Reconstructing 2D Art With Genetic Algorithms and Deep Learning

Elizabeth Stevens
Division of Science and Mathematics
University of Minnesota, Morris
Morris, Minnesota, USA 56267
stev0531@morris.umn.edu

ABSTRACT

The topic of this paper is reconstructing 2D art, such as frescoes and mosaics, using genetic algorithms and deep learning. This paper describes research about two algorithms solving real-world problems of fragmented wall paintings and dismantled tile panels. One algorithm focuses on wall paintings, while the other focuses on tile panels. They were both effective and improve upon algorithms already developed in their problem space. The algorithm for tile panels also improves upon the expert reconstructions they were using for testing.

Keywords

Genetic Algorithms, Deep Learning, Archaeology, 2D Art Reconstruction

1. INTRODUCTION

Reconstructing broken objects is a hard, time consuming task and it can take many forms, such as fractured frescoes, broken pottery, shredded documents or photographs, and other archaeological artifacts. Like a jigsaw puzzle, the pieces of these objects need to be put together to reconstruct what has been damaged. There are many broken artifacts, such as the over 100,000 tile panels in the National Tile Museum in Lisbon, Portugal. It would take decades for the experts working there to reconstruct all of them, not including the additional tile panels being delivered there. As seen in Figure 1, an expert from the National Tile Museum is in the process of reconstructing a tile panel.

Here, I will be focusing on the real-world problems of reconstructing wall paintings and tile panels. In this problem space, wall paintings and tile panels have similar issues but also significant differences. They share the issues of missing pieces and degraded edges on pieces, making them further harder to fit together. The main difference between wall paintings and tile panels are the shapes the pieces take and the techniques we have to put them back together. For example, the pieces of wall paintings have various irregular shapes, and someone trying to reconstruct the painting can use the different shapes to help find which pieces go together. However, the pieces of tile panels all have a square shape, excluding degraded edges, and anyone reconstructing the tile panels would have to rely solely on the images on the pieces to know which are next to each other. Figure 2 shows two examples of what a tile panel might look like before and after they are reconstructed.

Figure 1: Expert reconstructing a tile panel at the National Tile Museum in Lisbon, Portugal. Taken from [5].

The two algorithms I will discuss are the Wall Painting Algorithm (WPA) developed by Sizikova and Funkhouser in [7] and the Tile Panel Algorithm (TPA) developed by Rika et al. in [5]. As their names suggest, the WPA reconstructs wall paintings while the TPA reconstructs tile panels. Both use genetic algorithms (GA), but the TPA also uses deep learning to aid in the reconstruction. In Section 3, I will go over the structure of the WPA and the results found from testing the algorithm. Afterwards in Section 4, I will go over the structure for the TPA and its testing results. I will conclude with further work to be done in this problem space.

2. BACKGROUND

In this section, I will give background on genetic algorithms and deep learning. Both the WPA and TPA use genetic algorithms, while the TPA incorporates deep learning.

2.1 Genetic Algorithms

Genetic Algorithms (GAs) are algorithms based on evolution via natural selection. GAs use a selection procedure to find candidates with the best traits from the population along with recombination, otherwise known as crossover, and mutation to produce new potential solutions [2]. For the Wall Painting Algorithm, a solution is an assemblage of a portion of a wall painting. For the Tile Panel Algorithm, a solution is a complete tile panel. Often, the can-

UMM CSci Senior Seminar Conference, November 2019 Morris, MN.

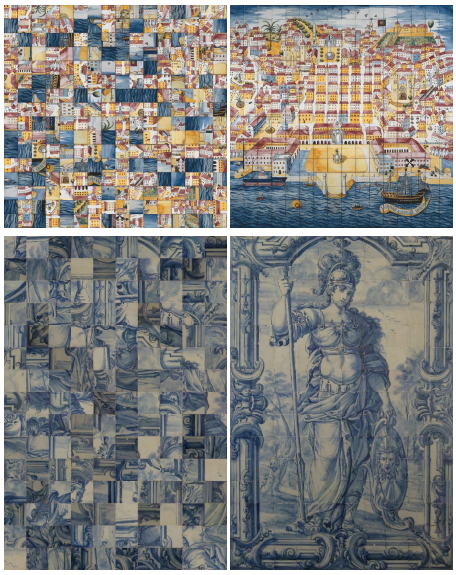

Figure 2: Example of what the Tile Panel Algorithm takes in, disassembled tiles on the left, and what it returns, assembled tile panel on the right. Taken from [5]

didates found from the selection process are called parents, and the solutions produced by crossover and mutation are called children. As Figure 3 illustrates, the basic GA structure initializes the population with solutions, often randomly generated, that then iteratively goes through the selection process and recombination process until an ending criteria is met.

The selection procedure uses a fitness function to evaluate which solutions from the population will go to the recombination step to create children. Fitness functions are specific to the problem an algorithm is trying to solve, and they contain the criteria of what a good solution is. An algorithm's fitness function assigns scores to children from the previous generation. The children with higher fitness scores are selected to be parents in the next generation. Then the selected parents go through the recombination process, which creates hopefully better solutions, i.e. children. In general, the recombination process takes some characteristics of the parent, and combines them in a different way. For example, in the Tile Panel Algorithm, if both parents have the same match between two tiles, then the child would have that match as well.

A mutation will introduce some randomness to a GA. Mutation isn't always used in GAs, but is generally used to create solutions not normally allowed in recombination. An example of mutation being implemented in a GA can be found in Section 4.

2.2 Deep Learning

Deep Learning is a type of machine learning that is based on neural networks, which are systems loosely imitating how human brains process information and recognize patterns. A deep neural network (DNN) is made up of multiple layers that each extract features from the input, progressively extracting more complex data and creating representations of increasingly abstract concepts [1]. Rika et al. use multiple

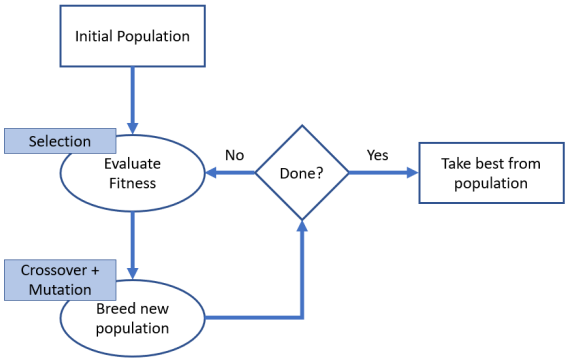

Figure 3: A simplified illustration of a genetic algorithm. In selection, the population has their fitness evaluated, and the ones with the highest score go to crossover to recombine into a new population. If the GA has mutation, it will occur during crossover. Then the GA checks if it's done, and either goes through the algorithm again, or takes the best from the population.

DNNs to analyze the color values of a given pair of tiles and to produce a value determining the compatibility between them. For an example of the basic idea, when comparing two tiles to see if they match, the DNN's first layer might extract only the red color values on the tiles. The second would then find lines in the colors, while the third layer finds more complex patterns on the tiles. Then, the fourth layer would compare the two tiles to see if they have matching patterns and return a corresponding number.

3. WALL PAINTING ALGORITHM

Sizikova and Funkhouser develop a GA, the Wall Painting Algorithm (WPA), in [7] to solve the real-world problem of reconstructing fractured wall paintings. The WPA focuses on the shape of the fragments when putting the painting together, ignoring the image. It takes in clusters, or collections of painting fragments with matches between them, and produces a solution, or a reconstruction of part of a wall painting. Something to note, is a single fragment can be part of multiple clusters.

3.1 Algorithm Structure

When the WPA initializes, it takes in singleton clusters, which are single fragments with no matches, and paired clusters, which are two fragments with a match between them. The WPA's selection process starts with ranking the clusters using the fitness function created in [7], which is explained below. After the clusters have been ranked, the next step is to filter them so that ones with a ratio of total fragments to unique fragments passed a threshold are kept. Sizikova and Funkhouser used 0.85 as the threshold. An example of a unique fragment is one found in only one cluster. The filtering is to encourage diversity in the clusters, so that the ones being passed along don't all have the same fragments in different configurations.

For the WPA, the fitness function ranks clusters by calculating $MaxST(C_i)$ and the number of fragments, $span_{f_i}$, or the number of matches, $span_{m_i}$, that are a part of the

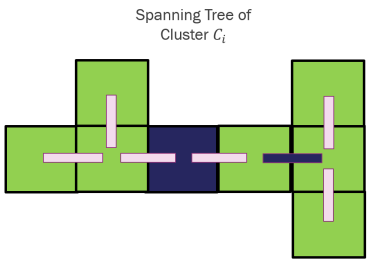

Figure 4: An example of a spanning tree for a cluster. The squares represent fragments and the lines between them represent matches. The dark blue square and dark blue line are examples of loose connections, because if they were removed the single cluster would become multiple clusters.

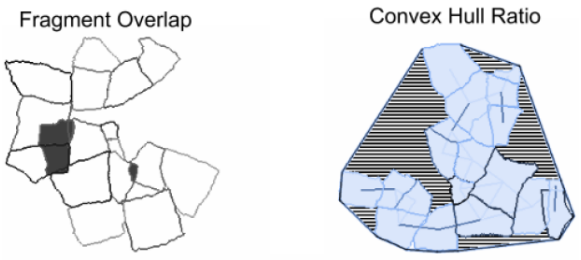

Figure 5: The shaded in fragments on the left show fragment overlap. The stripped shaded part on the right is the convex hull. These are examples of what the feasibility check looks for. Taken from [7].

spanning tree of cluster C_i (i.e. if a fragment or match is removed from a cluster, it will disconnect and become two separate clusters). $MaxST(C_i)$ is the sum of the match scores — numerical values attached to matches (described below) — of the maximal spanning tree of cluster C_i. Figure 4 shows an example of a spanning tree and highlights some loose connections within it. The goal of the fitness function is to maximize the match scores within the cluster, $MaxST(C_i)$, and minimize the number of loose connections, $span_{f_i}$ and $span_{m_i}$, so that the clusters are strongly connected and less likely to include an incorrect placement of a fragment. Sizikova and Funkhouser also included W as a weighting parameter to control the effect of the spanning fragments and matches in the clusters. The fitness function of the cluster C_i is

$$f(c_i) = MaxST(C_i) - W(span_{f_i} + span_{m_i}).$$

During the recombination process, new matches being created receive a match score to signify how strong it is. When combining cluster C_i and C_k, the match between them is scored by $C_{ik}(1+0.1M)$ where C_{ik} is the fitness score of the new combined cluster, and M is the number of additional matches being added to the this cluster. In other words, if a match was created between two clusters, each with many fragments, M would be the additional matches found between the two clusters as a result of the match receiving the score.

The WPA recombines through two different methods: by fragments or by match. When the parent clusters are recombined by fragment, the two clusters must have the same fragment within their clusters. The WPA considers all possible shared fragments within the clusters and the different ways they can connect them, choosing the child with the highest match score. When the clusters are recombined by match, they need to have a match between a fragment from one cluster and a fragment from the other. Since the number of potential children from this type of combination is vast, Sizikova and Funkhouser [7] use a weighted probability to choose which matches produced are considered. Specifically, when given a set of N spanning matches with match scores $f_1, f_2, ..., f_n$, match i will be selected with probability P,

$$P(i) = \frac{f_i}{\sum_{k=1}^{N} f_k}.$$

This tends towards considering matches with higher match scores to the population, but since it is a probability some lower match scores will be considered as well.

After a match has been created, but before it gets added to the population, the new cluster has to go though three feasibility checks to make sure the configuration of fragments is a feasible one. The first check makes sure that no fragments overlap each other. The second makes sure that clusters containing a certain number spanning fragments or matches passed a threshold are infeasible. The third checks that the cluster has a high fragment to convex hull ratio. A convex hull is the maximum area that the cluster will take up connecting the farthest points in the cluster with straight lines. An example of fragment overlap and a low fragment to convex hull ratio can be seen in Figure 5. After the cluster has passed the feasibility check, it gets added to the population and could become a parent when its generation goes through the selection process.

3.2 Wall Painting Algorithm Results

When testing this algorithm, Sizikova and Funkhouser used an artificial data set from a fresco that was created by others to specifically test algorithms for this problem space. The fresco was created, broken up, and weathered so people creating algorithms for this problem space could know the ground truth when testing their algorithm and could compare their algorithm against others. During testing, Sizikova and Funkhouser also made sure that the initial set of data was the same for all algorithms. They compared WPA against three others: two commonly used algorithms, dense cluster growth (DCG) and hierarchical clustering (HC), and the previous state of the art algorithm developed by Castañeda et al. [3]. The results from comparing DCG, HC, and WPA is described in Section 3.2.1 while the comparison results from Castañeda et al. and WPA is described in Section 3.2.2.

3.2.1 Comparison with DCG and HC

When evaluating DCG, HC, and WPA, Sizikova and Funkhouser compared results using the number of fragments in the final cluster and the F-score for that cluster, which is the average of precision and recall. In this case, precision is the proportion of correct matches within the cluster, while recall is the proportion of correct matches within the whole painting. The results are shown in Table 3 while the solutions for the three algorithms can be seen in Figure 6. As can be seen from Figure 6 and Table 1, more than double the fragments were found using WPA than the second biggest,

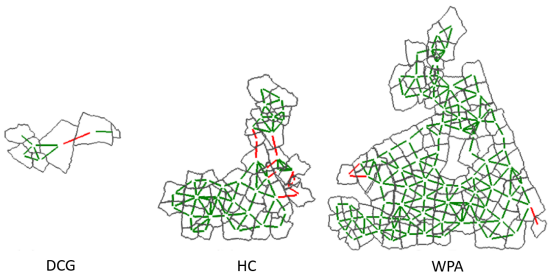

Figure 6: These are the solutions created when comparing from the Dense Cluster Growth (DCG), Hierarchical Clustering (HC), and the Wall Painting. Green lines signify correct matches, while red lines are incorrect matches. Figure taken from [7].

Method	# of Fragments	F-score
WPA	90	0.823
HC	42	0.411
DCG	7	0.082

Table 1: Based on table from [7]

which was HC. WPA's F-score was also more than double the second best, which was HC again. The size and accuracy of the WPA's solution are vastly improved over both the HC and DCG solutions, reconstructing much more of the painting and lessening the workload of the experts who would finish the reconstruction, if it was needed. Along with improved size and accuracy, something to note is that the mistakes present in the WPA's solution are all along the edge of the cluster while the mistakes from both HC and DCG are within the interior of the cluster.

3.2.2 *Comparison with Castañeda et al.*

When comparing the WPA to Castañeda et al. [3] they didn't have access to the code or data for it, so they just compared the visuals. The solutions from the WPA and Castañeda et al. [3] can be found in Figure 7. When looking at the visuals, it can be seen that, while the Castañeda et al. and WPA solutions are of similar size, WPA had fewer incorrect matches. As was also seen in Section 3.2.1, Castañeda et al.[3] had more mistakes within its interior, while the WPA had only three mistakes, all of which are along the outer edges of the solution, two of them from the same fragment. Since there are less mistakes within the solution from WPA, if experts were to continue the reconstruction, they would make less mistakes as well, and would be able to have a higher confidence that the solution produced by the algorithm is accurate.

4. TILE PANEL ALGORITHM

The Tile Panel Algorithm (TPA) developed by Rika et al. in [5] is a hybrid of a GA and a DL based compatibility measure (DLCM) to reconstruct Portuguese tile panels. In the TPA, the compatibility measure determines the compatibility between two given tiles to determine if they share an edge.

The DLCM, when given two tiles in some orientation, returns a real number called the compatibility score, determining the compatibility of the two tiles by processing the images on the tiles. To more easily process the image, Rika et al. use four networks: one for the combined color channels (RGB), one for red (R), one for green (G), and one for blue (B). All four networks follow the same process, and each network returns a score that is then added to give the overall compatibility score. When it is required to find the most compatible piece, or tile, within the pool of unconnected pieces, the DLCM determines the compatibility score between the free edge and the pieces in the pool.

Figure 7: The reconstruction on the left is the visual from Castañeda et al. [3], while the reconstruction on the right is from the Wall Painting Algorithm. Green lines signify correct matches, while red lines signify incorrect matches. Figure taken from [7].

The GA portion of the TPA is based on the *kernel-growth* scheme proposed by Sholomon et al. [6]. A kernel-growth GA selects parents during the selection procedure like a typical GA, but during the recombination process, it starts with a random piece, or collection of connected pieces, called a kernel and gradually adds more pieces on the edges of the kernel until a complete child is produced, not just a portion as happened with the WPA. For example, in the TPA a complete child would be a complete tile panel. This scheme is more conductive to tile panels because the pieces are uniform in shape, unlike wall paintings fragments which are inconsistent in shape.

For the TPA, Rika et al. use six hierarchical phases to decide which piece to add, as follows:

- **Phase I:** If there is a free boundary, i.e. an unmatched edge, in the kernel, and an unused neighboring piece from the parent with a greater fitness score that has an average compatibility score greater than $max(0.8, C_{mean})$, then the neighboring piece is added to the kernel. The average compatibility score is between the piece and all of its neighbors and C_{mean} is the parent's average compatibility across the border of the tile panel.
- **Phase II:** Similar to Phase I, but using the parent with the lower fitness score.
- **Phase III:** If both parents contain the same neighboring piece on the boundary being considered, then the piece is added to the kernel.
- **Phase IV:** If the most compatible piece to a given boundary in the kernel is free, i.e. not already in the kernel, then that piece is added
- **Phase V:** Similar to Phase IV, but adds the second most compatible piece.

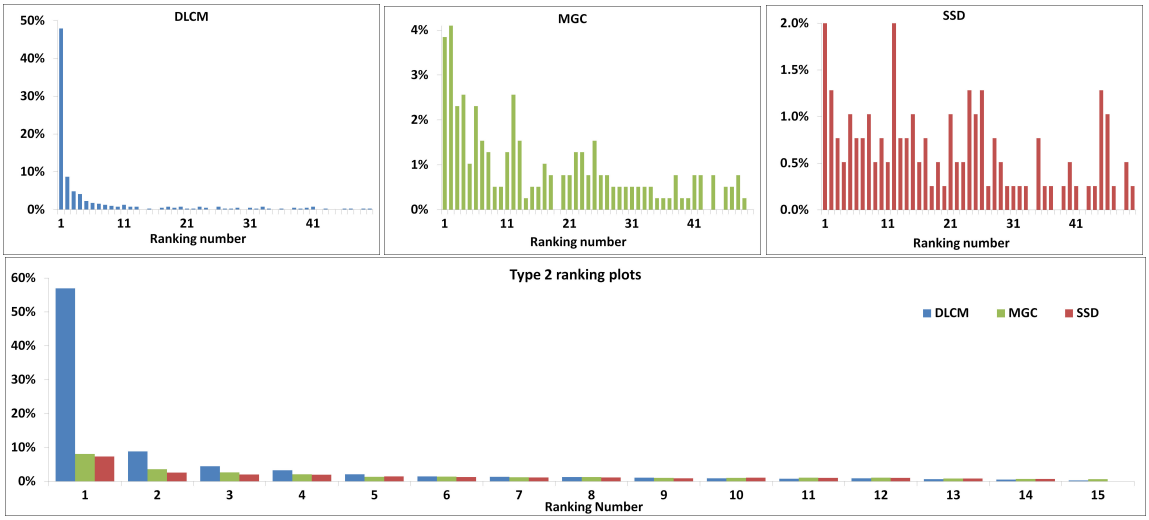

Figure 8: Top three plots correspond to a single test image, for the compatibility measures: Deep Learning compatibility measure (DLCM), Mahalanobis gradient compatibility (MGC), and sum of squared differences (SSD). The bottom graph corresponds to the average of the entire test set. All tests are done in the Type 2 test case. Figure taken from [5]

- **Phase VI:** Picks a random free piece, and adds it to the free boundary in the kernel.

Rika et al. introduce mutation to their GA by skipping Phase I and II with 10% probability and Phase III with 20% probability. They use this mutation because it gives the opportunity for errors from previous generations to be fixed, since the phases which aren't skipped, IV-VI, don't rely on the child's parents [5].

Rika et al.'s GA differs from the kernel-growth proposed by Sholomon et al. [6] - the one it's based on - by using hierarchical phases rather than the notion of *best-buddies*, where both pieces consider the other as the most compatible. The reason Rika et al. don't use the best-buddies principle is because when combined with their DLCM and used on the Portuguses tile panels, the best-buddy pairs were only correct with 70% probability.

4.1 Tile Panel Algorithm Results

Rika et al. used images of tile panels when testing the DLCM and the TPA as a whole. Half of them were reconstructed by experts from the National Tile Museum (Museu Nacional do Azulejo, MNAz) in Lisbon, Portugal, and the other half were found online. They ran two different test cases: Type 1, where orientation of the pieces was known, and Type 2, where orientation was unknown. The DLCM results are discussed in Section 4.1.1 and the results for the whole TPA are discussed in Section 4.1.2.

4.1.1 Deep Learning Compatibility Measure Results

The DLCM was tested against the compatibility measures sum of squared differences (SSD), which uses the sum of squared color differences of all neighboring pixels and color bands, and Mahalanobis gradient compatibility (MGC) [4], which penalizes changes in color intensity from the image and uses the Mahalanobis distance to find the color channels' covariance. The results of the tests can be seen in Table 2 and Figure 8. As is seen from Table 2, the DLCM has a significant increase in accuracy in both Type 1 and Type 2 test cases. The DLCM is accurate over half the time for both cases, while the SSD and MCG are less than 20% acccurate for Type 1 and less than 10% for Type 2.

Compatibility Measure	Type 1	Type 2
SSD	12.7%	7.3%
MGC	17.4%	9.1%
DLCM	68.45%	56.9%

Table 2: The percentage of how often the compatibility measure ranked the most compatible piece as the first choice for Type 1, known orientation, and Type 2, unknown orientation, test cases. Based on table from [5]

Figure 8 shows the frequency that the most compatible piece was placed at the given rank by the compatibility measures. It attests to the relatively high quality of the DLCM due to the having the highest first rank frequency and the sharp, mostly consistent decrease in frequency after the first rank, while the SSD and MGC are more uniformly distributed. This means that the DLCM has a higher chance to rank the most compatible piece towards the top. Since the SSD and the MGC have a more even distribution, there's a similar chance that the most compatible piece will be ranked towards first as ranked towards last. For example, when using SSD there is an equal chance it will be ranked first or twelfth. Another thing of note for the single test image rankings, is that the DLCM had the most compatible piece ranked first just under 50% of the time, but MGC and SSD only ranked it first just under 4% and 2%, respectively.

4.1.2 Whole Tile Panel Algorithm Results

When testing the TPA, Rika et al. compared it against two algorithms: an algorithm proposed by Gallagher [4] that uses the MGC compatibility measure and the original kernel-growth algorithm proposed by Sholomon et al. [6] using the

Method	Type 1		Type 2	
	Known dims.	Unknown dims.	Known dims.	Unknown dims.
Gallagher + MGC	—	13.0%	—	3.5%
kernel-growth + DLCM	84.5%	—	58.6%	—
TPA (using DLCM)	96.3%	96.0%	86.8%	82.2%

Table 3: Test results found by Rika et al. using the test cases of Type 1, known orientation, and Type 2, unknown orientation, along with known and unknown dimensions of the tile panel. Results are the percentage of correct matches within Tile Panel. Based on table from [5]

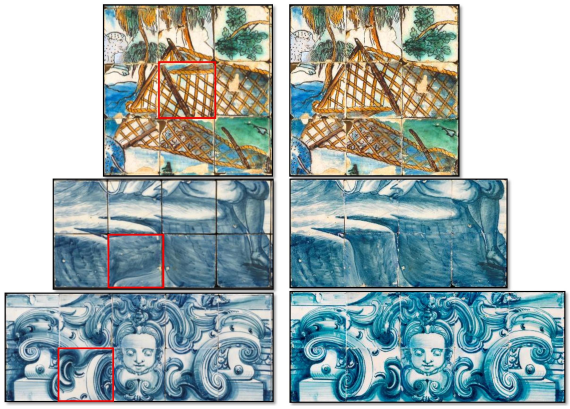

Figure 9: Left: Images with human errors with tile highlighted in red. Right: image produced by TPA with unknown orientation and known dimensions

DLCM as described above. As they say in [5], they decided to use Gallagher's algorithm because it can be used in the variants they wanted to test, and it is relatively state of the art. They ran a test for each test image ten times for the TPA and reported the best result; no other results were reported. Along with Type 1 and 2 test cases, Rika et al. used a further subdivision where the dimensions of a tile panel where known or unknown, i.e. the algorithms knew the tile panel was 10x20 or it didn't know. The results from the tests can be found in Table 3.

As can be seen from Table 3, the TPA is a vast improvement over Gallagher's GA and MGC, for both Type 1 and Type 2 test cases. In addition, their GA was an improvement because, as seen in Table 3, even when they combined their DLCM with another kernel-growth GA, their GA performed better.

4.2 Discovered Human Errors

While running their tests, they ran into the situation were it was being reported that the reconstruction produced was wrong, even though the "overall global score was greater than the ground truth" [5]. After further manual inspection, it was found out that the museum experts had not assembled the ground truth panel correctly, and the TPA's evolved solution was correct. Figure 9 shows the tile panels and the pieces in question.

5. CONCLUSION

There are strengths and limitations to both approaches I have talked about. The reason the TPA can use a kernel-growth based GA is because of the uniformity of the pieces, since all the tiles will have a square shape, excluding broken tiles. It isn't conductive to the WPA problem space, because the shapes are so inconsistent. This is why the WPA's approach was to use the shape of the pieces and focused less on the image.

More can still be done to improve algorithms in this problem space. For the WPA, a next step could be to apply it to wall paintings reconstructed by experts. For the TPA, some next steps would be to be able to account for missing tiles, since at the moment the algorithm assumes all the tiles are present. It would also be great if the algorithm could also deal with tiles from more than one panel, i.e. given a mix of two tile panels at once, return two correctly constructed tile panels.

Acknowledgments

I would like to thank Nic McPhee and KK Lamberty for their advice and feedback.

6. REFERENCES

[1] Deep learning, Oct 2019. `https://en.wikipedia.org/wiki/Deep_learning`.

[2] Genetic algorithm, Oct 2019. `https://en.wikipedia.org/wiki/Genetic_algorithm`.

[3] A. G. Castañeda, B. J. Brown, S. Rusinkiewicz, T. A. Funkhouser, and T. Weyrich. Global consistency in the automatic assembly of fragmented artefacts. In *VAST*, volume 11, pages 73–80, 2011.

[4] A. C. Gallagher. Jigsaw puzzles with pieces of unknown orientation. In *2012 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 382–389, Los Alamitos, CA, USA, jun 2012. IEEE Computer Society.

[5] D. Rika, D. Sholomon, E. O. David, and N. S. Netanyahu. A novel hybrid scheme using genetic algorithms and deep learning for the reconstruction of portuguese tile panels. In *Proceedings of the Genetic and Evolutionary Computation Conference*, GECCO '19, pages 1319–1327, New York, NY, USA, 2019. ACM.

[6] D. Sholomon, O. David, and N. S. Netanyahu. A genetic algorithm-based solver for very large jigsaw puzzles. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2013.

[7] E. Sizikova and T. Funkhouser. Wall painting reconstruction using a genetic algorithm. *J. Comput. Cult. Herit.*, 11(1):3:1–3:17, Dec. 2017.