mia robotics

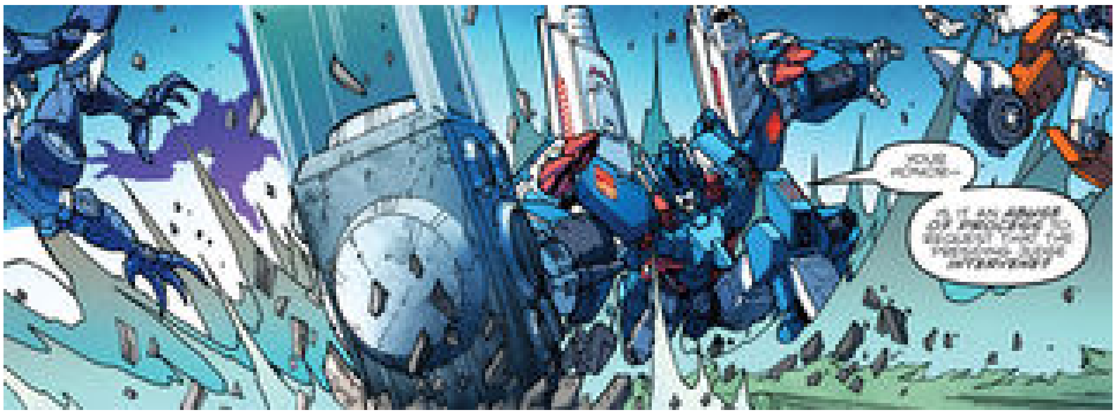
ELECTRICAL TEAM TRAINING

TASK 11

Group Tasks

Note: you will submit a repo containing both task 11.1 and 11.2 as separate branches,
And you will also submit a video of Task 11.1

Task 11.1: Wack-a-Decepticon



Ultra Magnus has arrived at the Autobots' training grounds, where mischievous Decepticon drones keep popping out of underground tunnels. Instead of fighting with blasters, Magnus decides to test his strength and precision with his **war hammer** in a "Whack-a-Mole" style training game.

Description

You will create a Wack-a-Mole game using Opencv and YOLO. The game will be played using hand gestures detected through a live video stream. A grid will be displayed on the screen, the size of the grid is customizable depending on how difficult the game will be, and Decepticons will appear randomly then disappear after a couple of seconds. Players will make moves by performing specific gestures. A 🖐gesture will select and highlight a cell, while the ✊gesture will wack the selected and highlighted cell.

Requirements

- Class Detection:
  - Implement gesture detection using YOLO to detect and differentiate this two specific hand gestures using a custom made model:
    - 🖐 for "select"
    - ✊ for "wack"
- Grid Layout:
  - Divide the screen into a nxn grid.
    - The size of the grid can be customized to increase/decrease the difficulty of the game, as the higher the n is the smaller the cells become
  - Decepticons (which can be represented as color dots) will appear randomly in the cells of the grid
  - Players will make gestures in specific grid areas to do the following actions:
    - 🖐 will highlight and select the cell its detected at
    - ✊ will wack the selected cell, if the wacked cell contains a Decepticon, then that Decepticon should be removed from the cell and player should receive a point
    - **Do note that players must select the cell first before wacking,** If the program detected a ✊ then it should wack the previously selected cell even if the gesture wasn't detected inside the cell
    - **BONUS:** A tracking cursor should appear at the middle of the detection box of the select gesture, where it draws a temporarily line of where the hand moved

- **Bonus:** Game system:
    - The game should be customizable, where players can select how long the game should go, the size of the grid and how fast the Decepticons should disappear from the cell
    - There should be a point system where a player receives a point for every time they wacked a decepticon, and a penalty for every Decepticon that they missed
- Dataset Gathering:
    - Teams must collect and annotate their own dataset of hand gestures from scratch to train the YOLO model.
    - Teams are responsible for gathering high-quality data for both gesture types and ensuring diversity in the training set.
- Documentation:
    - Teams must write a proper README, documenting the entire process from data collection, model training, and game implementation to testing and evaluation.
    - It should also include the approach, methodology, results, and any challenges faced during the project.

Task 11.2 Classical Detection

The Autobots are running out of time. A Decepticon strike is imminent, and training a neural network would take days they don't have. With only four hours to prepare, they turn to classical detection methods as they are fast, reliable, and effective without massive datasets.

Requirements

You are provided with a dataset of 17 images each containing different geometric shapes. The task is to design a system that can accurately detect and classify these shapes using **classical**

computer vision techniques (rather than deep learning).
The performance of the system will be evaluated using **accuracy**, defined as:

$$\text{Accuracy} = \frac{TP}{TP + FP + FN}$$

Where:
TP = True Positives (correctly detected shapes)
FP = False Positives (non-shapes detected as shapes)
FN = False Negatives (missed shapes)

Individual Tasks

Task 11.3 Shape detection

Requirements

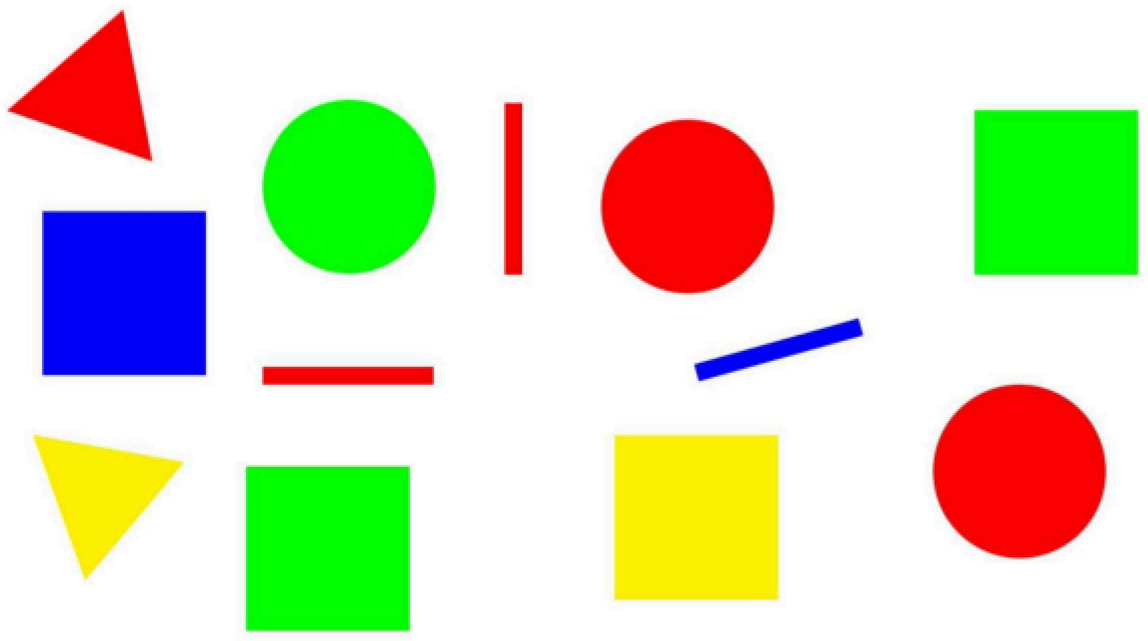
1. **Shape Detection**:
    - Detect and classify four geometric shapes:
        - Rectangle
        - Square
        - Circle
        - Triangle
    - Use classical methods like edge detection, contour analysis, Approximation algorithms and geometric properties to identify each shape.
2. Color Identification:
    - For each detected shape, identify its color (e.g., red, green, blue, yellow) using OpenCV's color space conversion (like HSV or RGB) and color thresholding techniques.
3. Implementation:
    - Build the program in Python using OpenCV.

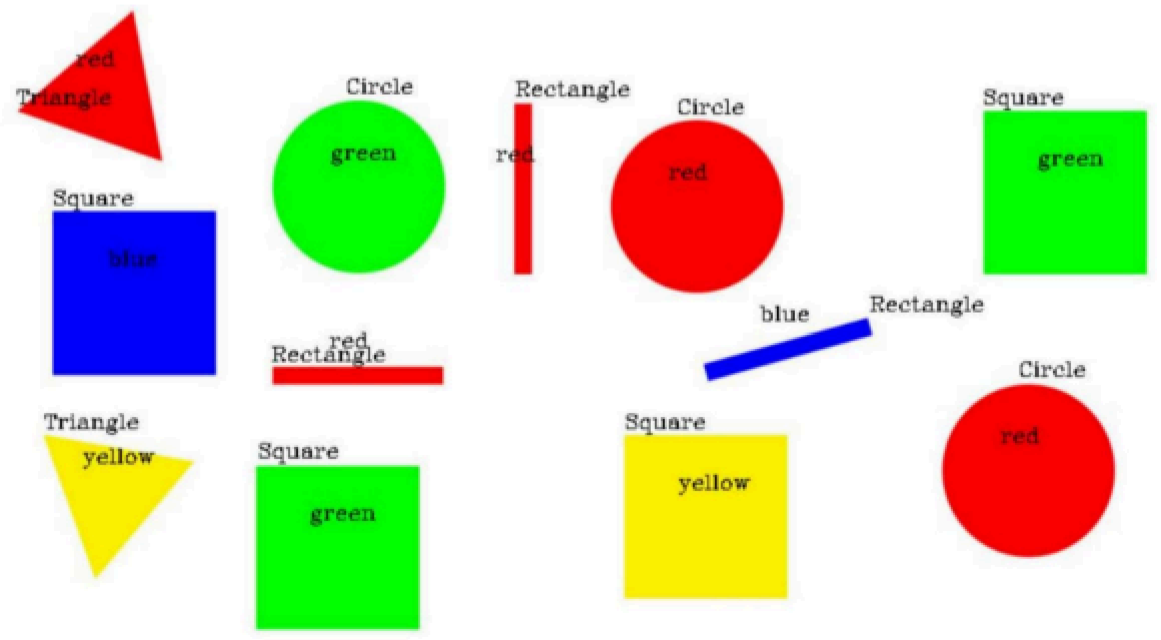
- The program should process an input image of the following test image to detect the shapes and their colors.

4. Documentation:

- Include a README file explaining:
  - How you approached the task using classical methods.
  - The algorithms and techniques used for shape and color detection.
  - Any challenges or insights gained from working without modern deep learning tools.
- You should also include in the readme a picture of the results of the test image

Test image: test.jpg





Submission

- You will submit the link of the repo containing your group task
    - You will also submit the link of the video of task 11.1
- You will submit the link of the repo containing task 11.3
- The Task's deadline is at Friday, 5/9 11:59 PM.
- Submission link: https://forms.gle/N99Z42B7PSRKgTjv8