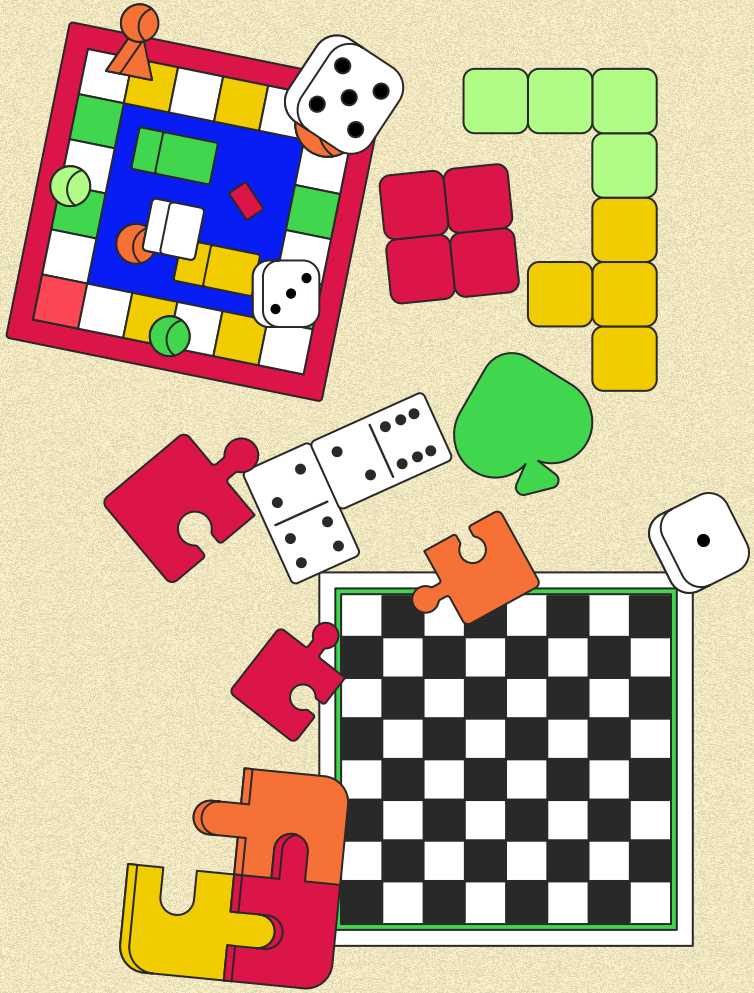
Boundary Analysis of Outcome Segments in Subtraction Games in more than one dimension

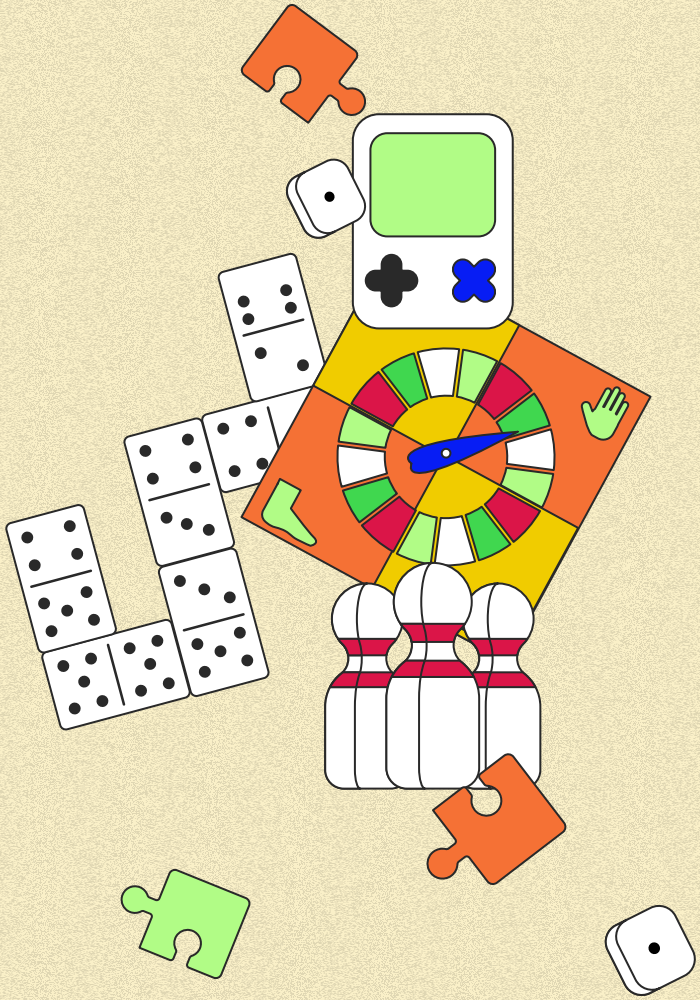


DDP Phase 1 Presentation

Guide: Prof. Urban Larsson
Tirthankar Adhikari (190070003)

“**In order to improve your game**, you must study the endgame before everything else, for whereas the endings can be studied and mastered by themselves, the middle game and the opening must be studied in relation to the endgame.”

– José Raúl Capablanca



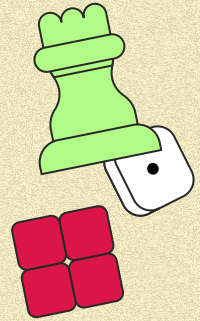
What's a Subtraction Game?

A subtraction game is a simple combinatorial game where, given a heap of n beads, 2 players A and B take turns alternatively, picking a and b beads from the heap alternatively where a and b belong to some subset of the Natural nos., the player to make the final move wins.

Subtraction games in more than one dimension

- $n = (n_1, n_2, \ldots, n_m)$ for some $m > 1$
- $A = \{a_1, a_2, \ldots.\}$
- $B = \{b_1, b_2, \ldots.\}$

  Here a_i and b_i are m-dimensional vectors and possibly infinite in no.



What are Game Positions?

A Subtraction Game has 4 different positions: N, P, L and R

N position

- The first player has a winning strategy

P position

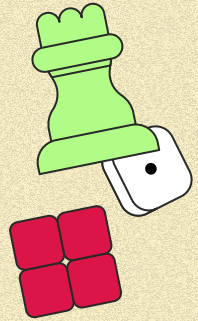
- No matter what the first player does, the second player always has a winning strategy

L position

- No matter whether the Left player goes first or second, it has a winning strategy

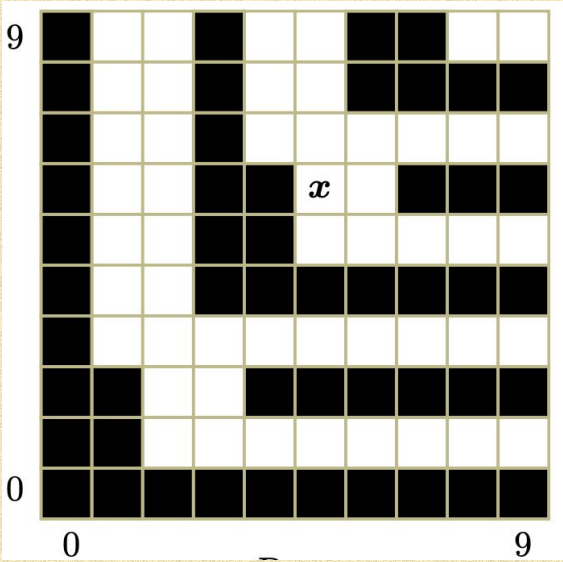
R position

- No matter whether the Right player goes first or second, it has a winning strategy

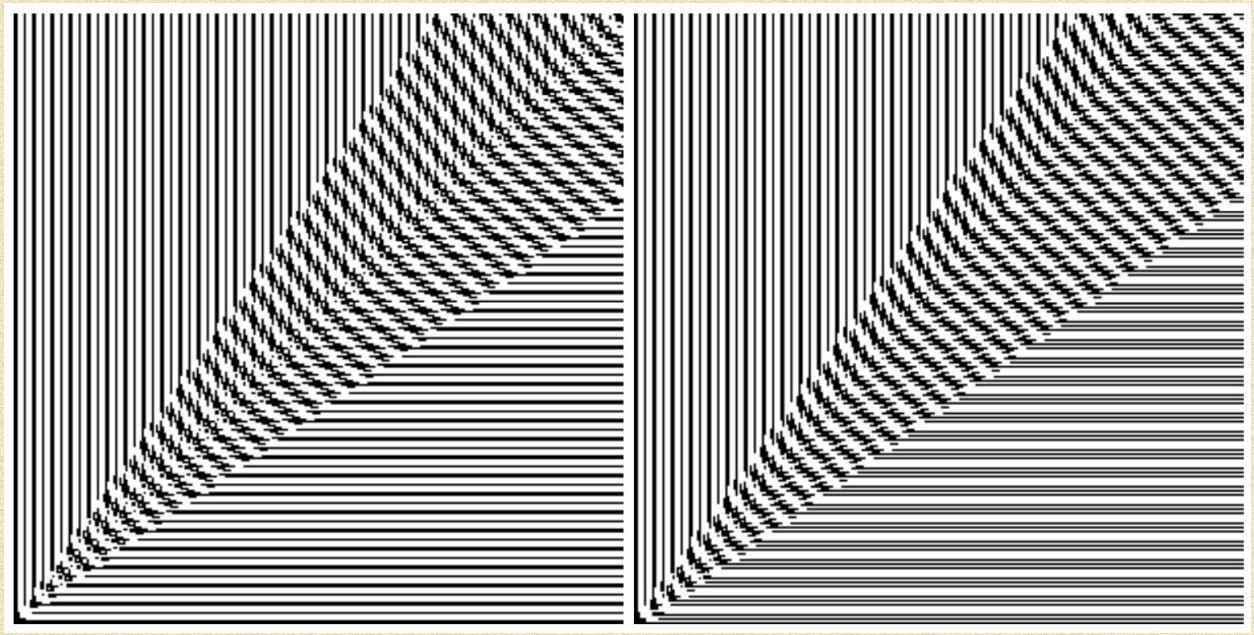


Hedgehog & Squirrels

Consider the following game: The picture illustrates the initial outcomes of the ruleset hedgehog&squirrel, i.e. S = {(2, 1),(1, 3)}. Here x = (5, 6) represents the given starting position in the second paragraph. The black cells correspond to the losing positions for the current player, a.k.a. the P-positions.



As we go on plotting more points?

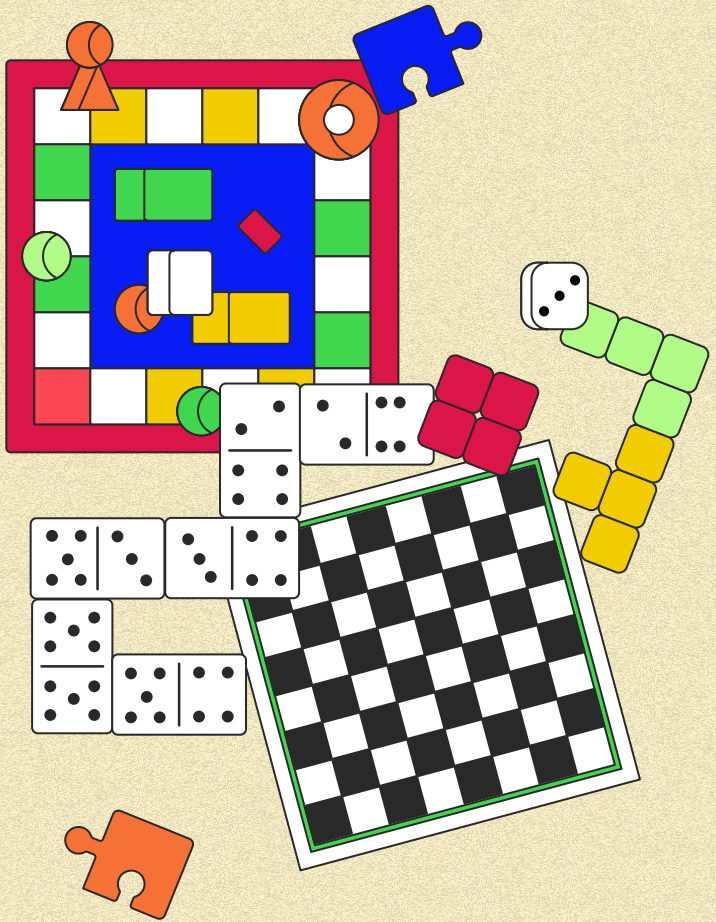


A symmetric and an asymmetric 4-segmentation respectively: the initial 300 by 300 outcomes of three move S = {(2, 6),(6, 2),(3, 3)} (left) and S = {(2, 6),(3, 3),(6, 1)} (right).

01

Need to analyse the “boundaries” of these outcome segments

But how do we do it?



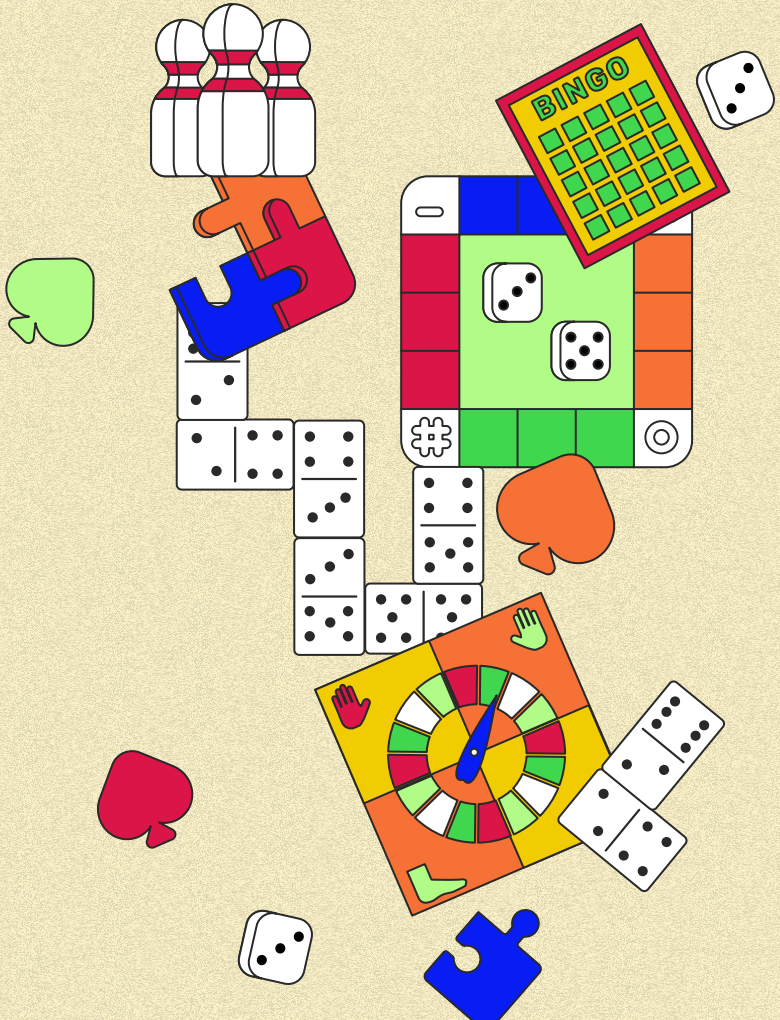
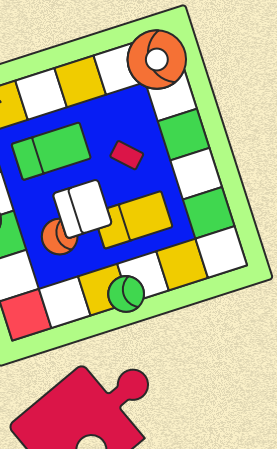


Image Processing!

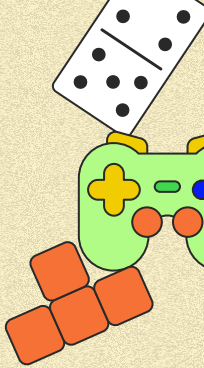
By Thresholding these position plots based on the “texture” of P and N positions!

Texture Segmentation

Texture analysis refers to the characterization of regions in an image by their texture content. Texture analysis attempts to quantify intuitive qualities described by terms such as rough, smooth, silky, or bumpy as a function of the spatial variation in pixel intensities. In this sense, the roughness or bumpiness refers to variations in the intensity values, or gray levels. Texture analysis is used in various applications, including remote sensing, automated inspection, and medical image processing. Texture analysis can be used to find the texture boundaries!



Current situation & problems

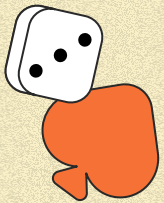


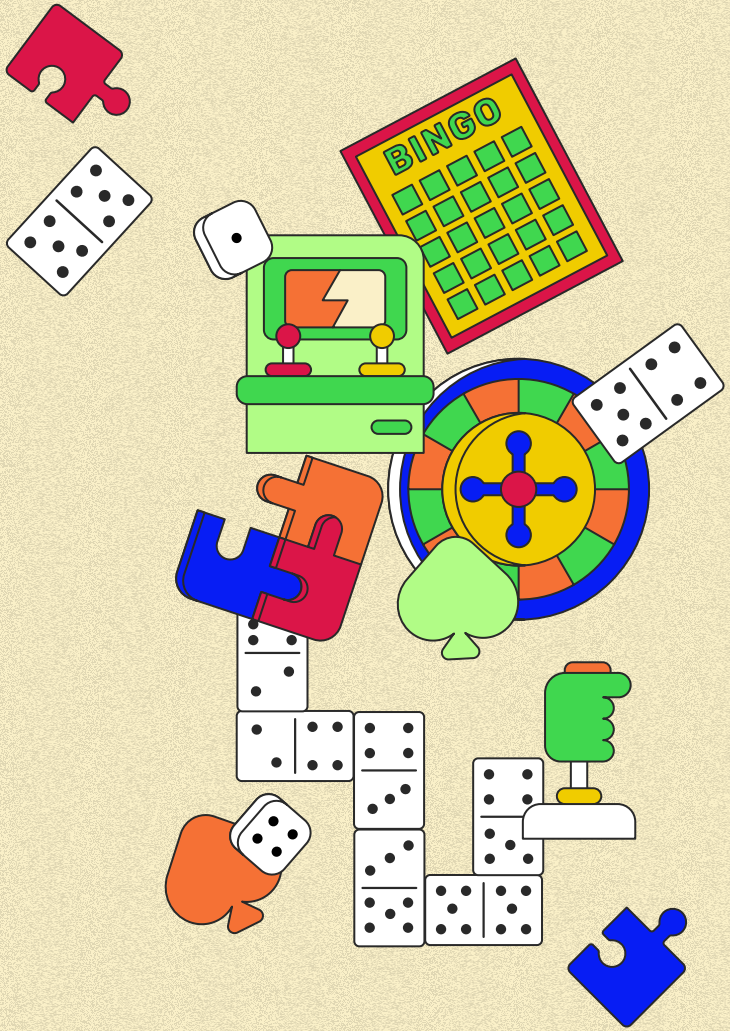
Current situation

Given several graphs obtained by plotting the N and P positions (Black and White dots) of different Combinatorial games, it is observed that the area under the graph is divided into several distinct regions based on the Texture formed by the dots and the task is to separate out these regions and comment on the slopes of the boundaries formed between them.

Problems

Automating this process ain’t that easy!





Process:

1) Variance Filter

2) Entropy Filter

3) Hough Transform

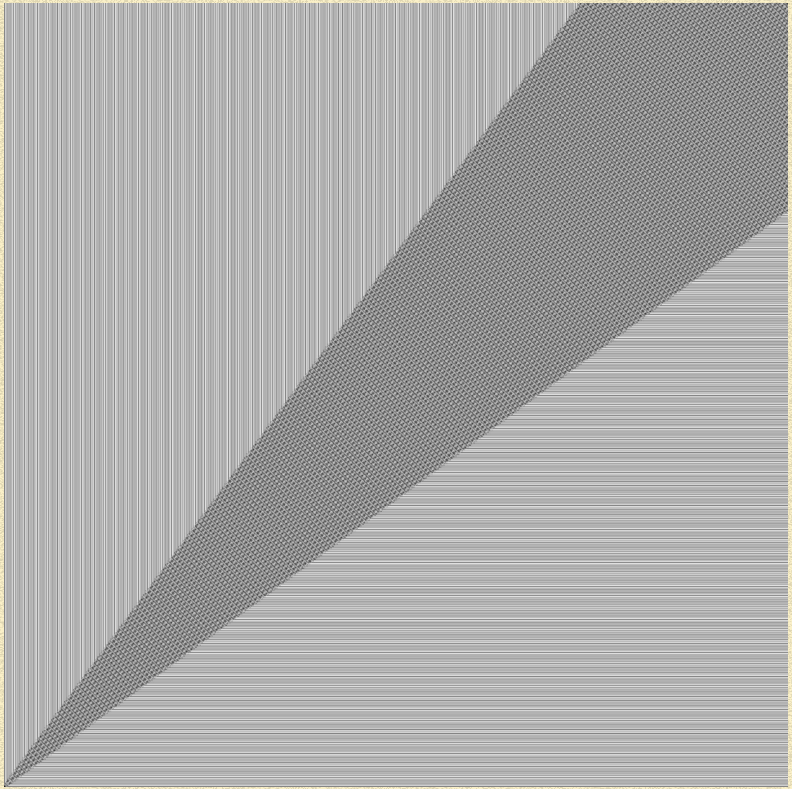
Variance Filter!

Highlights edges in the image by replacing each pixel with the neighborhood variance.

```
def var(img, n, l):
    for i in range(n):
        img_mean = ndimage.uniform_filter(img, (l, l))
        img_sqr_mean = ndimage.uniform_filter(img**2, (l, l))
        img_var = img_sqr_mean - img_mean**2
        img = img_var
    plt.imshow(img, cmap='gray')
    ch2pil(img)
    return img
```

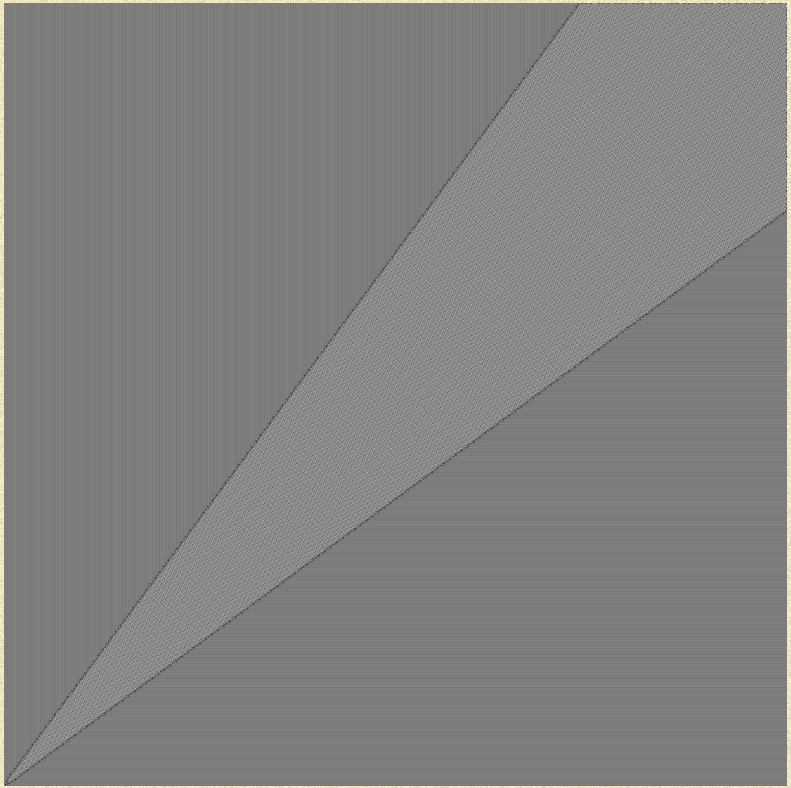
Variance Filter!

For the following Input image:



Variance Filter!

We get the following output:



Entropy Filter!

Entropy is a statistical measure of randomness that can be used to characterize the texture of the input image. This filter can segment portions of an image by their complexity.

```
def entro(img_var, k):
    for i in range(k):
        entropy_img = entropy(img_var, disk(5))
        entropy_img = entropy_img.astype('uint8') * 255
        img_var = entropy_img
    plt.imshow(img_var)
    ch2pil(img_var)
    return img_var
```

Entropy Filter!

After applying the Entropy filter:

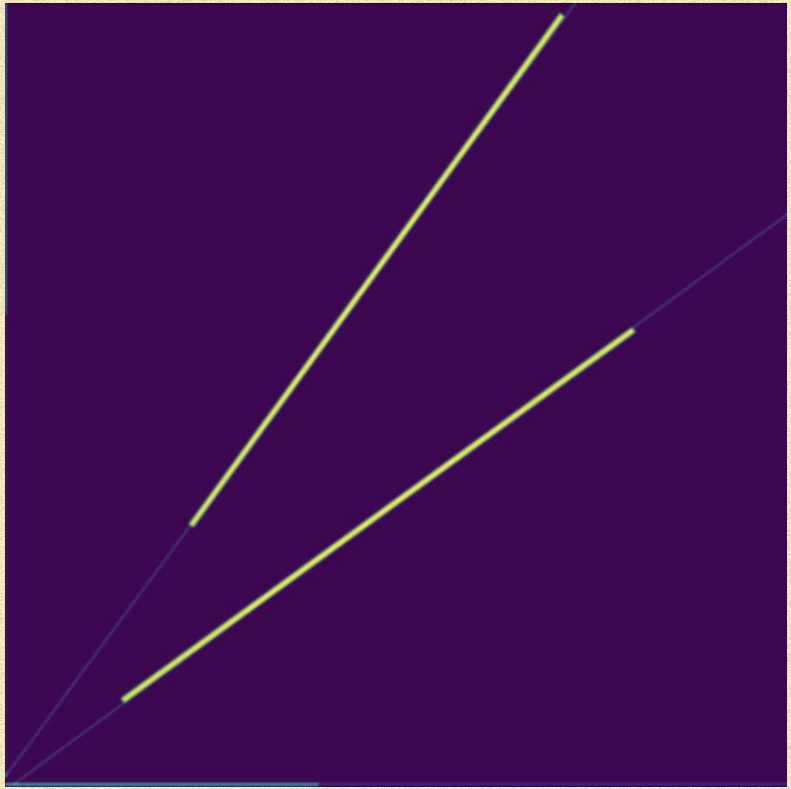
Houghline Detection!

A popular technique for detecting straight lines and curves on gray- scale images. It maps image data from image space to a parameter space, where curve detection becomes peak detection problem.

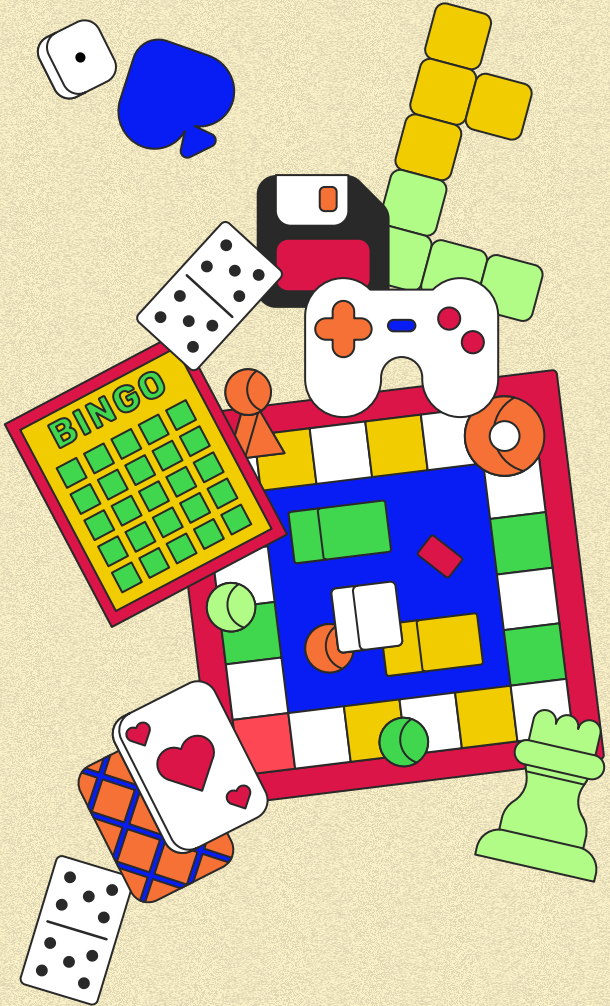
```
def houglines(entropy_img):
    image = entropy_img

    edge_image = cv.GaussianBlur(image, (3, 3), 1)
    edge_image = cv.Canny(edge_image, 100, 200)
    edge_image = cv.dilate(
            edge_image,
            cv.getStructuringElement(cv.MORPH_RECT, (5, 5)),
            iterations=1
        )
    edge_image = cv.erode(
            edge_image,
            cv.getStructuringElement(cv.MORPH_RECT, (5, 5)),
            iterations=1
        )
    edges = cv.Canny(edge_image, 50, 150, apertureSize=3)
    lines = cv.HoughLines(edges, 1, np.pi/180, 200)
```

Houghline Detection!

After applying the houghline transform:




Boundary line finally detected!



Now what?

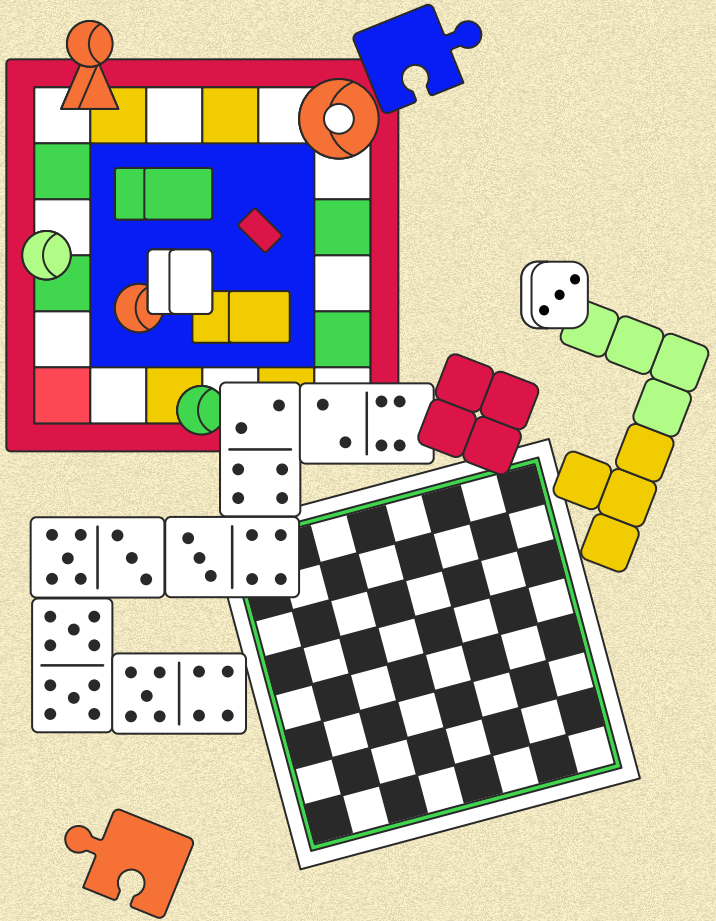
1) Once we get the boundary lines, we essentially partition the image into several different regions of similar behaviour

2) Among these, now we can do our analysis for the points away from the boundary lines and can possibly generalise the P and N positions for these different regions!

3) We successfully analyse and game as a Combinatorial Game Theorist… Mission Successful!

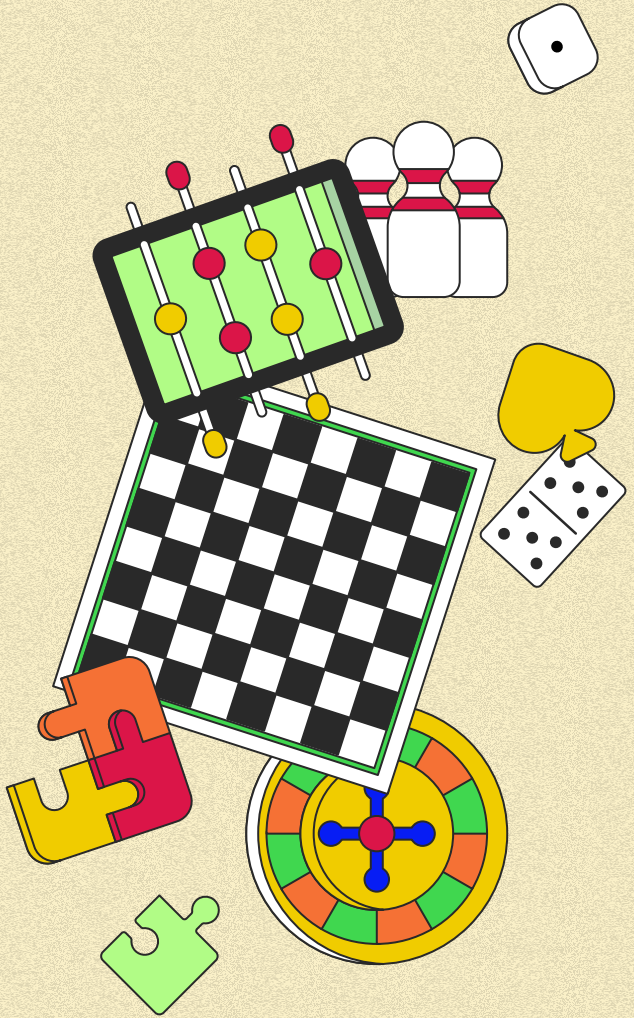
Future work:

Use Machine Learning to automatically detect the smallest repeating pattern and boundary analysis rather than manual Image Processing!



Literature review

- **U. Larsson. (2023)**. *SUBTRACTION GAMES IN MORE THAN ONE DIMENSION.*
- **Variance filter:** **https://scikit-learn.org/stable/modules/generated/sklearn.feature_selection.VarianceThreshold.html**
- **Entropy filter:** **https://scikit-image.org/docs/stable/auto_examples/filters/plot_entropy.html**



Thanks!

Do you have any questions?

190070003@iitb.ac.in
+91 9372094254

CREDITS: This presentation template was created by Slidesgo, and includes icons by Flaticon, and infographics & images by Freepik