Use Of Jigsaw Puzzle Solving Algorithms In The Real World

Luca Sartore

May 2023

Contents

1 Abstract

The jigsaw puzzle problem has been in the eye of computer scientists for a while, and some clever solutions have already been found. These algorithms are made to work with a "digital" jigsaw puzzle F1, but there aren't papers (at least not popular enough to be searchable) that try to apply the solution to a "real world" jigsaw puzzle F2.
The problem has been tackled by some small projects. But, as said earlier, the process and eventual challenges has never been documented by a full paper, this wants to be the first.
As a bonus the paper will also cover the creation of a user friendly app that will be open source and free to use.

2 Introduction

2.1 Classification

This paper will focus on type 2 puzzles. A type 2 puzzle is a puzzle where the position, and the orientation of each piece is unknown.

2.2 Digital vs Real-World Jigsaw Puzzles

There is another important distinction between different types of puzzles. They can be divided into "digital" and "real world" jigsaw puzzles.

Figure F1: An example of a "digital" jigsaw puzzle

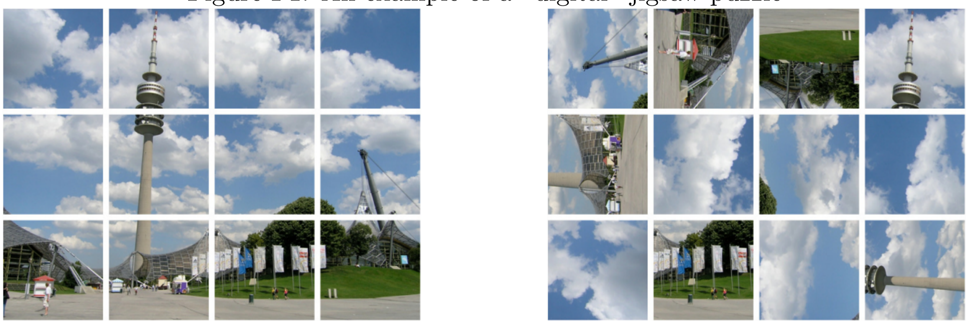
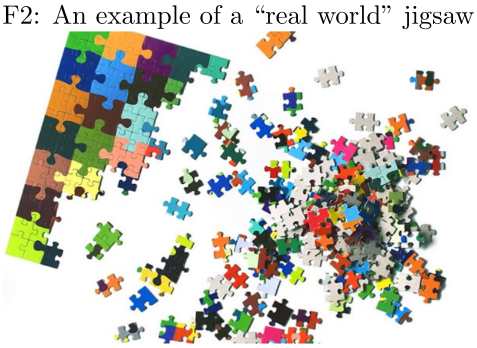


Figure F2: An example of a "real world" jigsaw puzzle



The reason this distinction is important is because, despite the generic concept of the puzzle not changing, obtaining accurate matches of a piece's characteristics is far easier with a digital puzzle, since there are far less things that can go wrong.

Figure F3: An example of what can go wrong when dealing with the real world

3 Previous Literature

This section will analyze 3 different algorithms that have been proposed as a solution of type 2 puzzles. The objective is to understand the strengths and the weaknesses of each one, to build up some knowledge that will be useful for the next section.

3.1 General Structure Of The Algorithms

All the algorithms that will be analyzed are composed of 3 sub algorithms:

3.1.1 The Splitter:

This component takes as input one or more images containing all the pieces. It then split all the pieces from each other, 9and split each piece into his four sides.

3.1.2 The Comparator:

This component compares each side with all the others, in order to understand whether they match or not.
There are two distinct kinds of “Comparator” algorithms: The “Binary Comparison” and the “Non Binary Comparison”. As the name suggests when comparing two sides with a “Binary Comparison” the result can either be 0 (they do not match) or 1 (they match). In contrast a “Non Binary Comparison” can give any value between 0 and 1. This allows states of uncertainty to be represented.

3.1.3 The Solver:

This component uses the information provided by the Comparator 3.1.2, and tries to find a solution (i.e. a position and an orientation for each piece) that is the most likely to be correct.

3.2 Solving Jigsaw Puzzles By The Graph Connection Laplacian [1]

This algorithm falls in the “binary comparison” category 3.1.2. And can be used to understand the strength and weaknesses of this approach.
The main benefit of this approach is speed, having a binary comparison allows some specific optimization, in particular the use of some graph theory techniques.
The algorithm has a time complexity of ruffly $O(N^2)$. Which is in line with the theoretical minimum for jigsaw puzzle solving, according to the study: “No easy puzzles: Hardness results for jigsaw puzzles [3]”.
The negative aspect might be the accuracy, since by using only two states (match or not match) some informations are lost, compared to a non binary comparison. To quantify accuracy the paper used the “neighbors comparison” metric, which “calculates the percentage of pairs of image patches that are matched correctly”.
Is important to note that the discussed algorithm works with digital puzzles F1, and this gives it the

advantage of having more precise data.
Whether this algorithm could work with a real world puzzle F2 will be discussed in section DOTO INSERT RFERENCE.

3.3 A Genetic Algorithm-Based Solver for Very Large Jigsaw Puzzles [2]

This paper applies a genetic algorithm to the jigsaw puzzle problem, with some promising results.
The algorithm seems to have a $O(N^2)$ time complexity, with time slightly lower than the previous example (But it is impossible to know for sure which one is faster, given that they didn't specify the hardware used).
The paper has used the same method to evaluate accuracy that the previous one used (neighbors comparison) so it is easy to compare them.
The algorithm falls in the "Non binary comparison" 3.1.2; this should give it an advantage, since it has more data to work with.
Unfortunately this advantage does not compensate for the worst precision of the algorithm itself, and the accuracy results are equivalent, if not slightly worse than the previous algorithm.
It is important to keep in mind that for the very nature of genetic algorithms, it might be possible that the accuracy would have been better if they had allowed it to run for more generations.
Is also important to note that the discussed algorithm works with digital puzzles, and this gives it the advantage of having more precise data.
Whether this algorithm could work with a real world puzzle will be discussed in section DOTO INSERT RFERENCE.

References

[1] Vahan Huroyan, Gilad Lerman and Hau-Tieng Wu, Solving Jigsaw Puzzles By The Graph Connection Laplacian, 2020. https://arxiv.org/pdf/1811.03188.pdf.

[2] Dror Sholomon, Omid David and Nathan S. Netanyahu, A Genetic Algorithm-Based Solver for Very Large Jigsaw Puzzles, 2013. https://openaccess.thecvf.com/content_cvpr_2013/papers/Sholomon_A_Genetic_Algorithm-Based_2013_CVPR_paper.pdf.

[3] Michael Brand, No easy puzzles: Hardness results for jigsaw puzzles, 2015. https://www.sciencedirect.com/science/article/pii/S0304397515001607.