Drawing robot playing Tic-Tac-Toe

Project 3.1

Department of Data Science and Artificial Intelligence
Maastricht University, 2021-2022

Stijn Overwater, Matthijs Kusters, Jean Janssen, Adèle Imparato, Daniël Tossaint

Table Of Contents {

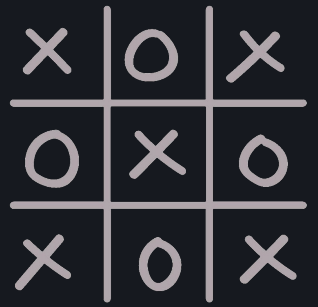
}

What we have:

- ROBOTIC ARM
- KNOWLEDGE OF KINEMATICS
- SETUP + CAMERA

Our goal:

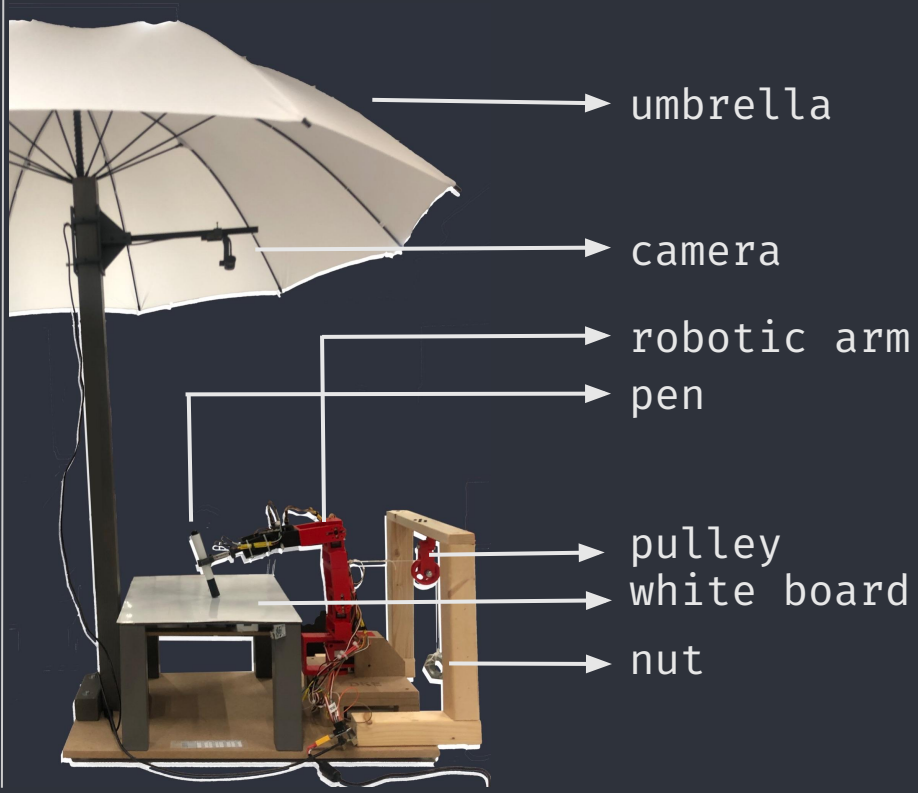
- ROBOTIC ARM ABLE TO SKETCH ACCURATELY
- ROBUST COMPUTER VISION



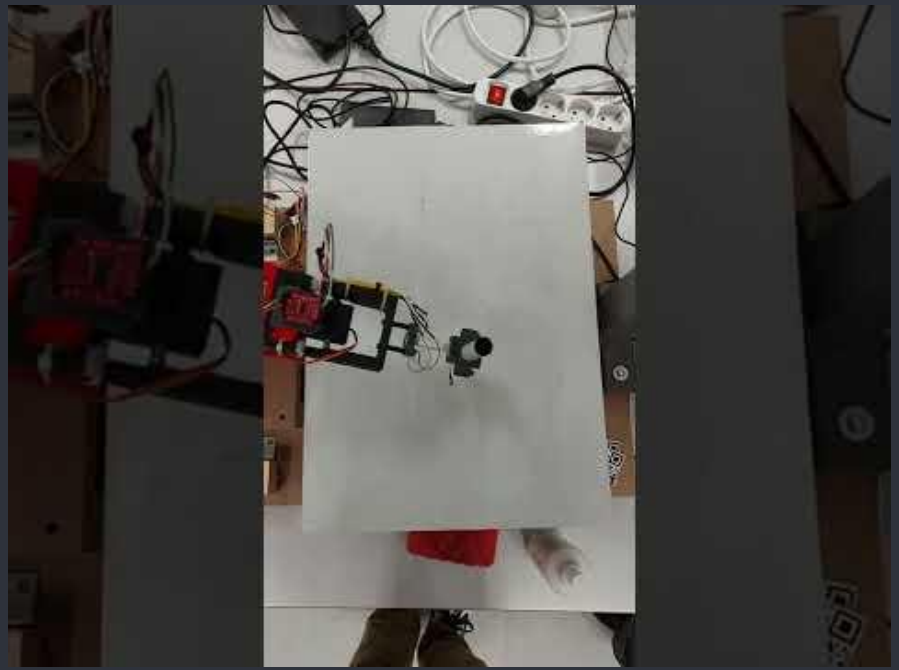
- **RQ1:** How accurate is the deep neural net described at categorizing cells compared to the base model CNN ?

- **RQ2:** For the computer vision, what is the optimal method for thresholding with regards to adaptive method and block size ?

- **RQ3:** How robust is the robotic arm at drawing lines ?


umbrella
camera
robotic arm
pen
pulley
white board
nut

Drawing a plus

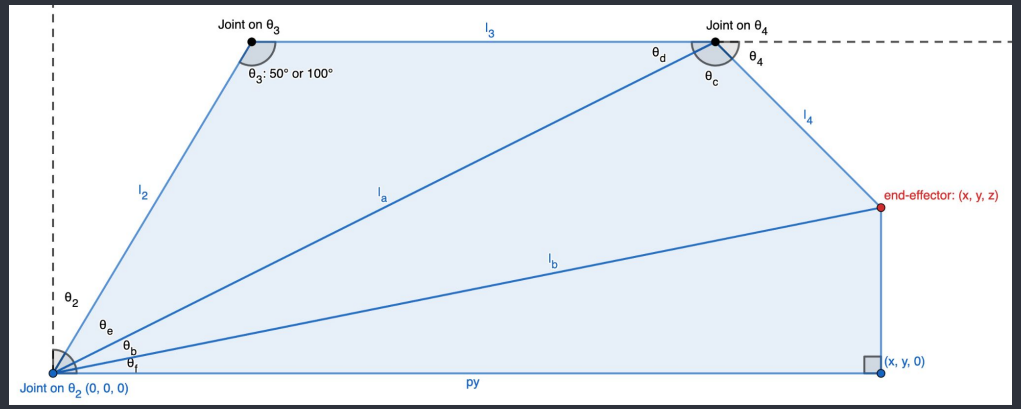


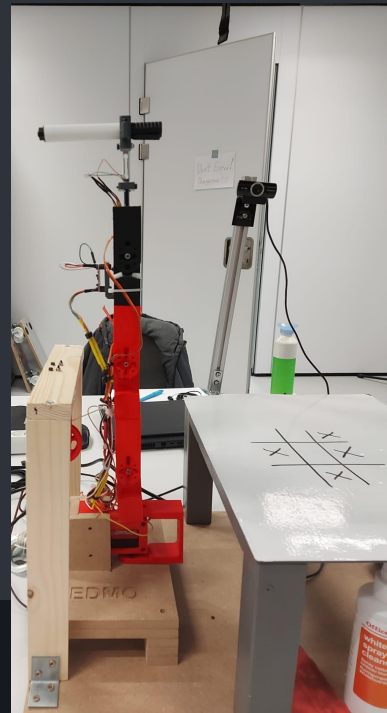
Drawing a box

Forward Kinematics

$$z = l_2 \cos(\theta_2) + l_3 \cos(\theta_2 + \theta_3) + l_4 \cos(\theta_2 + \theta_3 + \theta_4) + l_{pen} \cos(\theta_2 + \theta_3 + \theta_4 + 90^\circ)$$

$$l_{xy} = l_2 \sin(\theta_2) + l_3 \sin(\theta_2 + \theta_3) + l_4 \sin(\theta_2 + \theta_3 + \theta_4) + l_{pen} \sin(\theta_2 + \theta_3 + \theta_4 + 90^\circ)$$

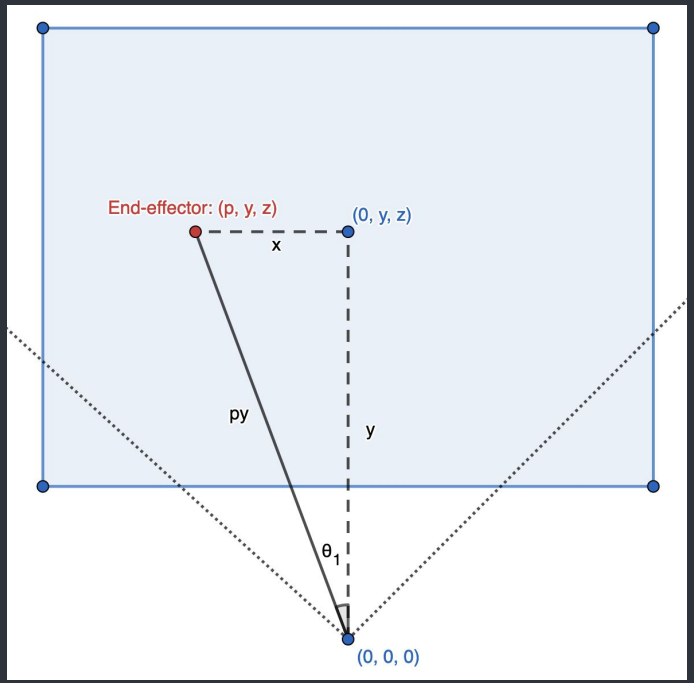




Forward Kinematics

$$x = l_{xy} \sin(\theta_1)$$

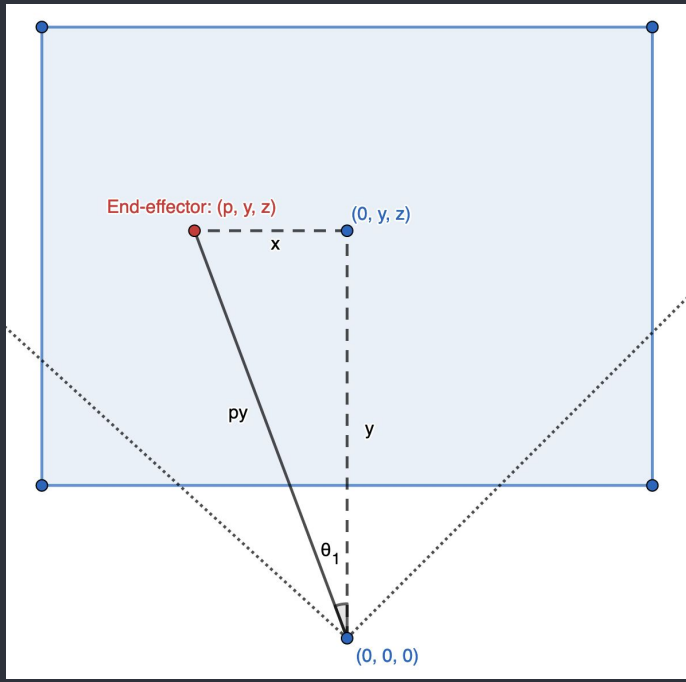
$$y = l_{xy} \cos(\theta_1)$$



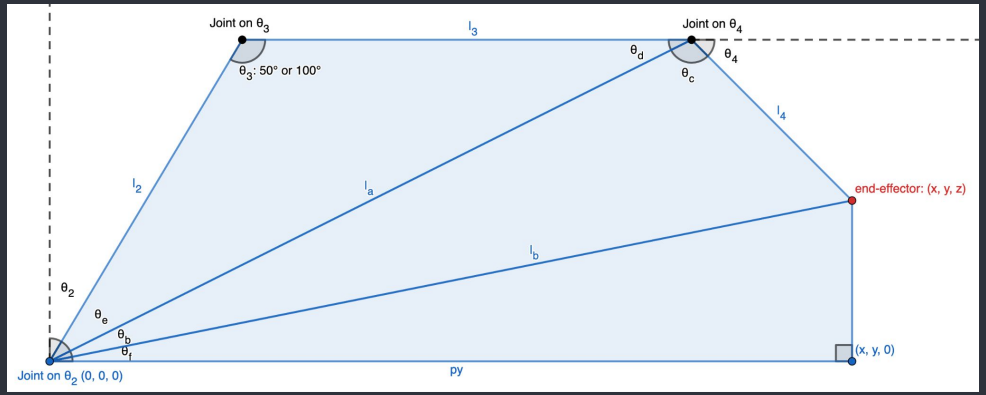
Inverse Kinematics: Top-down view

- py = x^2 + y^2
- z = 1
- Length of board in y-dimension = 30.5
- Width of board in x-dimension = 42.5

$$\theta_1 = atan^2(x, y)$$



Inverse Kinematics: Cosine Rule

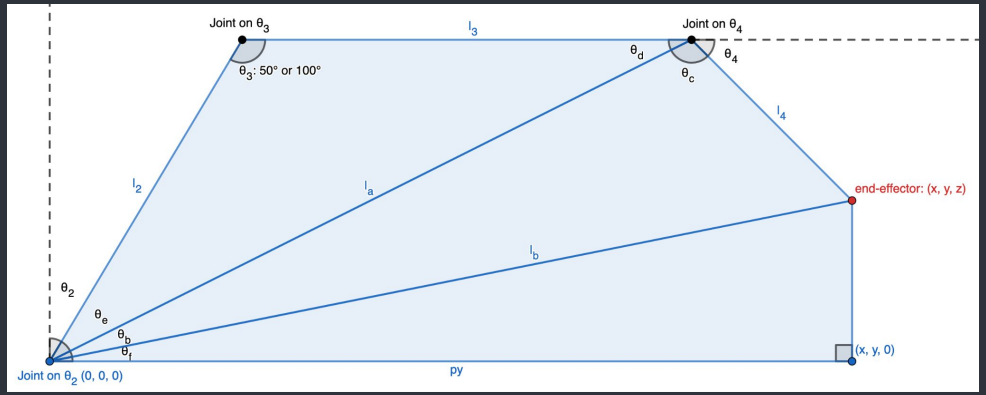


$$\theta_c = acos((l_a^2 + l_b^2 - l_c^2)/(2l_a l_b))$$

$$\theta_d = acos((l_a^2 + l_3^2 - l_2^2)/(2l_a l_3))$$

$$\theta_4 = 180 - (\theta_c + \theta_d)$$

Inverse Kinematics: Cosine Rule



$$\theta_e = acos((l_2^2 + l_a^2 - l_3^2)/(2l_2l_a))$$

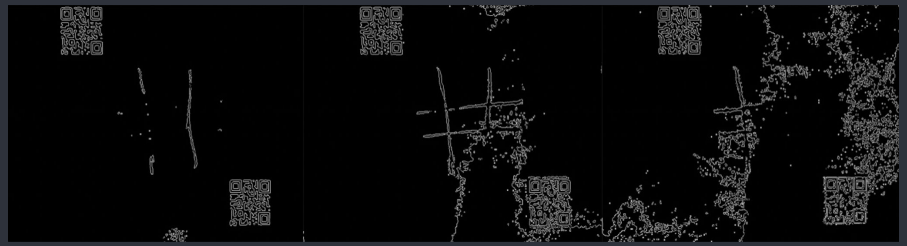
$$\theta_b = acos((l_a^2 + l_c^2 - l_4^2)/(2l_al_c))$$

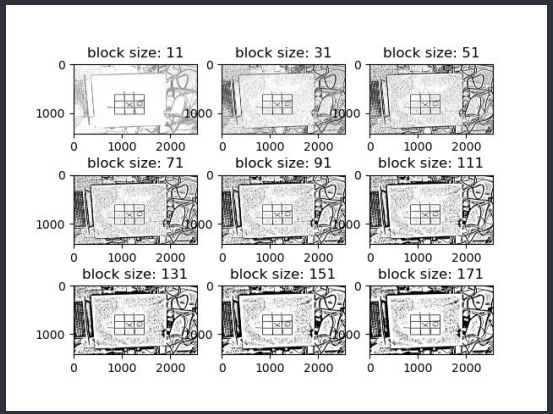
$$\theta_f = atan(z/py)$$

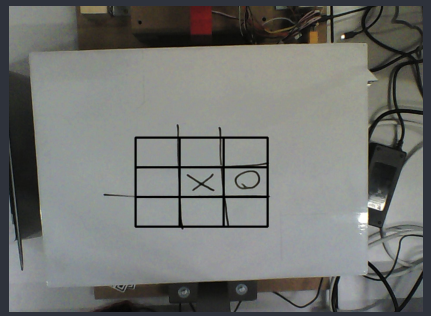
$$\theta_2 = 90 - (\theta_e + \theta_b + \theta_f)$$

Image processing

- increase in robustness
- simple threshold
- adaptive threshold







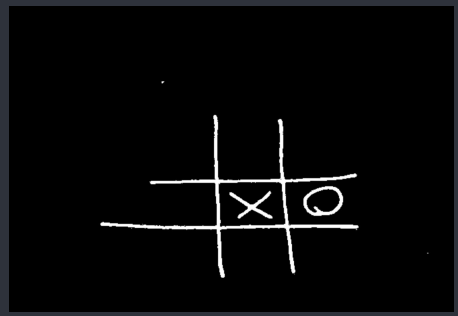
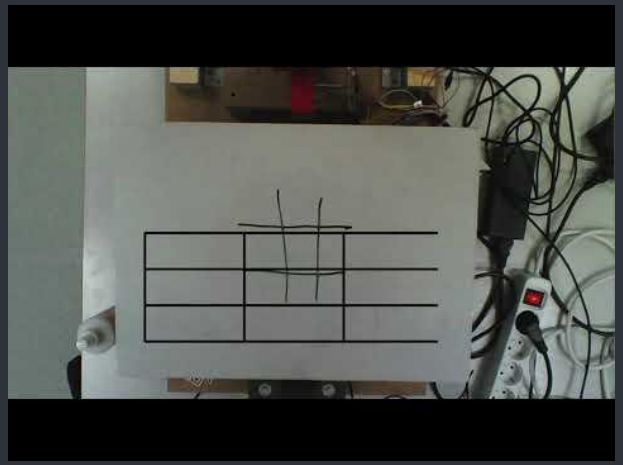
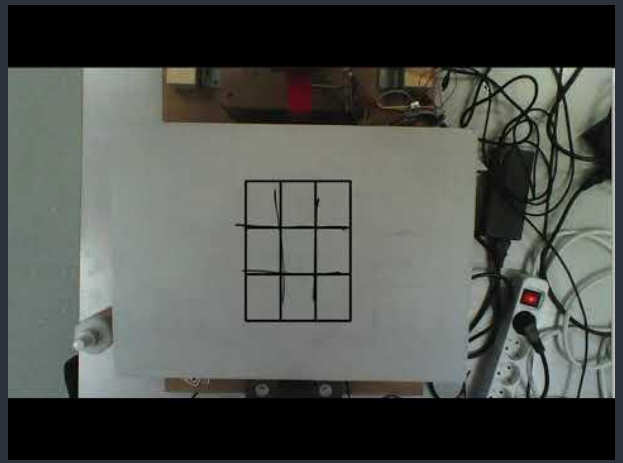
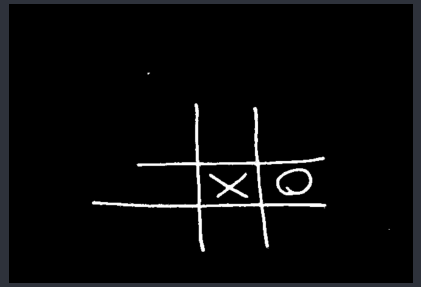


Image processing





Analyzing Game-state



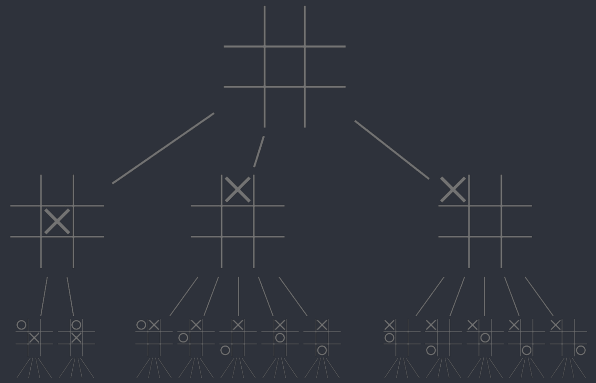
- isolate each cell
- CNN
  - the feature extraction front-end
  - the classifier back-end
  - VGG blocks

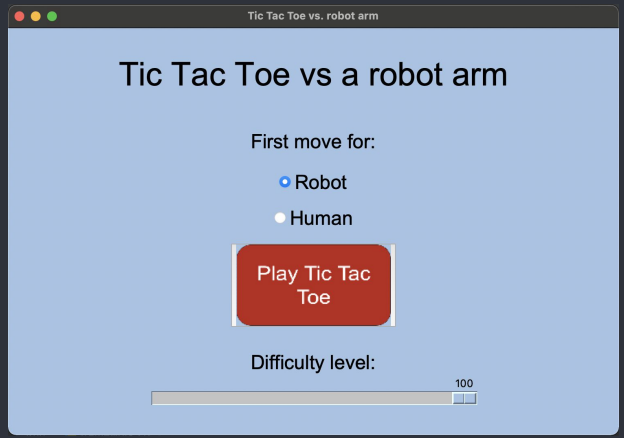
Motion Detection

- Grayscale conversion and noise removal
- Background subtraction
  - single background
  - weighted sum background
- Threshold application
- Blob detection

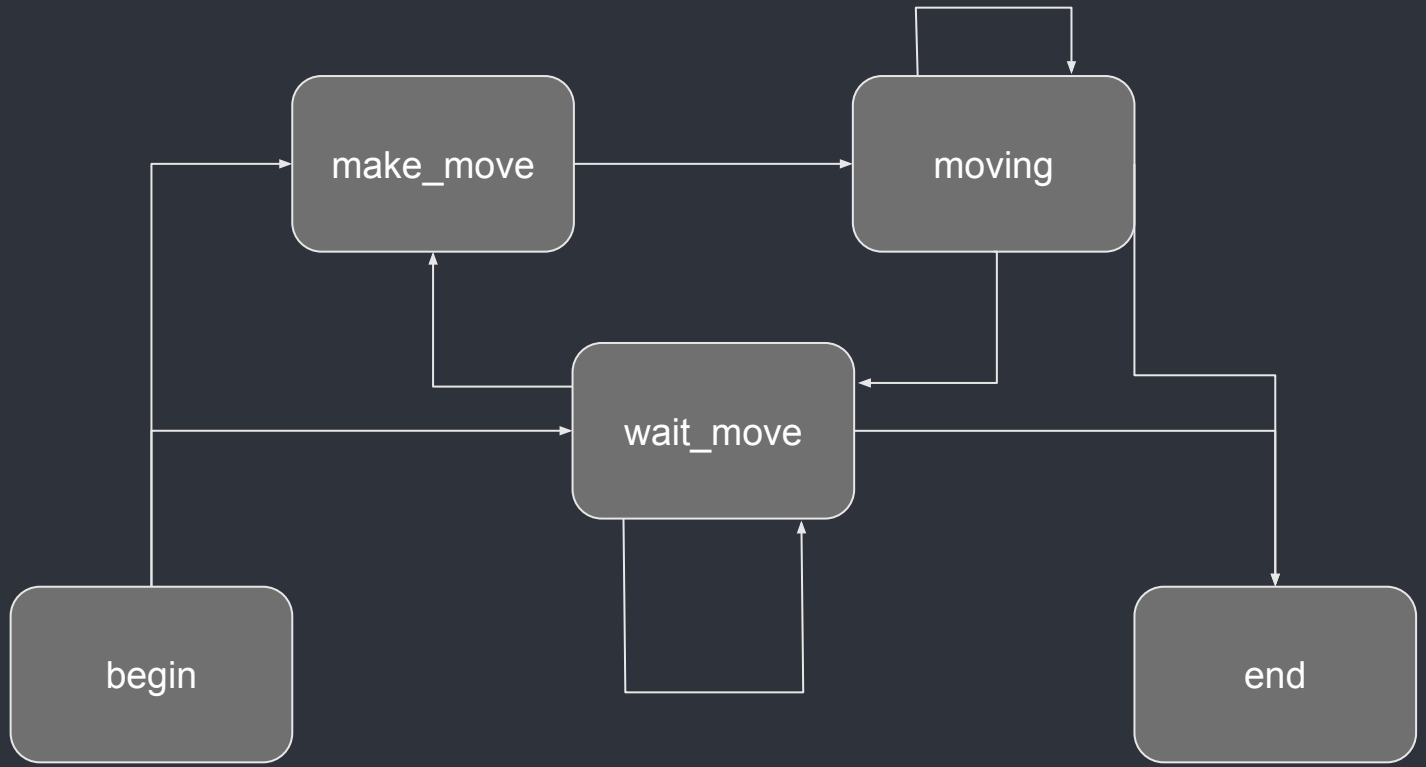
Minimax algorithm

- best move
- level of difficulty slider





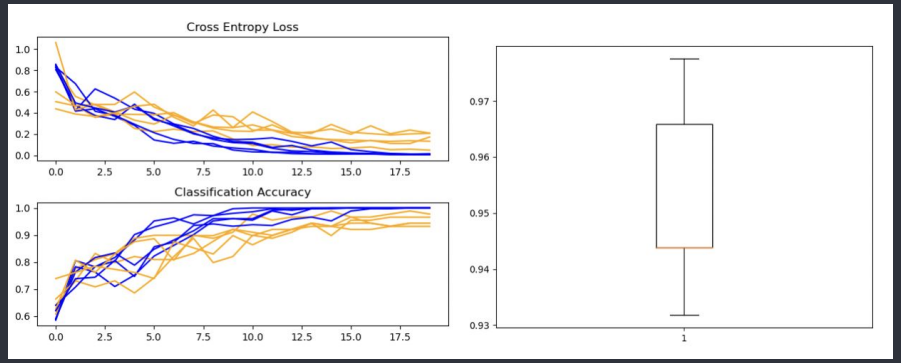
- State machine
  - begin
  - end
  - make_move
  - moving
  - wait_move

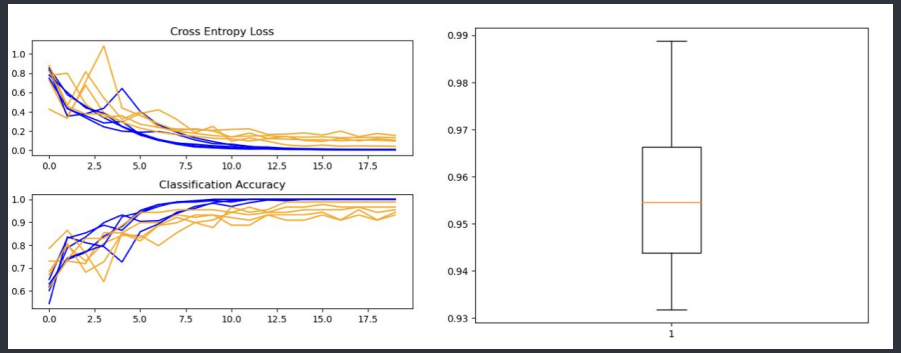


RQ1: How accurate is the deep neural net described at categorizing cells compared to the base model CNN ?

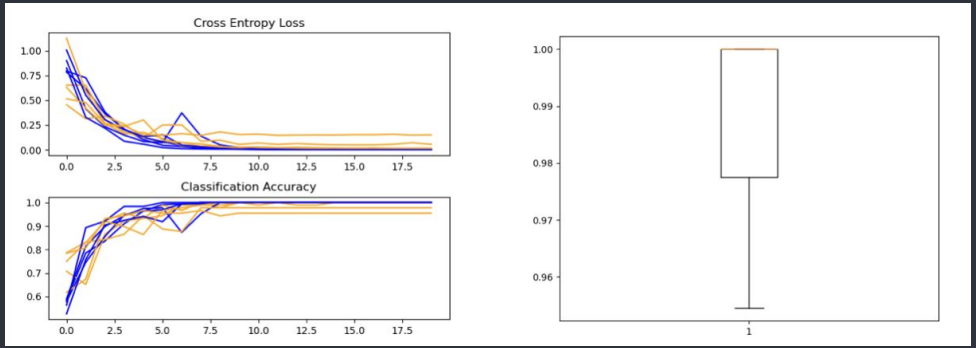
Experiment 1: Determine how accurate the deep neural net is by measuring the level of distortion and size of crosses and circles that are detected.

- Experiments
  - Research questions
  - Experiments
  - Results
  - Discussion



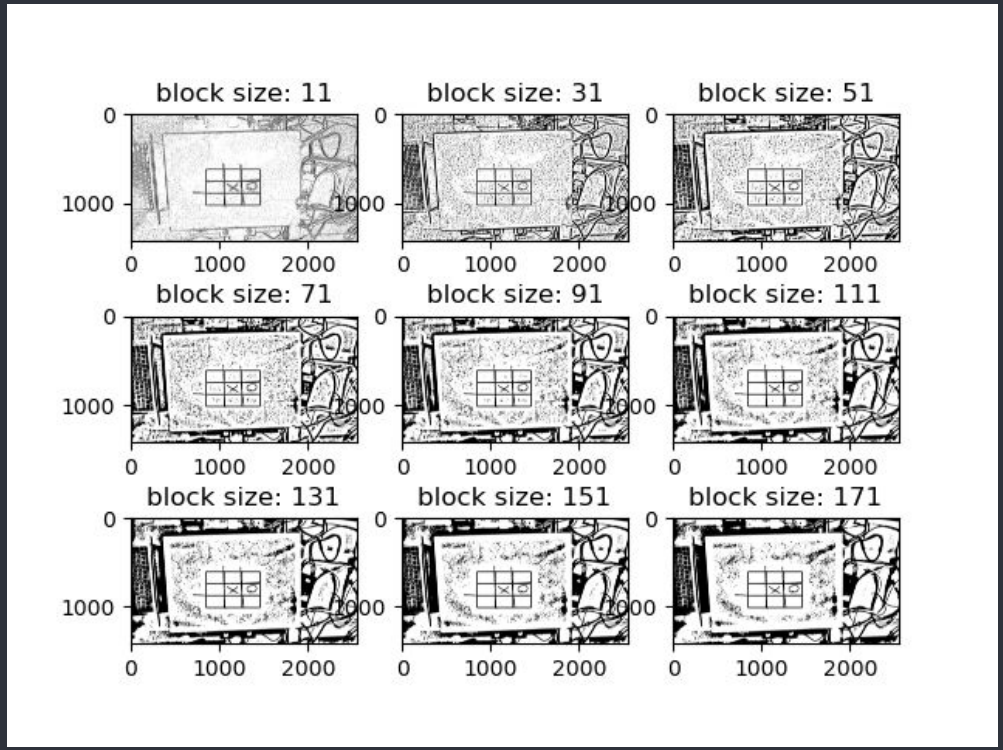


- Experiments
  - Research questions
  - Experiments
  - Results
  - Discussion

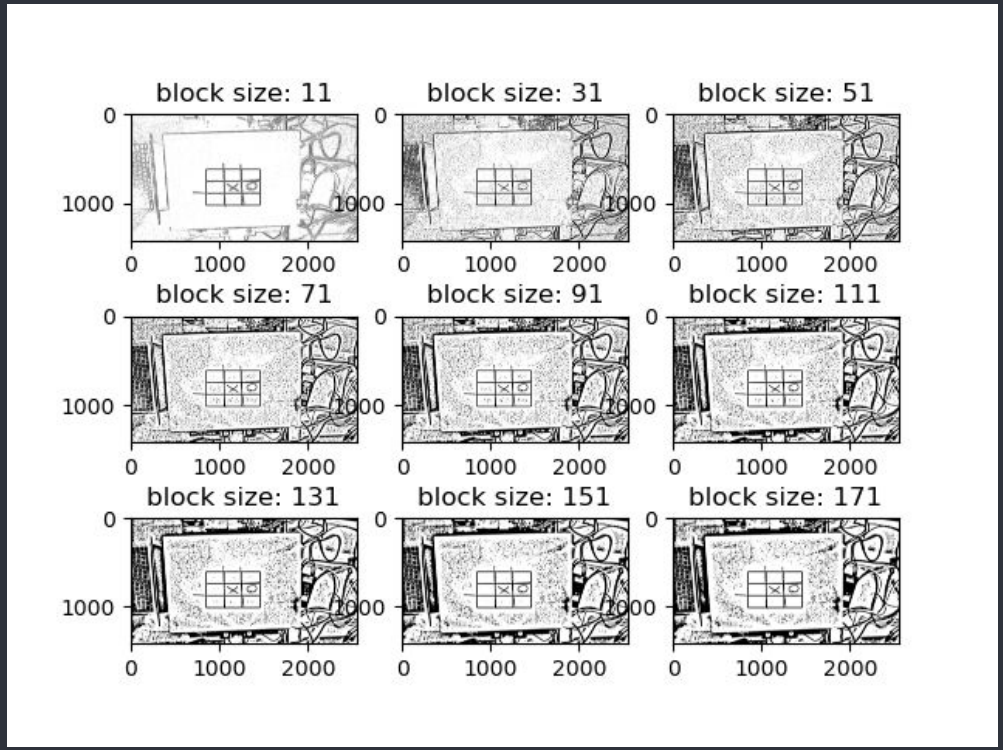


RQ2: For the computer vision, what is the optimal method for thresholding with regards to adaptive method and blocksize ?

Experiment 2: Determine the optimal parameters used in the computer vision by changing the blocksize and adaptive method and see how it performs.



Mean Threshold Test

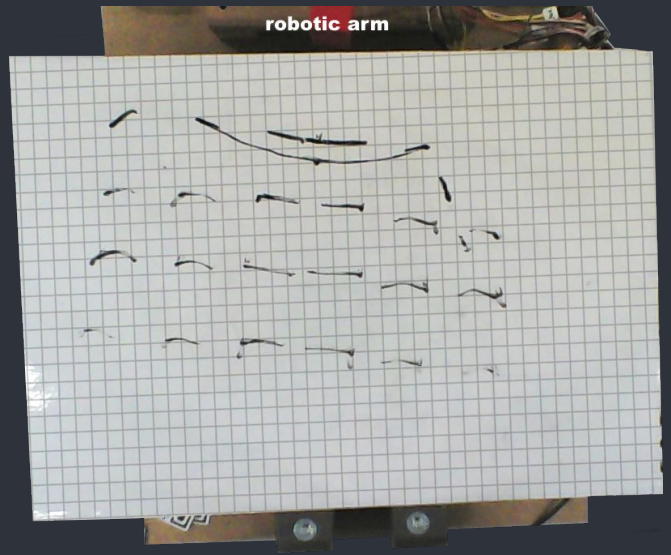


Gaussian Threshold Test

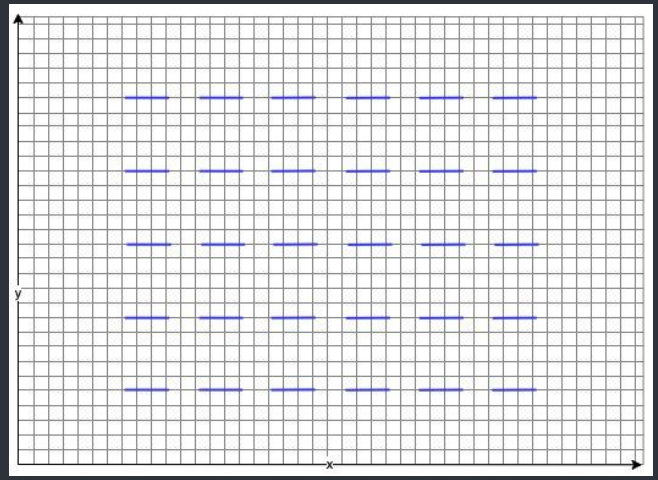
RQ3: How robust is the robotic arm at drawing lines ?

Experiment 3: Determine how robust the robotic arm is by making it draw several lines and see how it performs.

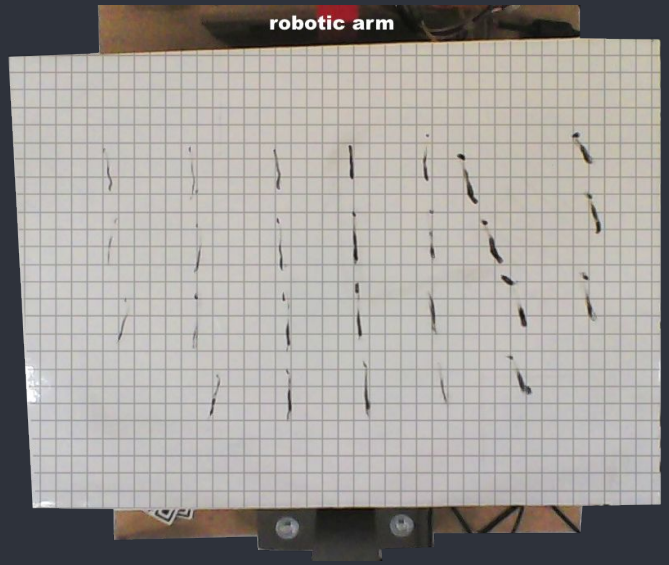
reality



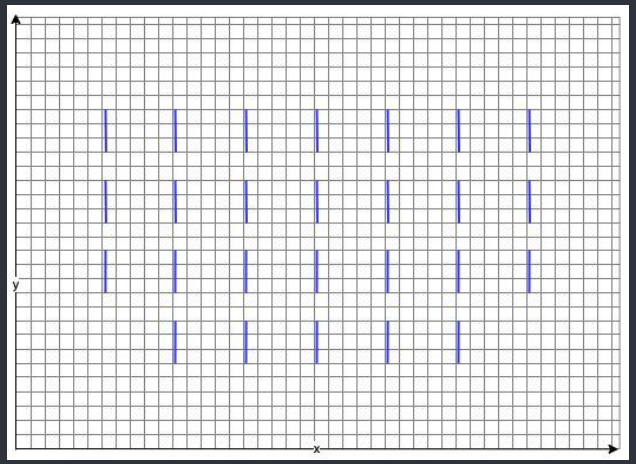
ideal case



reality



ideal case



Conclusion

- sketch + and ⬜
- robust computer vision: strong neural network, optimal method for thresholding

Further work

- robotic arm more accurate at drawing + and ⬜
- robotic arm able to sketch X and O
- object detection
- testing motion detection
- play an entire game (connecting everything together)