Computer Vision

Project 1 - Mathable game reader

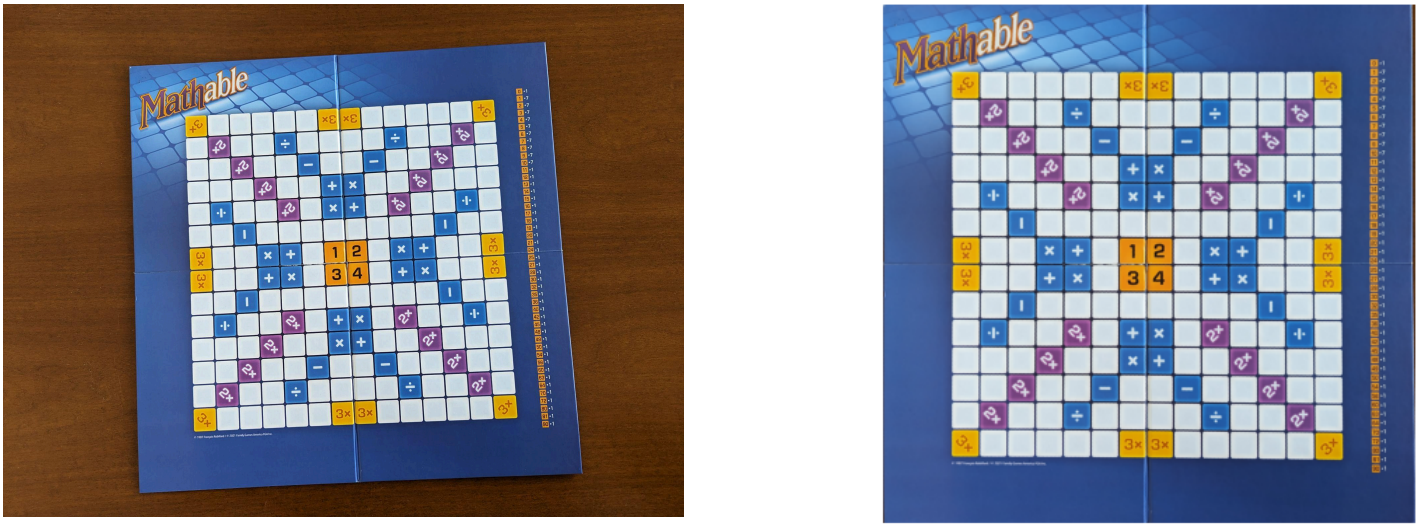
Alex-Mihai Serafim, group 407

I. Main idea and initial board processing

The main idea behind my implementation is to warp the perspective of each test image to a base image. To this warped test image, we can superimpose a grid which will more or less correspond to the actual tiles of the game board. After this, we can extract each tile and go on with the number detection from here.

In order to obtain the base image which will be used as reference, I have taken the first board in the *board+tokens* folder and applied the following transformations:

- Rotated and cropped the image in order to contain only the board
- Found the points corresponding to the corners of the board, calculated a perspective transform matrix and warped the perspective to the points corresponding to the corners of the image, so that the corners of the board now corresponded to the corners of the image

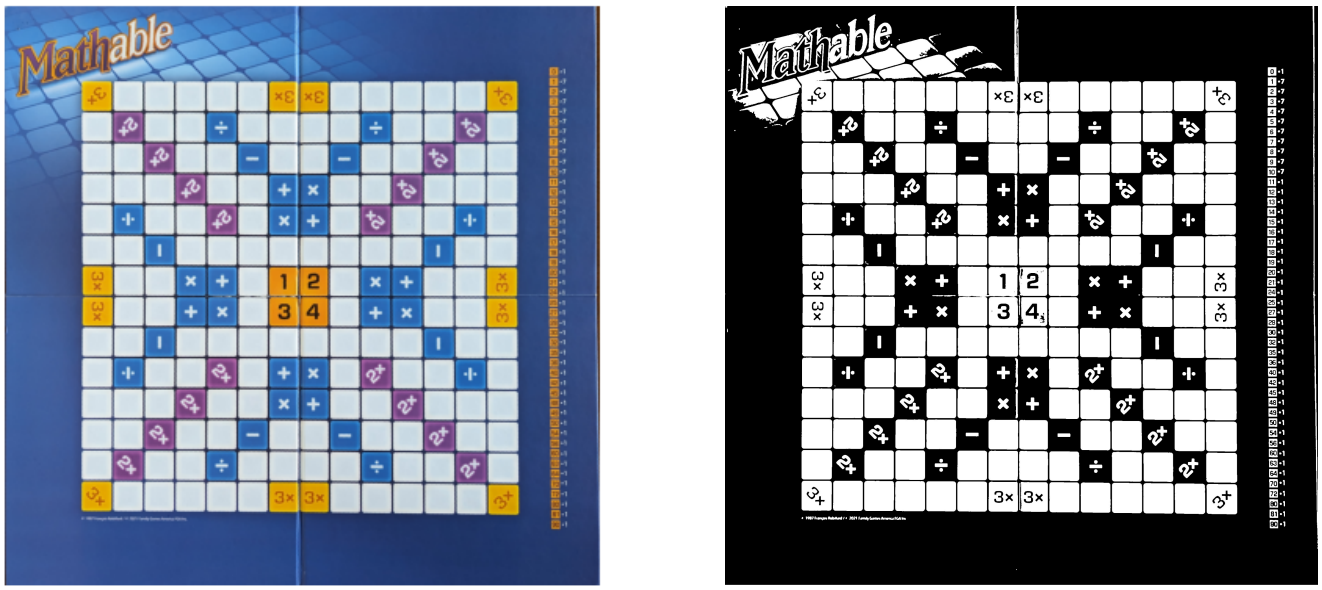


Base image before and after processing

II. Image processing

Image processing consisted of two steps:

1. Warping test images to base image
    - First, I computed keypoints and descriptors using BRISK
    - Then keypoints were matched between each test and base image using a kNNMatcher from FlannBasedMatcher
    - From the best matching keypoints, a homography matrix was computed
    - Using this homography matrix, a perspective warp was performed to bring the test image to a base image view (top-down)
2. Image transformation in order to aid number detection
    - Test images were grayscaled
    - A Gaussian blur was applied
    - Thresholding (Binary + Otsu)
    - Dilation in order to accentuate features

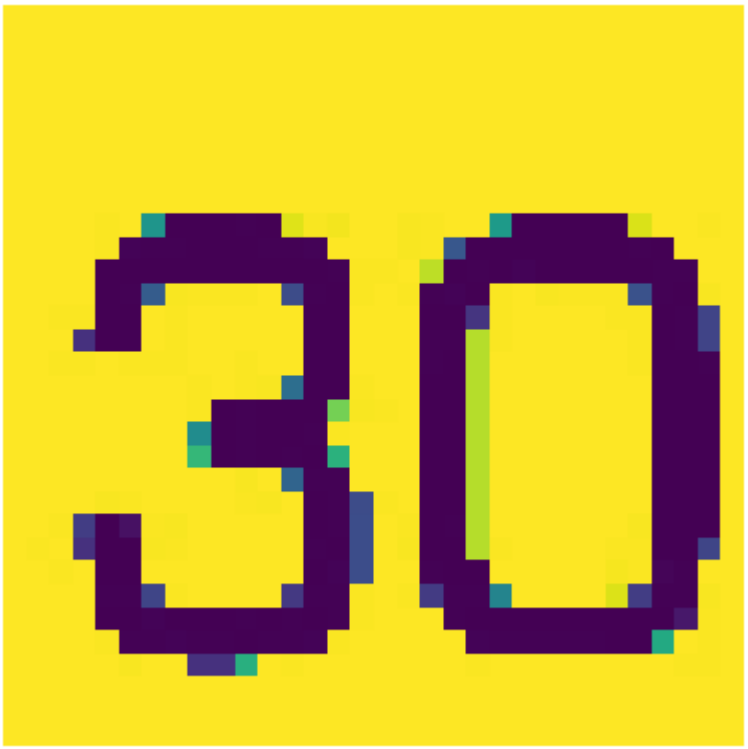


Example of image grayscaling+GaussianBlur+Thresholding applied after perspective warp

III. Model training

To obtain the training data for the model, I have used the tokens available in the example *board+tokens* directory, plus the tokens in the *train* directory games. This amounted to 722 tokens, 20% of which were left out for the validation set.

Following the processing steps presented above, the 196 tiles were extracted from the images using the superimposed grid, each of them being resized to 32x32 pixels.



Example of image fed into the neural network. Actual images in grayscale.

I have trained a CNN model for number detection, with the following architecture:

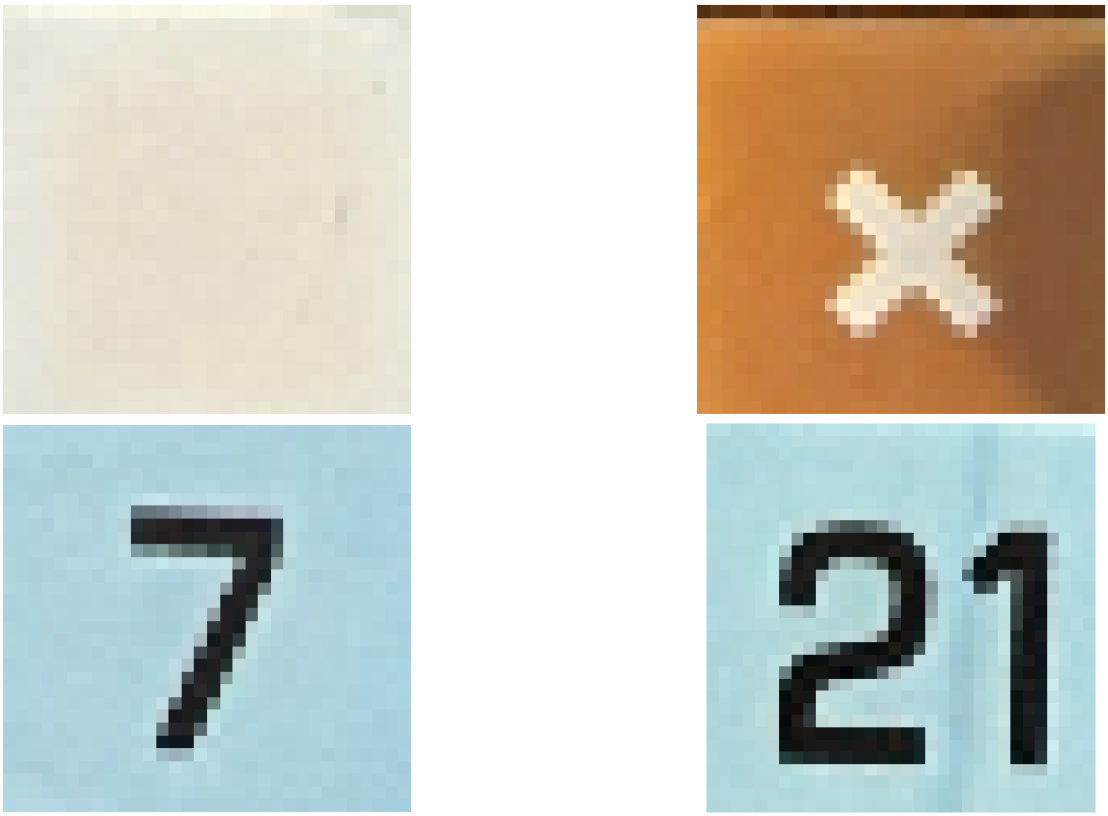
```
      self.conv1 = nn.Conv2d(in_channels = 1, out_channels = 32,
kernel_size = 3, stride = 1)
      self.conv2 = nn.Conv2d(in_channels = 32, out_channels = 64,
kernel_size = 3, stride = 1)
      self.conv3 = nn.Conv2d(in_channels = 64, out_channels = 64,
kernel_size = 3, stride = 1)
      self.fc1 = nn.Linear(in_features = 2 * 2 * 64, out_features = 64)
      self.fc2 = nn.Linear(in_features = 64, out_features = 46)
      self.activation_fn = nn.ReLU()
      self.pool = nn.MaxPool2d(kernel_size=2, stride=2)
```

This neural network was trained for 40 epochs and obtained an accuracy of 89.5% on the validation set. The output layer of the network has 46 neurons, for each of the 46 possible numbers.

IV. New tile detection

In order to detect which position a tile has been placed on, I have used histogram comparison. More specifically, for every two images I computed their histograms and then calculated their correlation using *cv2.HISTCMP_CORREL*.

Each tile from an image at index *y_x* was compared with its corresponding tile from the image at index *y_x-1*. The tile for which we have minimum correlation with its corresponding tile from the previous image is the tile which was newly placed. With this new tile detected, we can go ahead and feed it into the neural network and calculate the score.



Tiles vs. their correspondent from previous images

V. Score keeping

With the new tile identified and the number on it predicted, all we have to do is to write this new value into the game-keeping matrix and calculate the score: check how many equations this new tile completes and whether it is on a multiplier spot or not.

VI. Conclusion

My solution can be considered successful at detecting the tiles, numbers and keeping the score, with only 3 number detections failing in the fake test provided. The histogram comparison works really well, with the only bottleneck of this solution being the accuracy of the neural network. With more hyperparameter fine-tuning it would probably achieve a better accuracy; further improvements can also be found in changing the descriptor algorithm (perhaps with SIFT) or the matching algorithm (perhaps with BF instead of FLANN).