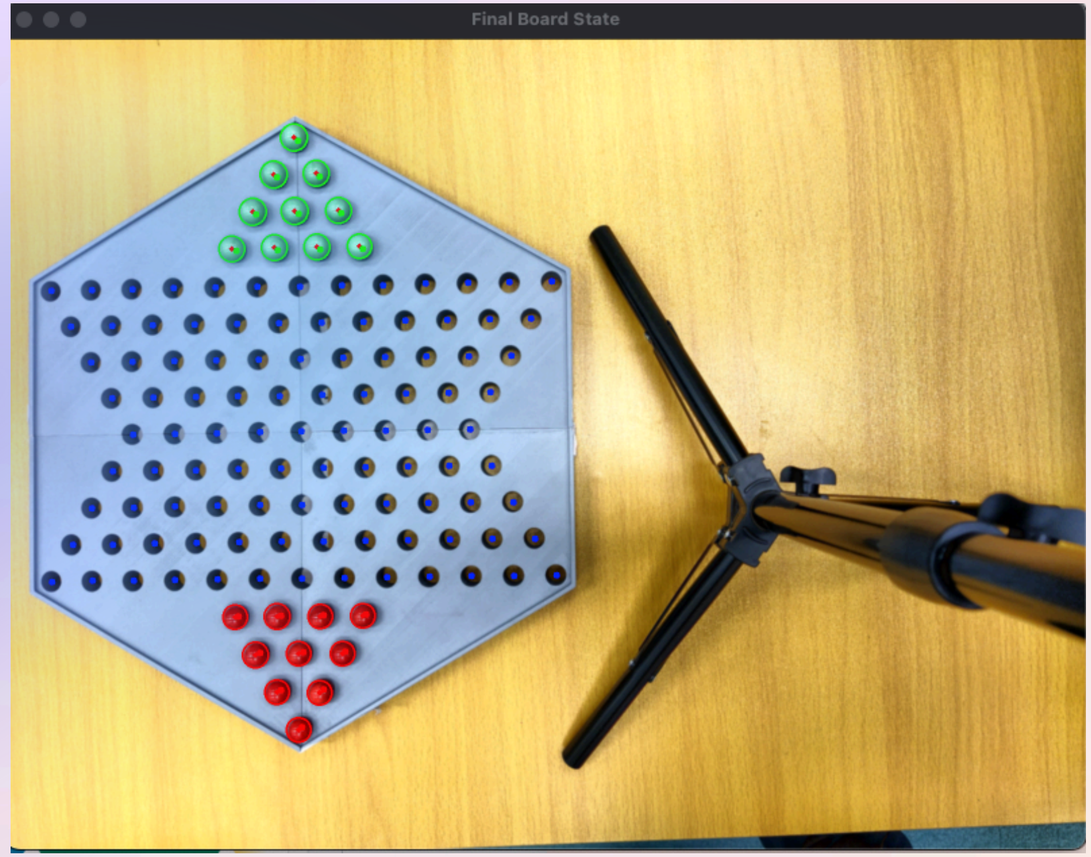
Assign Cells to Array

Group cells (x, y) into rows if their y-coordinates are within 'row_threshold' of each other.

1. Sort by ascending y (then x as a tiebreaker).
2. Start a new row when the y-difference is bigger than 'row_threshold', ~15
3. Sort each row by x ascending.
4. Return the grouped cells (flattened back into a single list, but now row-by-row).

```
board_layout = [
    [None],                 # row 0 has space for 1 cell
    [None, None],           # row 1 has space for 2 cells
    [None, None, None],     # row 2 has space for 3 cells
    [None, None, None, None],     # row 3 has space for 4 cells
    [None, None, None, None, None, None, None, None, None, None, None, None, None],  # row 4 has space for 13 cells
    [None, None, None, None, None, None, None, None, None, None, None, None],  # row 5 has space for 12 cells
    [None, None, None, None, None, None, None, None, None, None, None],          # row 6 has space for 11 cells
    [None, None, None, None, None, None, None, None, None, None],                # row 7 has space for 10 cells
    [None, None, None, None, None, None, None, None, None],                      # row 8 has space for 9 cells
    [None, None, None, None, None, None, None, None, None, None],                # row 9 has space for 10 cells
    [None, None, None, None, None, None, None, None, None, None, None],          # row 10 has space for 11 cells
    [None, None, None, None, None, None, None, None, None, None, None, None],    # row 11 has space for 12 cells
    [None, None, None, None, None, None, None, None, None, None, None, None, None],  # row 12 has space for 13 cells
    [None, None, None, None],   # row 13 has space for 4 cells
    [None, None, None],         # row 14 has space for 3 cells
    [None, None],               # row 15 has space for 2 cells
    [None],                     # row 16 has space for 1 cell
]
```

Marbles Assignment



Assign Cells to Array

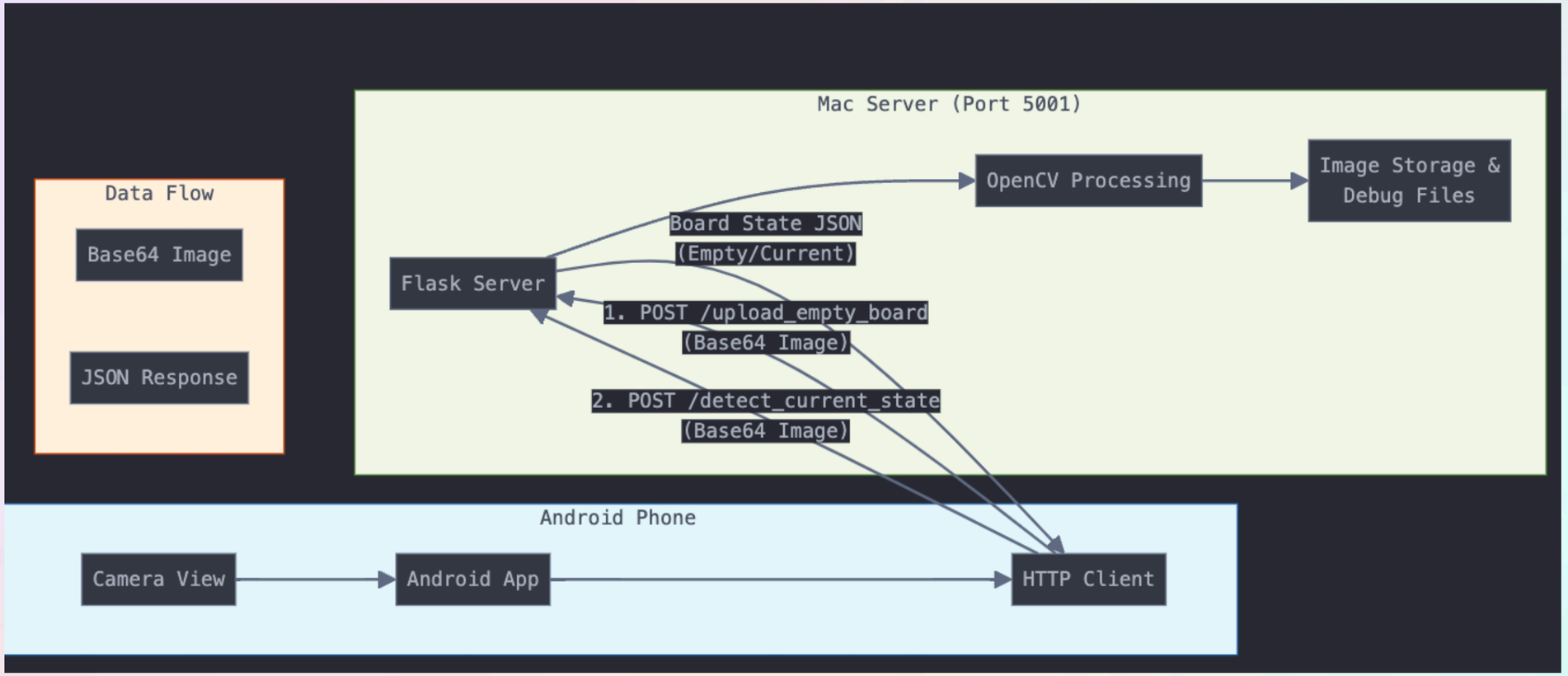
- Assign the marbles to the nearest cell
- Using Euclidean distance to sort

```
G
G G
G G G
G G G G
. . . . . . . . . . . . . .
. . . . . . . . . . . . . .
. . . . . . . . . . . . . .
. . . . . . . . . . . . . .
. . . . . . . . . . . . . .
. . . . . . . . . . . . . .
. . . . . . . . . . . . . .
. . . . . . . . . . . . . .
. . . . . . . . . . . . . .
R R R R
R R R
R R
R
```

2.5

Communication Architecture

Server Communication Architecture

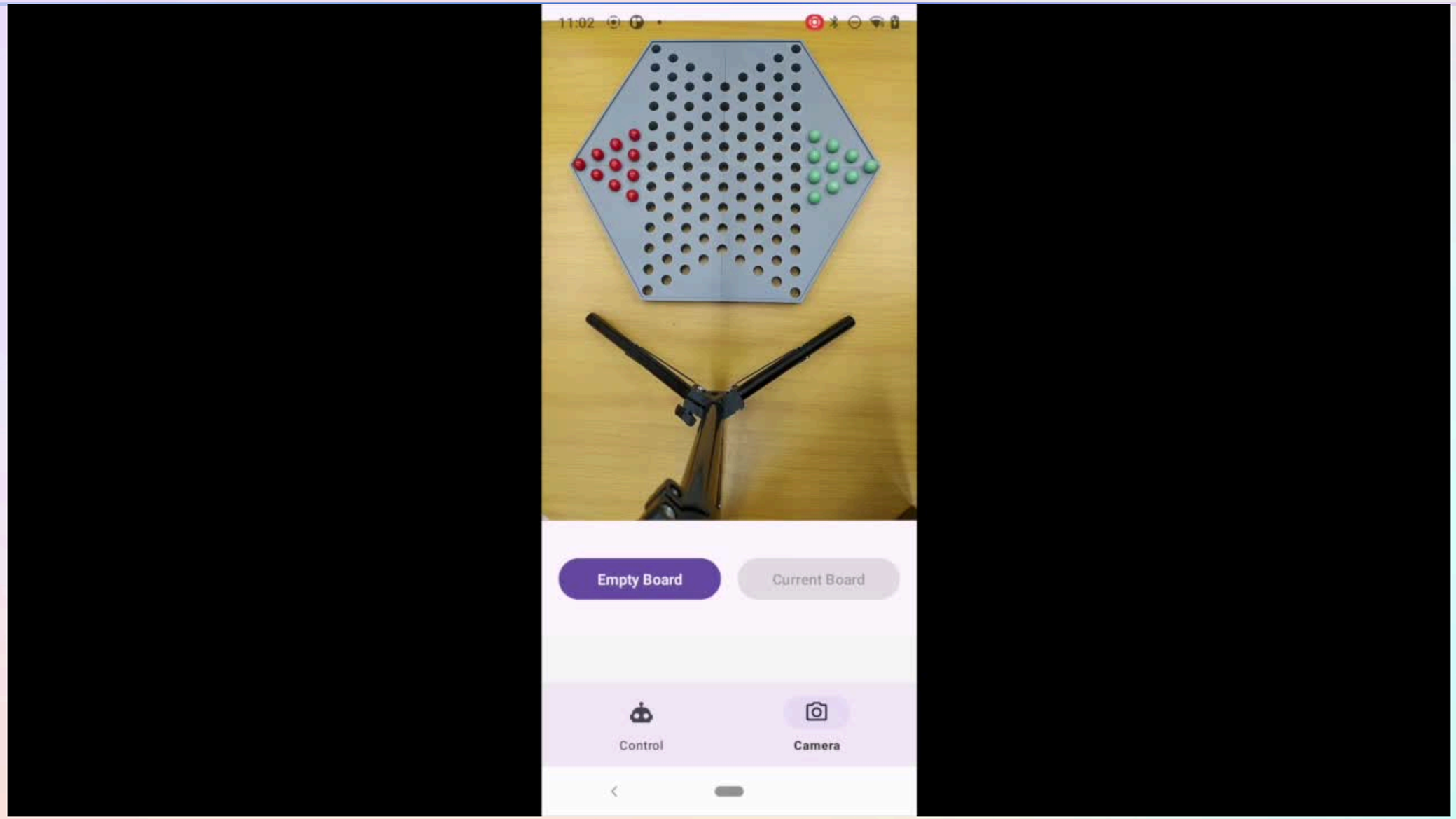


Future: Migrate to AWS, or migrate to phone application

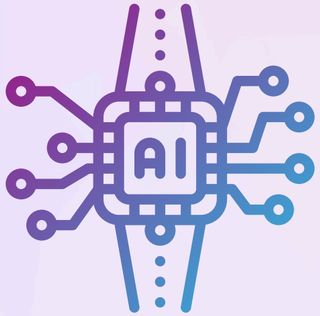
2.6

Quick Demo

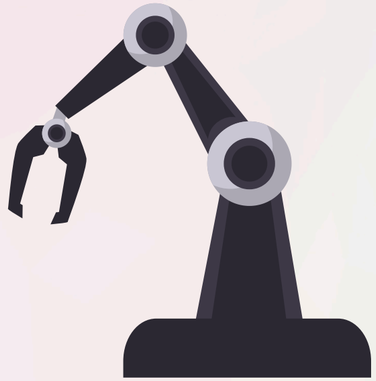
Quick Demo



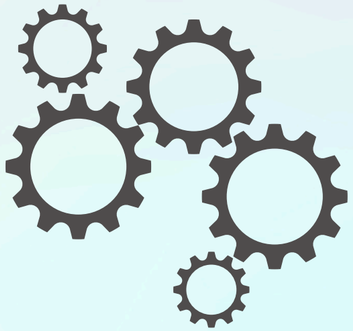
Future Work



AI Strategy for Chinese Checkers



Robotic Arm Mechanism



Integration of AI and Arm

Refining the AI Algorithm

Goal:

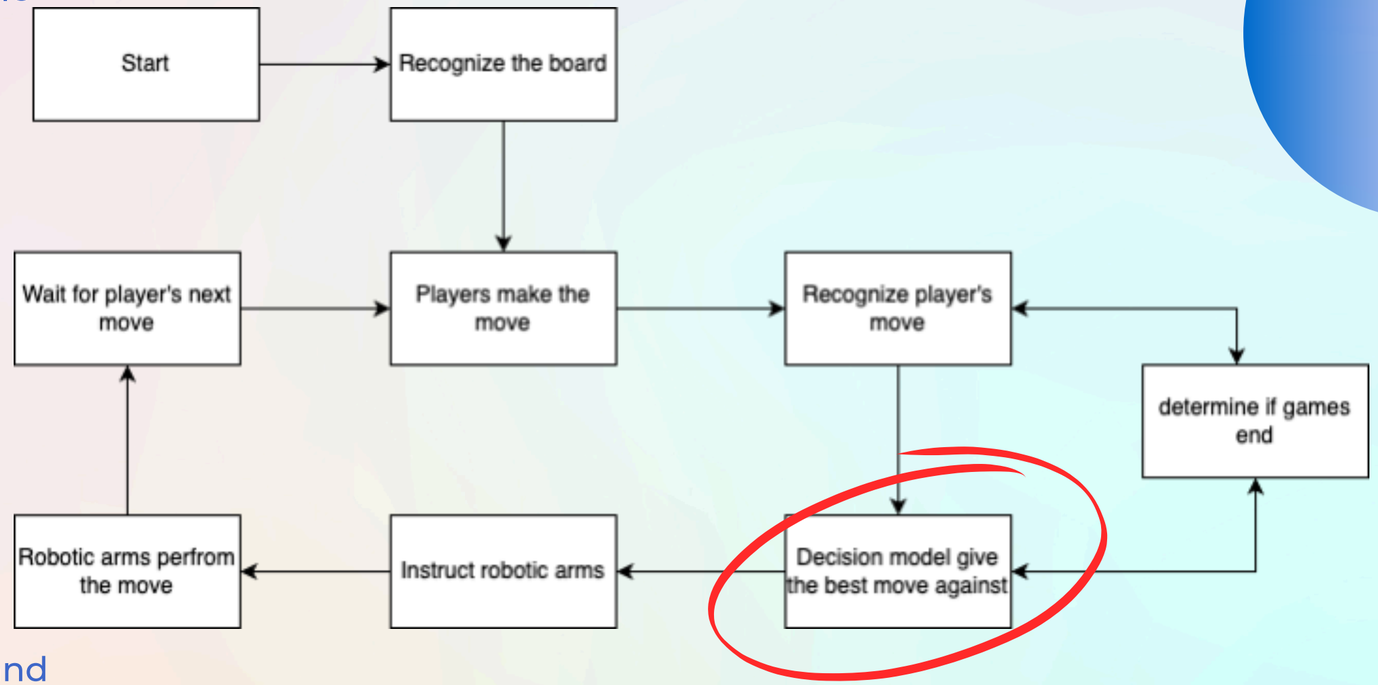
Develop an AI capable of determining the best possible move with the highest winning probability

Focus Area:

- Rule-based strategy
- graph-based pathfinding
- Minimax Algorithm
- Alpha-Beta Pruning
- Endgame Optimization

Planned Outcome:

A strategic AI that adapts to various game scenarios and player strategies



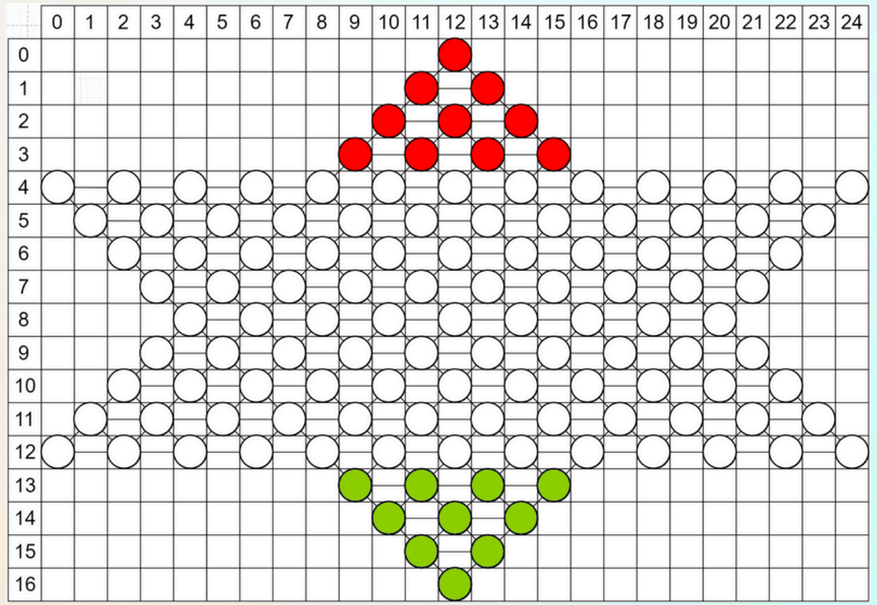
Perfecting the Robotic Arm Mechanism

Goal:

Enhance the robotic arm's ability to pick up and place checker pieces with precision

Focus Area:

- Customize the grip or add a sucker at the end of the arm to securely handle checker pieces
- Map the checkerboard into a structured array where each cell corresponds to an X, Y, Z coordinate



Planned Outcome:

A robotic arm that can reliably and precisely interact with every cell on the checkerboard

Autonomous Chinese Checkers Playing Robot Arm

Thank You.
Thank You.
Thank You.