Table Organizer

Final project by

Uri ziv, Michal Shechter, Nadav Sananes

Introduction

Organizing seating arrangements for a big family dinner, large business meeting or other occasions often becomes a time consuming and complicated task. Each guest has prefrences for who they would like to sit next to or to keep a distance from.
In this project we will write a program that allows users to upload a photo of a table with only plates on top. After the program analyzes the layout of the table, users can provide constrains, who would sit together or apart. The program will then generate a seating chart, a picture of the table with names arranged according to the program output. This picture can be shared with guests before they arrive.

Approach and Method

We choose to implement our idea using 2 main consepts from the course: edge detection and label relaxtion.
first using edge detection we will analyze the table's seat layout and after we will use label relaxtion to compete the best seating chart givien the constrains.

Plates recognition

In order to seat the guests around the table, we first need to understand the table's structure (how many seats are there, which seats are near each other, and which are across one another).

We created a dining plate detector to understand the table's structure. After receiving an image of the table, we apply Gaussian Blur and thresholding to remove anything that isn't the white plates.

After that, we use Canny's edge detector and a dilation function to emphasize the edges in the image. We use cv2.findContours() to get a list of all the contours in the processed edge map.

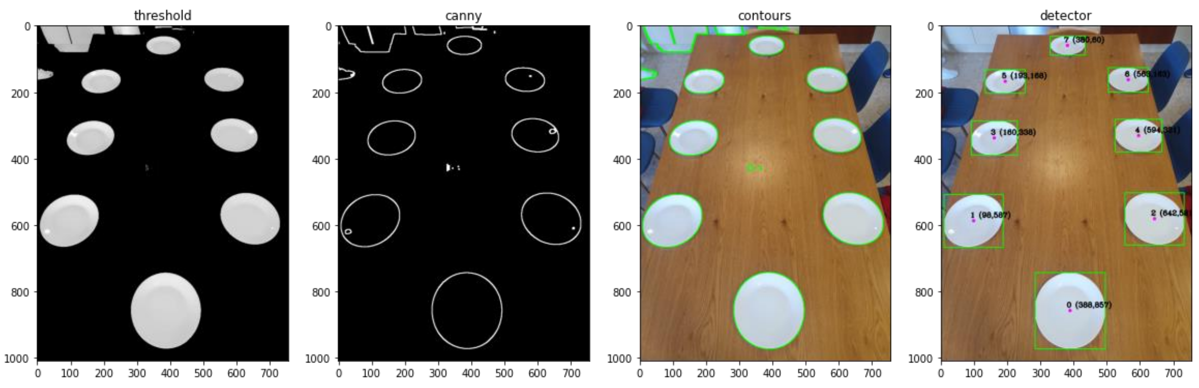
The contours are then filtered by area, circularity and the number of sides of their approximating polygon (min. 5). The detected contours are counted as plates, and a Spot object is created for each one.

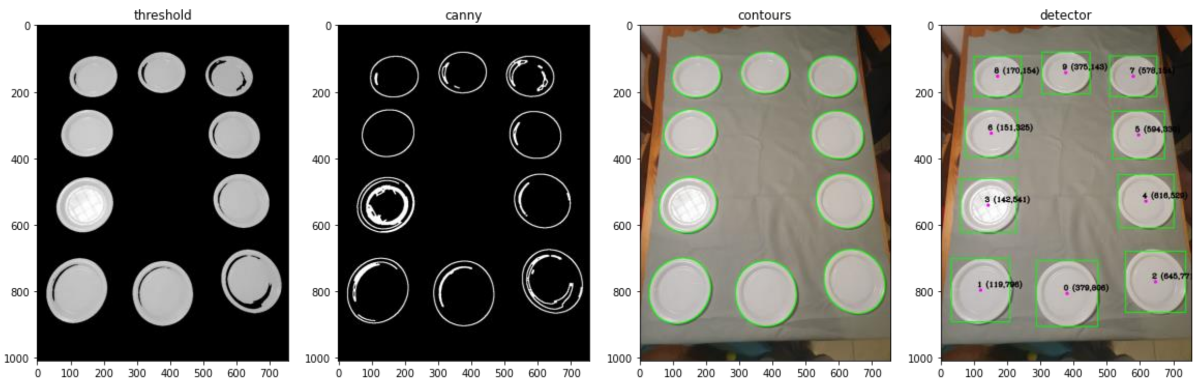
A Table object is constructed using the spots created at the previous stage. We calculate the relations between the spots and write the results in a relations weighted graph.

For each Spot, the top 2 nearest spots are considered as the spots right next to it at the table.

If 2 spots' x or y coordinates are within epsilon of each other, and the distance between them is close enough, we consider them to be across each other, and give them a smaller weight on the graph. That way, spots next to each other get more attention than spots across each other, but they are still taken into consideration.

Results





Other methods

We tried some other methods for detection, but they gave poor results:

- Using Hough transform took too long to compute and missed some far away plates that appear more elliptic than circular because of the angle of the shot.
- Using a generic AI object detector like YOLOv8 did manage to get bounding boxes around the plates, but it tagged them as other objects, or sometimes ignored the plates altogether and just detected the entire table as one object. We could have improved this method by creating a dataset of dining table pictures, but we did not have enough time.

Known issues

- The detector currently only supports vertical images of white plates, with a non-white table or map beneath them.
- Lighting conditions and reflections of light from the table may cause false positives, as we are looking for white circular contours.

reciving constarins & Label relaxation

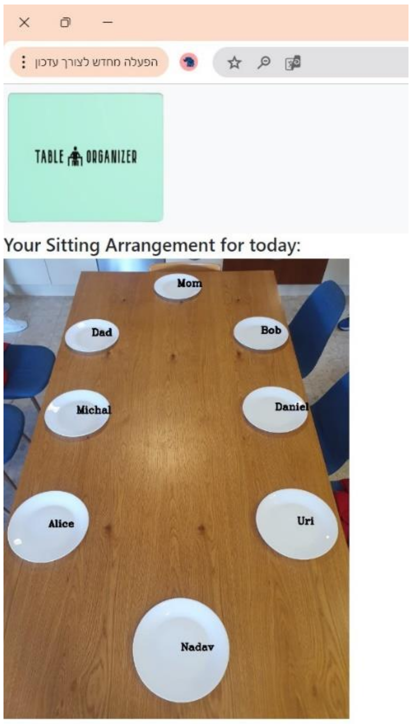
In order to apply label relaxtion in our project we had to establish four key components:
1. labels: The labels in our program represent the guests, identified by their index numbers.
2. objects: The objects are the spots around the table where guests will sit.
3. Compatibility function: We designed compatibility matrix, 'personMatrix', of size n*n where n is the number of spots around the table.
Each cell (i,j) in the matrix represent the relationship between person i and person j.
if the two people want to sit next to each other 'personMatrix[i][j] = personMatrix[j][i] = 1'. if they prefer to sit apart 'personMatrix[i][j] = personMatrix[j][i] = 0'. in cases where their prefernence is neutral or undefined (i=j) we assign 0.5.
We also created a second matrix, 'prMatrix', to represent the initial probability that a particular guest will

sit in a given spot.
This information, along with the connections between spots obtained from the table structure, forms the basis of our compatibility function.
If the relationship between two people matches the relationship between the spots on the table, the absolute value of their difference tends to be close to zero. This occurs when, for example, two people who want to sit next to each other are assigned to spots that are physically adjacent. Conversely, if the people's preference does not align with the physical arrangement of the spots, the absolute value tends to be closer to one.
By leveraging this observation, we defined an "R value" for every possible combination of two people and two spots. This value represents the level of alignment between the people’s preferences and the physical configuration of the table. If the R value is low, it indicates a good match between the people’s preference and the spots’ proximity. If the R value is high, it suggests a mismatch.

With this concept of the R value, we could calculate a "support" metric, which serves as a measure of how well the proposed seating arrangement aligns with the participants' preferences.
4. inital probabilities: When creating a guest list for a dinner party, people often write the names in a somewhat ordered fashion, which can serve as a basis for initial probabilities.
The first spot on the table has the highest probability of hosting the first person entered into the program, and the rest are proportioned so that their total probabilities sum to 1.
Continuing this pattern for all the spots, where for spot i the highest probability of hosting the i person entered into the program.
Additionally, we had to modify the label relaxation process to ensure each spot is assigned a unique label. In traditional label relaxation, multiple objects can share the same label, but in our context, each label represents a unique guest, and each spot can hold only one guest.
To address this, we introduced two safeguards. Firstly, in the compatibility function, we ensure that the compatibility of to spots getting the same label is zero.
Secondly, after the relaxation process, we perform a final check to confirm that each label is unique to prevent any overlapping assignments.

Results

After running the program the user will recive a photo of his table he provided in the beggining and on top the names in the order the program found to be the optimal sitting order. if the user entered less names then spots we identified the program will mention this is an empty sit.



Your Sitting Arrangement for today:

Conclusions

In many instances, we successfully identified the plates and provided the user with an organized table layout. However, we encountered some challenges in the following scenarios:

- When the plates were darker, or the table/background was lighter, this contrast mismatch made it difficult to identify the plates accurately.
- Uneven lighting also created issues, as some areas were overexposed while others were underexposed, leading to inconsistent detection.

To address these challenges, we plan to improve our program's recognition accuracy by:

- Enhancing our image processing algorithms to better handle variations in lighting and contrast.
- Expanding our recognition capabilities to support a wider range of plate colors, ensuring greater flexibility in diverse settings.

By implementing these enhancements, we aim to increase the robustness of our program, making it more reliable in various conditions and improving the user experience.

References

Cvzone - Money/Coin Counter using Computer Vision - https://youtu.be/-iN7NDbDz3Q?si=TlaNaGBcWH1cW4t0