Pic-a-pix Solver

By Maayan Zehavi

Introduction

What is a pic-a-pix puzzle?

Pic-a-pix is a paint-by-number logic puzzle, in which cells in a grid must be colored or left blank according to numbers at the side of the grid in order to reveal a hidden picture.

Motivation

Solving logic puzzles is enjoyable most of the time, but once confronting an error or a dead-end situation, one might get frustrated. A little help in the shape of a hint, a glance at the erroneous marks or even just a validation of current progress may give one just the nudge he needs to happily accomplish the puzzle and reveal the desired hidden figure in the puzzle.

Project Goal

The project goal is to provide several options of solution presentation and validation of a partially-solved puzzle.

Course of action

General Steps

Image straightening, grid detection, grid completion, 'solve' and 'attempt' matrixes extraction, and presentation of the result according to the chosen revealing option.

Method:

Given: a partly-solved puzzle image, a solution image, number of rows and number of columns.

- Image straightening – done for both puzzle and solution image
  - Calculating Hough transform
  - Calculating a score for grid fitting (namely horizontal and vertical lines) for specific degrees

        - Iteration I – degree ranges between -90:90 with jumps of 5, tolerance is 3 degrees
            - Yields a rough estimation of rotation degree, denoted 'd1'
        - Iteration II – degree ranges between d1-5:d1+5, tolerance is 1 degree
            - Yields best rotation degree, denoted 'd2'
    - Rotation of the Image by d2 degrees
    - Background crop
- Grid fitting – done for both puzzle and solution image
    - Basic synchronization elements identifying:
        - Horizontal and vertical lines via Hough transform
        - Known outlines and bounds
    - Grid completion using:
        - A Histogram of adjacent lines distances
        - Known numbers of rows and columns
        - Other calculations based on the synchronization elements
- 'attempt' matrix extraction
    - Estimating whether each square in the puzzle's grid is marked or not
- 'solution' matrix extraction
    - Estimating whether each square in the solution's grid is marked or not
- Solution presentation
    - Detecting errors and unmarked squares
    - Plotting marks on original image according to the option chosen

- Off the record – an OCR attempt
    - Resizing the digits' templates to estimated size according to the grid
    - Defining search areas and the corresponding numbers' rows or columns
    - Calculating Hough Transform of flipped templates (for each digit)
    - Pick the best fit in each slot

Results

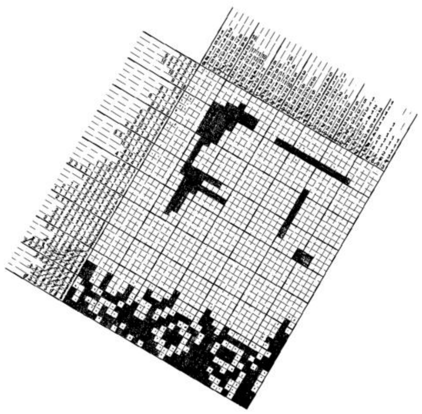
Input:

Dimension parameters:

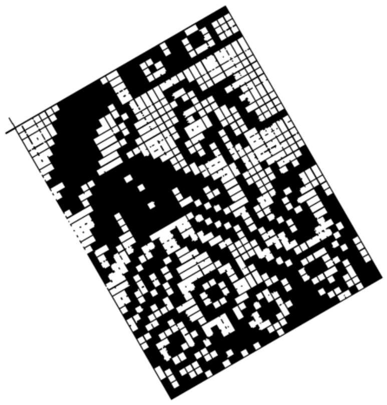
N = 35

M = 45

Puzzle image:

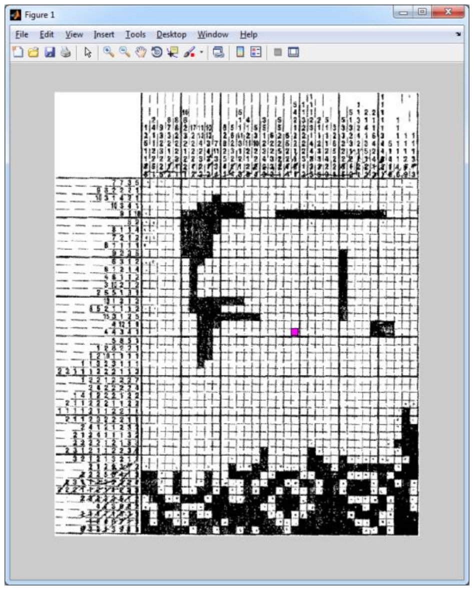


Solution image:

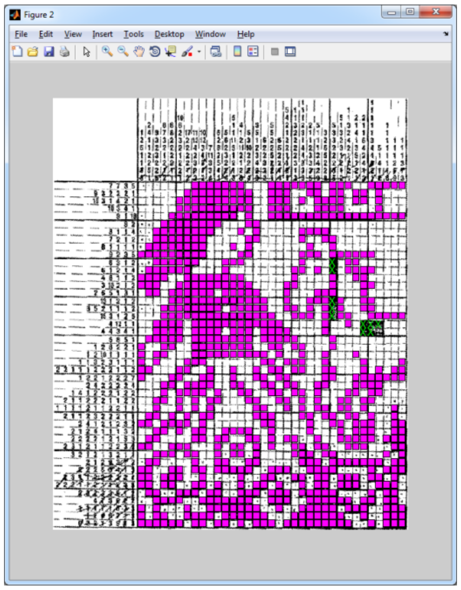


Output:

Hint :



Show Solution:



Conclusion

The project turned out to be both rewarding and disappointing. Even after giving up on OCRing, the options offered by the code were very helpful to me.

Also, with further improvements and fine-tuning, hopefully an autonomous solver is completely applicable.