אוניברסיטת בן-גוריון בנגב
Ben-Gurion University of the Negev

הפקולטה למדעי ההנדסה
המחלקה להנדסת חשמל ומחשבים
Faculty of Engineering Science
Dept. of Electrical and Computer Engineering

פרויקט ההנדסי שנה ד׳
Fourth Year Engineering Project

דו״ח מכין
Preliminary Design

רובוט לפתרון פאזלים

Robotic jigsaw puzzle solver

Project number:	**p-2013-091**	**מספר הפרויקט:**

Students (name & ID):	**Nadav Erell, ID 301249397** **Roey Nagar, ID 301021614**	**סטודנטים (שם ו ת.ז.):**

Supervisors:	**Prof. Ohad Ben-Shahar,** **Dr. Rami Hagege**	**מנחים:**

Sponsors:	**תומכים:**

Submitting date:	**02/12/12**	**תאריך הגשה:**

Contents

1 Abstract

As time goes by, computers and robots become increasingly capable of solving many problems, and are continually posed with new and more difficult ones. In the project, we will attempt one such challenge which has yet to be considered - that of solving jigsaw puzzles.

In this project, we will demonstrate a computer vision system for automatically solving jigsaw puzzles with square pieces. The system will comprise of a camera to acquire the puzzle pieces data, a computer which will process, analyze and solve the puzzle, and a robotic arm which will rearrange the pieces on the work desk.

We will use an algorithm developed in the CS department at BGU for solving square jigsaw puzzles, and will attempt to extend and improve upon it. In particular, we mean to research such problems as sensory uncertainty in acquiring piece data and its effect on the algorithm, handling missing pieces, and handling pieces with unknown orientation.

תקציר

ככל שעובר הזמן, מחשבים ורובוטים הולכים ומפתחים יכולות מתקדמות יותר להתמודדות עם שלל בעיות, ובהתאם מוצבות בפניהם בעיות חדשות ומסובכות יותר ויותר. בפרוייקט זה ננסה להתמודד עם אתגר מעניין אשר עדיין לא נבחן - פיתרון פאזלים.

בפרויקט זה, בכוונתנו לבנות מערכת ראייה ממוחשבת אשר תפתור באופן אוטומטי פאזלים עם חלקים מרובעים. המערכת תכלול מצלמה לזיהוי חתיכות הפאזל, מחשב לעיבוד וניתוח המידע ופיתרון הפאזל, וזרוע רובוטית אשר תסדר את החתיכות מחדש על שולחן העבודה.

אנו נשתמש באלגוריתם לפיתרון פאזלים אשר פותח במחלקה למדעי המחשב באוניברסיטת בן־גוריון, וננסה לשפר ולהרחיב אותו. בפרט, בכוונתנו להתמודד עם בעיות כגון רעש ואי־ וודאות בזיהוי חתיכות הפאזל והשפעה על ביצועי האלגוריתם, וכן מקרים בהם חסרות חתיכות, או שהאוריינטציה של החתיכות אינה ידועה.

2 Project Goals

The field of computer vision deals with acquiring, processing and inferring meaning from images. There are countless problems and possible applications, such as automating robotic systems, detecting and recognizing human faces, and creating new ways of interfacing with technology. One such problem is that of solving jigsaw puzzles, which, despite its recreational origin, may have applications in several fields.

In recent years, academic research has focused on solving jigsaw puzzles with square pieces, where the pieces are part of a digital image. Given ideal conditions, current algorithms have managed to solve puzzles of thousands of pieces with reasonable success rate. However, there remain limitations which have yet to be solved, such as handling puzzles with missing pieces, or handling piece data uncertainty and noise inherent to any camera-acquired image. We intend to tackle some of these limitations, whilst physically implementing a robotic system for solving such jigsaw puzzles.

Our project can be roughly divided into two main goals:

- Implementing an automated system for solving a *physical* jigsaw puzzle. The system will comprise of a standard camera, a computer for processing the image data and solving the puzzle, and a robotic arm to move the puzzle pieces on the table into their final position. The implementation of the system will require programming the robotic arm, which has a limited communications API. In order to streamline the process, we intend to create an interface in Matlab for controlling the robotic arm, which may also greatly benefit future users.

- Researching various expansions and improvements to the puzzle solving algorithm developed in the CS department at BGU, including:
  - Studying the effects of sensory uncertainty and distortions on the performance[1] of the algorithm.
  - Expanding the algorithm to handle puzzle pieces with unknown orientation.
  - Improving the greedy puzzle-solving algorithm by an informed choice of an initial seed piece, instead of the current random mechanism.
  - Expanding the algorithm to handle puzzles with missing pieces.

[1] Refer to the discussion of Performance Measures in the Literature Review section for more information.

3 Research Proposal

Robotic jigsaw puzzle solver

We intend to create a physical system, which will implement the algorithm and be capable of solving real-world jigsaw puzzles. The system will comprise mainly of a robotic arm, the Motoman NX100 from Yaskawa, paired with a camera and a standard computer for processing the data. The robotic arm is equipped with a suction cap, to allow it to move puzzle pieces around the work desk.

The system is intended to be fully automatic. Once presented with unordered puzzle pieces on the work desk, the system should be able to acquire the puzzle pieces, analyze and process them, apply the algorithm to solve the puzzle, and physically move the pieces to assemble the puzzle on the desk.

This very task, of having a robotic arm automatically analyze and solve a completely unknown puzzle, with no human input, is a difficult thing to achieve. We do, however, hope to be able to correctly solve puzzles with hundreds of pieces. Due to the length of the robotic arm and the size of the suction cap, the system will be inherently limited in terms of the size and number of pieces of the puzzles it will be able to solve. We aim to have pieces as small as 3x3 cm, on a work desk which measures about 40x50cm in size.

The algorithm itself is currently capable of solving puzzles with thousands of pieces, assuming the input data is ideal. Introducing measurement uncertainty and noise into the problem will definitely decrease this number, but we hope to still be able to provide a very good approximate solution for large puzzles, and perhaps still perfectly solve smaller ones. Further performance metrics are detailed later on, as they require a more in-depth understanding of the problem.

Even though it is not considered a critical feature of our system, we can provide rough estimates of the time which will be required to fully solve puzzles. Assuming we run the algorithm on a standard, modern computer, it should take no more than 1-2 minutes to acquire, analyze and solve a puzzle of about 400 pieces. Physically moving the pieces on the work desk could take several more minutes, and is mostly limited by the robotic arm hardware.

Additionally, we will investigate several expansions to the puzzle solving algorithm developed by the CS department at Ben-Gurion University. This will be an attempt to push the envelope in terms of performance and versatility of the state-of-the-art in automatic jigsaw puzzle solvers. Among our research directions are studying the effects of sensory noise and uncertainty on the performance of the algorithm, handling puzzles with missing pieces, puzzle pieces with unknown orientation and attemping to optimize the seed selection in the current algorithm.

While intriguing in its own right, the problem of solving jigsaw puzzles has possible applications in various fields. Several papers have noted uses in speech descrambling [14], reassembling archaeological relics [7, 8, 1], recovering shredded documents or pictures [10, 9], image editing [2] and more. See Section 4.6 for further details.

4 Literature Review

4.1 Introduction

The Jigsaw Puzzle is a type of puzzle, often presented as a game, that requires assembling a picture from numerous small parts. Most commonly, the complete picture is rectangular, and may depict an object, a scene or some other design. The pieces are often oddly-shaped, in a way that prevents most pieces from being assembled next to one another. Puzzles range in size and complexity from just a handful of pieces, usually intended for small children, to very large puzzles with hundreds or even thousands of pieces.

The problem of solving a jigsaw puzzle with a computer was first proposed by Freeman and Garder in 1964 [5], who declared it as a problem in pattern recognition. However, it is only in recent years that significant advances have been made in automatically solving large puzzles (i.e. several hundred pieces). It is important to note at this point that the Jigsaw Puzzle problem has been proven to be NP-Hard by Demaine *et al.* [4], implying that any attempt to automatically solve a large enough puzzle must employ some heuristic approach.

The problem generally requires constructing a complete image from a set of unordered pieces. Other characteristics, such as piece size, shape or orientation may vary, resulting in a class of similar problems. Early attempts at automatically solving jigsaw puzzles [5, 12, 13] predominantly relied on matching the shapes of the pieces rather than their content (color). The contours of the pieces provided a critical factor in determining whether two pieces could be placed next to each other in the constructed image. The contents of the image were incorporated in later algorithms to supplement the shape information and improve the performance of these algorithms.

4.2 Jigsaw Puzzles with Square Pieces

Recently, the focus has shifted to a variant of the problem where the puzzle pieces are square rather than irregularly-shaped. This variant is much harder to solve, as it forces any solving algorithm to focus exclusively on the image content and color information. This is also the problem we tackle ourselves, where we will attempt to improve upon current algorithms and perhaps expand to other variations.

In 2010, Cho *et al.* [3] introduced a probabilistic algorithm for solving a square jigsaw puzzle, along with compatibility metrics between puzzle pieces and measures for evaluating an algorithm's performance. These ideas provided the basis for further, rapid improvement of the state-of-the-art by Pomerantz *et al.* [11], who managed to automatically solve puzzles of 3,300 pieces, and Gallagher [6], who went as far as 9600 pieces while also not relying on piece orientation information.

In recent years, we have seen several algorithms take different approaches to the jigsaw puzzle problem. We will summarize these approaches very briefly for the sake of completeness, as this project will focus only on the algorithm presented by Pomerantz *et al.* [11] in the CS department in Ben-Gurion University of the Negev.

4.3 Compatibility Metrics

Following is a summary of the main compatibility metrics presented in recent years and their relative performance [3, 6, 11]. These metrics deserve our attention as they are an important component in all recent algorithms, and moreover may need reevaluation in the presence of measurement noise. Note that we adopt the notations used in [11] for this discussion.

A compatibility metric predicts the likelihood that two puzzle pieces will be placed next to each other in the reconstructed image. We can define a function $C\left(x_i, x_j, R\right)$ to denote the compatibility that part x_j will be placed in direction R relative to part x_i, where $R \in \{l, r, u, d\}$, representing the spatial relationship between the pieces. The function has a range of values of $[0, 1]$, where the higher the value, the more likely that the two pieces be neighbors with the given direction.

In order to evaluate the success of the compatibility metric, we measure the accuracy as the ratio between correct placements and the total number of possible placements for a given puzzle. A placement by a compatibility metric is determined to be correct if

$$\forall x_k \in Parts, \quad C\left(x_i, x_j, R\right) \geq C\left(x_i, x_k, R\right) \tag{1}$$

4.3.1 Dissimilarity-Based Compatibility

Cho *et al.* defined this metric as summing the squared color difference in LAB space along the abutting boundaries of two pieces:

$$D\left(x_i, x_j, r\right) = \sum_{k=1}^{K} \sum_{d=1}^{3} \left(x_i\left(k, K, d\right) - x_j\left(k, 1, d\right)\right)^2 \tag{2}$$

This metric proved most efficient among those tested by Cho *et al.*, despite its relative simplicity.

4.3.2 $(\mathbf{L_p})^{\mathbf{q}}$ Compatibility

This type of compatibility is naturally derived from the dissimilarity-based compatibility, which is related to the L_2 norm between the boundaries' vectors. In principle, any $(L_p)^q$ norm can

be used in the same manner:

$$D_{p,q}\left(x_i, x_j, r\right) = \sum_{k=1}^{K}\sum_{d=1}^{3}\left(\left|x_i\left(k, K, d\right) - x_j\left(k, 1, d\right)\right|^p\right)^{\frac{q}{p}} \tag{3}$$

Pomerantz *et al.* [11] showed empirically that optimal results for their solver can be obtained by choosing $p = 0.3$ and $q = \frac{1}{16}$:

$$D_{p,q}\left(x_i, x_j, r\right) = \sum_{k=1}^{K}\sum_{d=1}^{3}\left(\left|x_i\left(k, K, d\right) - x_j\left(k, 1, d\right)\right|^{\frac{3}{10}}\right)^{\frac{5}{24}} \tag{4}$$

These results can be qualitatively explained by the fact that an L_2 norm severely penalizes sharp transitions which happen to fall on the pieces' boundaries. Such differences can, however, occur in natural images, and this is a possible explanation for the superiority of less harsh norms.

4.3.3 Prediction-Based Compatibility

Clearly, the compatibility metrics described thus far used relatively little information, as they only take into account a single row or column of pixels. We know, however, that in most images, there are gradual transitions between colors, and rare are the cases where the transition is so sharp as to be evident in a single pixel only. This leads directly to the idea of using the derivative of the pixels in the direction of the piece boundary to estimate the pixel on the neighboring piece. This prediction can then be compared against the actual pixel using an $(L_p)^q$ norm as before. To complete the metric, the process is repeated from both sides to obtain the final compatibility measure:

$$\begin{aligned} Pred\left(x_i, x_j, r\right) = \sum_{k=1}^{K}\sum_{d=1}^{3} \Big[&\left(\left[2x_i\left(k, K, d\right) - x_i\left(k, K-1, d\right)\right] - x_j\left(k, 1, d\right)\right)^{\frac{3}{10}} + \\ &+ \left(\left[2x_j\left(k, 1, d\right) - x_j\left(k, 2, d\right)\right] - x_i\left(k, K, d\right)\right)^{\frac{3}{10}}\Big]^{\frac{5}{24}} \end{aligned} \tag{5}$$

4.3.4 Mahalanobis Gradient Compatibility

This new compatibility metric provides minor improvements over the prediction-based compatibility [6]. This compatibility metric penalizes changes in intensity gradients rather than changes to the intensity itself, stemming from the expectation that a gradient near the edge of one piece will continue to the next. Additionally, a Mahalanobis distance is used instead of the $(L_p)^q$ norm used before. We will not delve into the implementation details at this point, as they are immaterial to us in this project, and may be found in the original paper.

4.3.5 "Best Buddies" Estimation Metric

While not quite the same as the compatibility metrics, this "estimation metric" proposed by Pomerantz *et al.* [11] is used to estimate the quality of the solution without access to ground truth, and is used in the greedy algorithm described below. Two pieces x_i, x_j will be called *Best Buddies* if

$$\begin{aligned} &\forall x_k \in Parts, && C\left(x_i, x_j, R_1\right) \geq C\left(x_i, x_k, R_1\right) \\ & && and \\ &\forall x_p \in Parts, && C\left(x_j, x_i, R_2\right) \geq C\left(x_j, x_p, R_2\right) \end{aligned} \tag{6}$$

where R_1 and R_2 are opposite relations. Intuitively, this means that both pieces "agree" that the other piece is their most likely neighbor in the corresponding spatial relation.

4.4 Solving Strategies

4.4.1 Loopy Belief Propagation and Prior Evidence

Cho *et al.* [3] attempted two methods in solving the problem, both requiring some form of "evidence" to assist a loopy belief propagation method in finding the most likely configuration of pieces. The first, dubbed the "dense-and-noisy" used a graphical model which requires the availability of a low-resolution image to help approximate the location of each piece in the reconstructed image. Due to the lack of a priori knowledge in this instance of the jigsaw puzzle problem, Cho *et al.* devised a method to estimate such an image using statistical methods applied to the histograms of the pieces. Certain properties tend to appear in natural images, such as brighter pieces appearing at the top of an image due to the sky. Their experiments showed that a very coarse, low-resolution estimate can indeed be obtained in this manner, though predictably, in some cases this approach fails completely.

The second, considerably simpler method, used "sparse-and-accurate" evidence, namely using a small number of anchor pieces, fixed to their correct location in the assembled puzzle. With around 8-10 pieces, relatively good reconstruction performance could be obtained on puzzles of 432 pieces.

4.4.2 Iterative Greedy Solver with Zero Knowledge

While groundbreaking in its success at solving a puzzle of 432 pieces, these attempts by Cho *et al.* clearly left much to be desired. Both strategies were inadequate under certain conditions, such as the lack of given anchor points, or an image that does not conform well to the statistical properties used to train the algorithm. The greedy algorithm presented by

Pomerantz *et al.* [11] improved upon prior attempts in several ways. The algorithm uses no clues or prior knowledge whatsoever, yet still managed to greatly improve results both in terms of the various performance measures, and in its ability to solve puzzles with thousands of pieces. At the time of publication, the algorithm managed to solve puzzles up to 3,300 pieces, and later on went as far as 10,000 pieces. The paper also introduced the Prediction-Based Compatibility, which yielded better results than prior compatibility metrics.

The algorithm comprises of three modules, each solving a separate sub-problem. The following succinct description of the sub-problems is taken directly from the paper [11]:

- The *placement problem*: Given a single part or a partial constructed puzzle, find the position of the remaining parts on the board.
- The *segmentation problem*: Given a placement of all parts on the board, i.e. an approximated solution, divide it to segments which are estimated to be assembled correctly, disregarding their absolute location.
- The *shifting problem*: Given a set of puzzle segments, relocate both segments and individual parts on the board such that a better approximated solution is obtained.

The **placer** module begins with an initial, random seed piece, and "grows" the remaining puzzle pieces around it while following simple heuristics and using a compatibility metric between pieces. An important heuristic is that empty spaces neighboring many already placed pieces are given a higher priority, since there is more information to match them. For each such empty space, the placer chooses a "best buddy" if available, or the piece with the highest compatibility metric with the current neighbors otherwise.

Once the placer finishes placing all of the pieces, the **segmenter** looks for regions which are assembled correctly with a high level of confidence. Such segments are identified whenever two neighboring pieces agree on being "best buddies". This criterion has experimentally been found to be correct 99.7% of the time.

The **shifter** now relocates the largest segment found, and uses it as a new "seed piece" for the next iteration. The placer now continues as before, with the process repeated until the evaluation of "best buddies" reaches a local maximum (i.e. further iterations do not improve the size of the largest segment).

4.4.3 Tree-based Reassembly

Further improvement has recently been achieved by Gallagher [6], who used a tree-based greedy algorithm inspired by Kruskal's algorithm for finding a minimum spanning tree. The puzzle is modeled as a graph, where each piece is represented by a vertex, and edge weights correspond to the compatibilities between pieces. One might wonder if the problem has suddenly been reduced

to the relatively simple problem of finding a MST. Unfortunately, there remain difficulties, such as preventing the reconstructed pieces from overlapping. Introducing the required constraints results once again in an NP-Hard problem, but using certain heuristics, an efficient solution can be found. Applying this method, along with the new Mahalanobis Gradient Compatibility, Gallagher was able to solve puzzles of up to 9,600 pieces, while also handling puzzles where the piece orientation is unknown.

4.5 Performance Measures

As mentioned before, a number of performance measures are used to evaluate and compare the success rate of the various algorithms. We shall focus only on those measures relevant to our algorithm, and which we will also use experimentally to evaluate the effects of our own modifications to the algorithm.

Direct Comparison Metric: Calculates the ratio between the number of pieces placed by the algorithm in their absolute correct position within the reconstructed image, and the total number of pieces.

Neighbor Comparison Metric: Calculates the ratio between the number of correct neighbor placements and the total number of neighbors. This is a much more permissive metric, as it allows for an entire segment of the image to be shifted and still score fairly well.

Perfect Reconstruction: Binary score, indicating whether a puzzle has been assembled perfectly or not.

Gallagher [6] provides some comparison between the recent algorithms in terms of these metrics, when applied to the image set included with [3]:

	Direct	Neighbor	Perfect
Cho *et al.* [3]	0.10	0.55	0
Pomerantz *et al.* [11]	0.94	0.95	13
Tree-Based + LAB SSD [6]	0.814	0.892	8
Tree-Based + MGC [6]	0.953	0.951	12

Table 1: Performance comparison for puzzle assembly (oriented jigsaw pieces) for puzzles with 432 pieces.

Clearly, for oriented puzzle pieces, [6, 11] provide nearly the same performance, despite employing very different solving strategies.

Also, see *Appendix 1: Experimental results of the greedy puzzle solver*, for more information on current performance.

4.6 Possible Applications

As we have seen, the jigsaw puzzle problem is intriguing, and certainly merits research without additional motivation. However, it has been shown that several practical problems can be modeled as some variation of the jigsaw puzzle problem. We will not go into much detail, as it is not immediately relevant to our project, but we can briefly point out some interesting possibilities.

Several uses have been explored in the field of archaeology, where similar algorithms have been used for automatically providing an estimated reconstruction of ancient artifacts from broken pieces. A few papers in recent years have specifically mentioned jigsaw puzzles in the context of reconstruction attempts:

- Matching of fresco fragments. A team of researchers at the Akrotiri Excavation site built a system for capturing images of wall painting fragments [1]. While they focused on other properties of the pieces, mainly geometric, they have indicated that incorporating information such as color and texture among other things may improve performance and allow them to propose more relevant matches.
- Reconstructing the Severan Marble Plan of Rome. An attempt to apply algorithms on color photographs and three-dimensional computer models to find new matches and positioning of fragments is detailed in [8]. The paper once again cites geometrical properties as the most critical clues, but also mentions color, texture, marble veining direction and more as additional potentially useful information.

A somewhat similar problem is that of recovering shredded documents or pictures, where attempts have been made as well to apply puzzle solving algorithms or ones similar in principle [10, 9].

Another interesting use has been suggested by Zhao *et al.* [14], in a problem of speech descrambling. Many speech scramblers are based on a permutation of speech signals either in the time or the frequency domain. The paper suggests a method to treat the problem as that of solving a jigsaw puzzle much like the one we describe in this project.

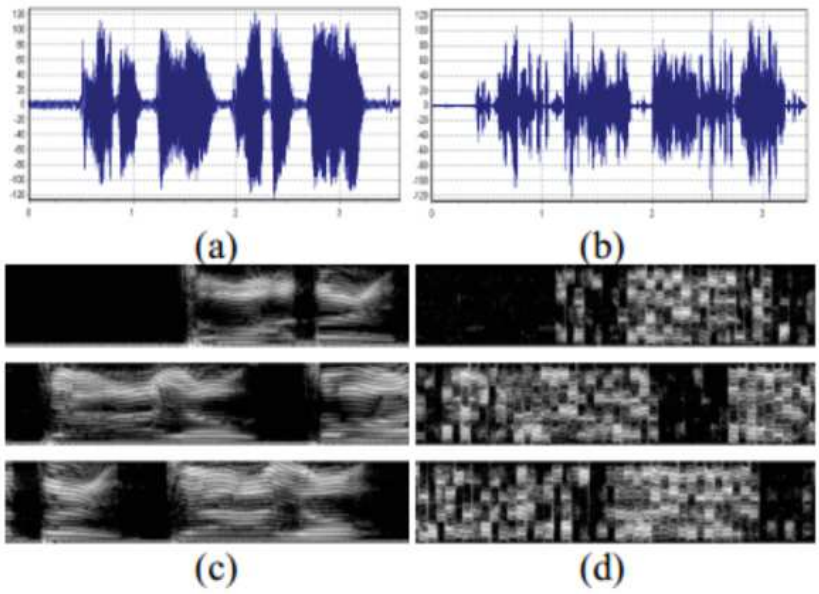

Figure 1: Taken from [14]: The idea of scrambling a speech signal: (a) the original speech signal in time domain, (b) the scrambled speech signal in time domain. (c) the original spectrogram, and (d) the scrambled spectrogram.

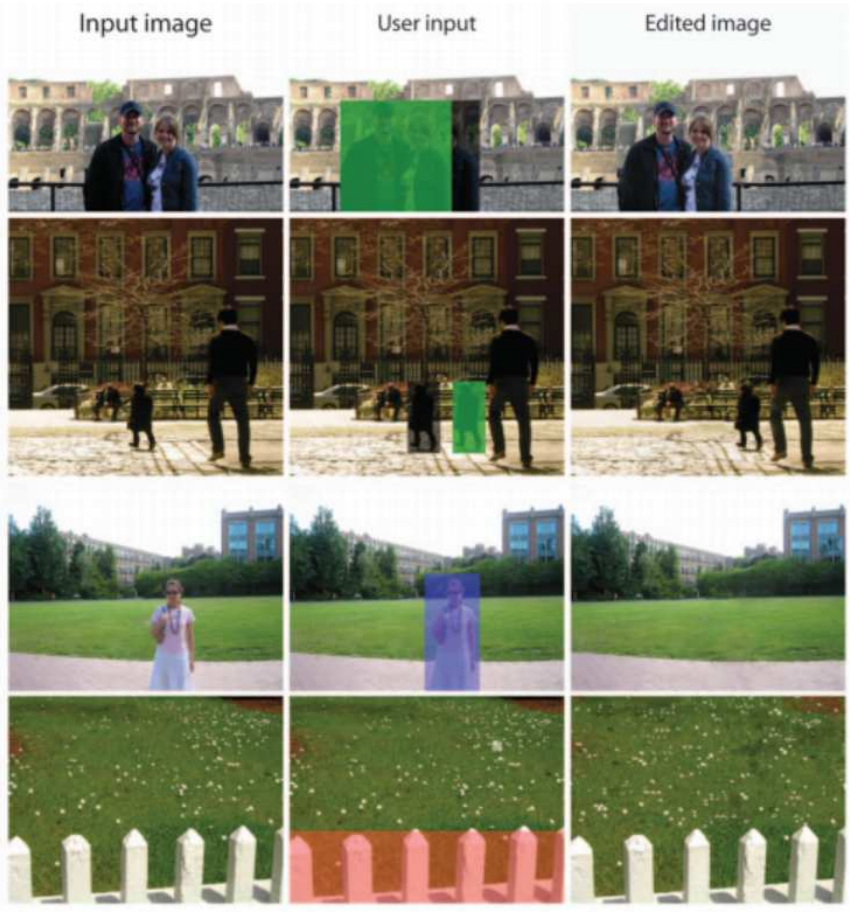

Figure 2: Taken from [2]: Image editing examples. (Color codes: red—fix, green—move to, black—move from, blue—remove, white—favor.)

Finally, there are several known applications in image editing. One of those is the concept of *patch transform*, introduced by Cho *et al.* [2], which allows a user to manipulate *patches* of an image separately, and apply an inverse transform to synthesize modified images. The reconstruction of the image reduces to solving a jigsaw puzzle if no constraints are imposed.

As we can see, there is no lack of possible applications for efficient and versatile jigsaw puzzle solvers. As time goes by, it is likely that new and novel uses may surface, perhaps enabled by the recent great improvement in performance of puzzle solvers.

5 Work Plan

5.1 Physical system implementation

There are several non-trivial steps we must take in order to set up the system and perform the simplest proof-of-concept for solving a small puzzle:

- Place a camera above the work desk, and connect it to a computer. The camera will be placed about 180-190 cm above the desk, to minimize distortions resulting from the camera angle. Initial tests performed at such distance with a standard digital 8 MP camera proved adequate in terms of pixel density, allowing for pieces about 50x50 pixels at the smallest piece size we expect to handle. This is almost double the amount of pixels the algorithm usually expects, so there should be no need for higher resolution.
- Use the input data from the camera to automatically extract the puzzle pieces from the image. This step requires edge detection, segmentation, and careful processing to bring all acquired pieces to the exact same size in pixels, while suffering the minimum amount of distortions. Performing this step effectively will require studying techniques in image processing and learning how to apply them to our specific problem. Special attention will have to be paid to the fact that while the resulting pieces must be perfectly square and of exactly the same size, the raw data will definitely not be quite so. We will need to attempt various transformations and corrections, while testing which operations prove the least detrimental to the algorithm performance. Such operations may include rotation, perspective transformation, stretching, trimming and duplicating pixels.
- Apply the algorithm to the acquired pieces, resulting in a required placement for each piece in the assembled puzzle.
- For each piece, identify its current location on the work desk, apply appropriate coordinate transformations and send the robotic arm the coordinates for the piece. Then, pick up the piece, and move it to a second coordinate specifying its required location in the assembled puzzle. Controlling the robotic arm will be done via a special communication API called MotoCom32, programmable with C++. We intend to create another API abstraction layer so that we can issue commands directly from Matlab.

A flowchart and diagram of the planned system are shown in Figures 3, 4.

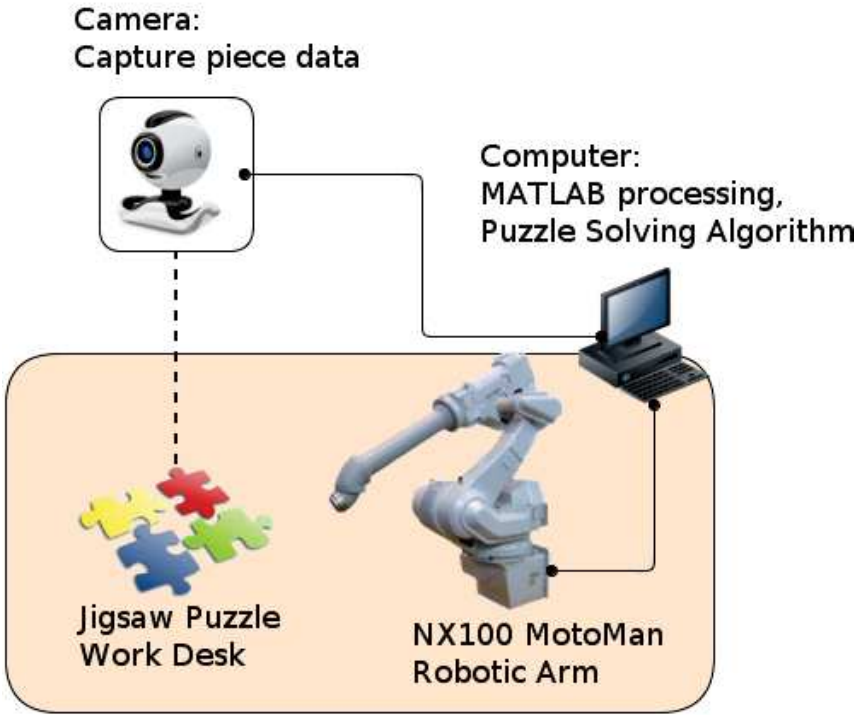

Figure 3: Puzzle solver general system diagram.

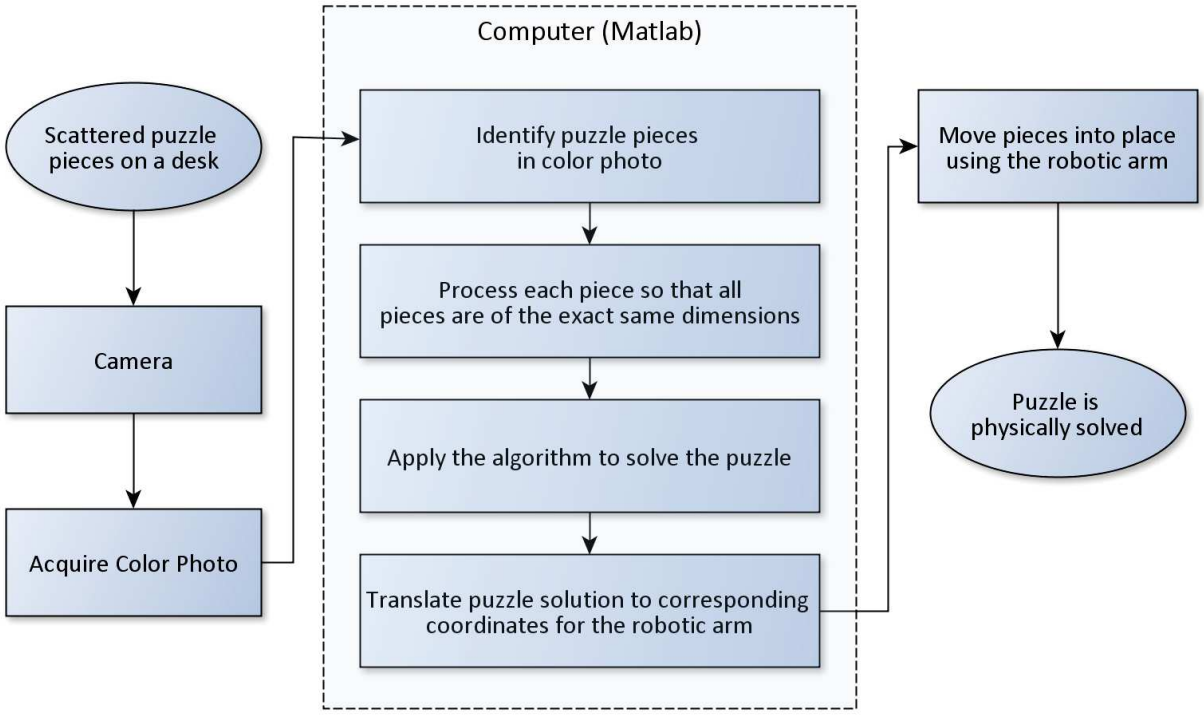

Figure 4: Puzzle solver flowchart.

5.2 Algorithm

There are currently certain limitations to our algorithm, which we have mentioned before. We will therefore begin by presenting the system with puzzles conforming to all of these limitations. If we are successful in tackling these limitations on the digital front, we will of course apply it to the physical system as well.

Current known limitations include the inability of the algorithm to handle puzzle pieces with unknown orientation, and its inability to handle puzzles with missing pieces. While the former problem has been solved recently [6], the latter is an open problem which may prove impractical for us to handle at this stage.

Any change to the algorithm will be tested according to the experimental framework described in [11], focusing on the image set from [3]. This will provide us with direct and meaningful feedback on the effects our changes have on the performance metrics.

We sum up the known issues we intend to investigate and an estimation of our likelihood to have a significant result concerning that issue in the table below.

Open Issue	**Likelihood of achieving significant result**
Effect of uncertainty and distortion in piece data	High
Initial seed	Moderate
Piece orientation	Moderate
Missing pieces	Low

Table 2: Known issues with the puzzle solving algorithm that we intend to research.

5.3 Constraints

There is a physical limit to the size of the work desk and the robotic arm, as well as the suction cap used to move pieces. These constraints limit us to puzzles of about 40x50 cm in total size, and pieces of about 3x3 cm. Finer measurements and experiments may allow us to reach slightly better dimensions, but we shall not expect to be able to handle puzzles of more than 200 pieces at the most under current conditions.

Schedule: The project must be completed by August 2013, setting a stern deadline to work towards. Detailed schedule information may be found below.

5.4 Assumptions

We assume that the amount of noise and distortions introduced by the camera and our processing of the data will be low enough to still successfully apply the puzzle-solving algorithm to large enough puzzles (and small enough pieces). If this is not the case, and the performance measures take too big of a hit, it may require introducing changes to the algorithm. This will steer our focus into researching these effects, and perhaps lead to introducing new compatibility metrics for the case of noisy input data.

In addition, if the problem proves more difficult than anticipated, we will be forced to limit ourselves to relatively simple puzzles. While perhaps somehwat disappointing, even such a scenario should not prove disastrous to the project, and since it is very lightly-researched, may in fact prove quite interesting.

5.5 Known Risks

The NX100 robotic arm may prove difficult to program, but we hope it will prove feasible within our time constraints. It is a professional piece of equipment which may take time to master, but with the help of a contact person from the Yaskawa company, we assume we'll be able to handle it. If this assumption proves wrong, it will completely prevent us from integrating the system as we proposed. It will not, however, prevent us from delving into the other research directions presented above, and from expanding upon the algorithm independently from a physical implementation.

6 Final Product

As described above, the main product of this project will be a fully automatic system for solving square jigsaw puzzles (see Figure 3). The system will be able to capture the puzzle pieces' data using a standard camera, analyze and apply the puzzle solving algorithm, and physically move the puzzle pieces to their correct location on the work desk.

To complement the system, we intend to create an API to control the Yaskawa Motoman NX100 robotic arm through Matlab, in order to assist both ourselves and future users. This API will leverage the available C++ communications API, but should provide a greater level of abstraction to end users.

Another important product of the project will be various expansions to the puzzle solving algorithm developed in the CS department. These are detailed in Table 2. Since these are open research issues, we cannot provide quantitative estimates at this stage. Note also that we may come across other issues or test other modifications to the algorithm as we go along, and will include any noteworthy data in following reports.

7 Testing Scenarios

Physical system testing will be done on printed images from the already available image sets, particularly from [3]. The images will be printed and glued to some thicker platform, such as cardboard or styrofoam, and cut into square puzzle pieces. The pieces will be presented to the system, which will be required to move them from their initial location to a second part of the work desk, in which the puzzle will be reconstructed.

We hope to test puzzles with 100-200 pieces, with the smallest pieces being about 3x3 cm, according to the limitations detailed above.

Any modification to the algorithm will be thoroughly tested in the same manner as in [11], and evaluated with the three performance metrics defined in the literature review section. Assuming we continue to include a random element in the algorithm, we will run each test 10 times, selecting the solution with the highest "best buddies" estimation metric score.

Tests on large images (thousands of parts) will be timed as well (i.e. how long it takes the algorithm to solve the puzzle), as an additional important metric.

These results will be compared to the current state-of-the-art, [6, 11].

8 Budget Estimate

8.1 Human Resources

Name	Position	Estimated work time [hours]	Payment [$/hour]	Total [$]
Prof. Ohad Ben-Shahar	Adviser	50	100	5,000
Dr. Rami Hagege	Adviser	20	100	2,000
Nadav Erell	Student	750	10	7,500
Roey Nagar	Student	750	10	7,500
	Subtotal			**22,000**
	Overhead 25%			5,500
	Total			**27,500**

8.2 Hardware Equipment

Type	Estimated work time [hours]	Cost [$/hour]	Total [$]
Computer station	1000	2	2,000
	Total		**2,000**

8.3 Special Equipment

Type	Estimated work time [hours]	Cost [$/hour]	Total [$]
Motoman Robotic Arm*	200	10	2,000
Camera (rent)	150	0.5	75
	Total		**2,075**

* The Yaskawa Motoman NX100 robotic arm is already installed in the Computational Vision lab in the CS department at BGU. Therefore, the cost reflects a rough estimate for use of the robotic arm, and not the entire price of the equipment.

8.4 Various Expenses

Type	Cost [$]
Printing and editing services (documents)	100
Office equipment	50
Printing jigsaw puzzles	100
Unpredictable expenses	600
Total	**850**

8.5 Budget Summary

Component	Cost [$]
Human Resources	27,500
Hardware Equipment	2,000
Special Equipment	2,075
Various Expenses	850
15% budget deviation	4,864
Total	**37,289**

9 Schedule

The project is intended to be completed by the end of August 2013. Other important milestones, such as a Progress Report and Poster presentations, are detailed below.

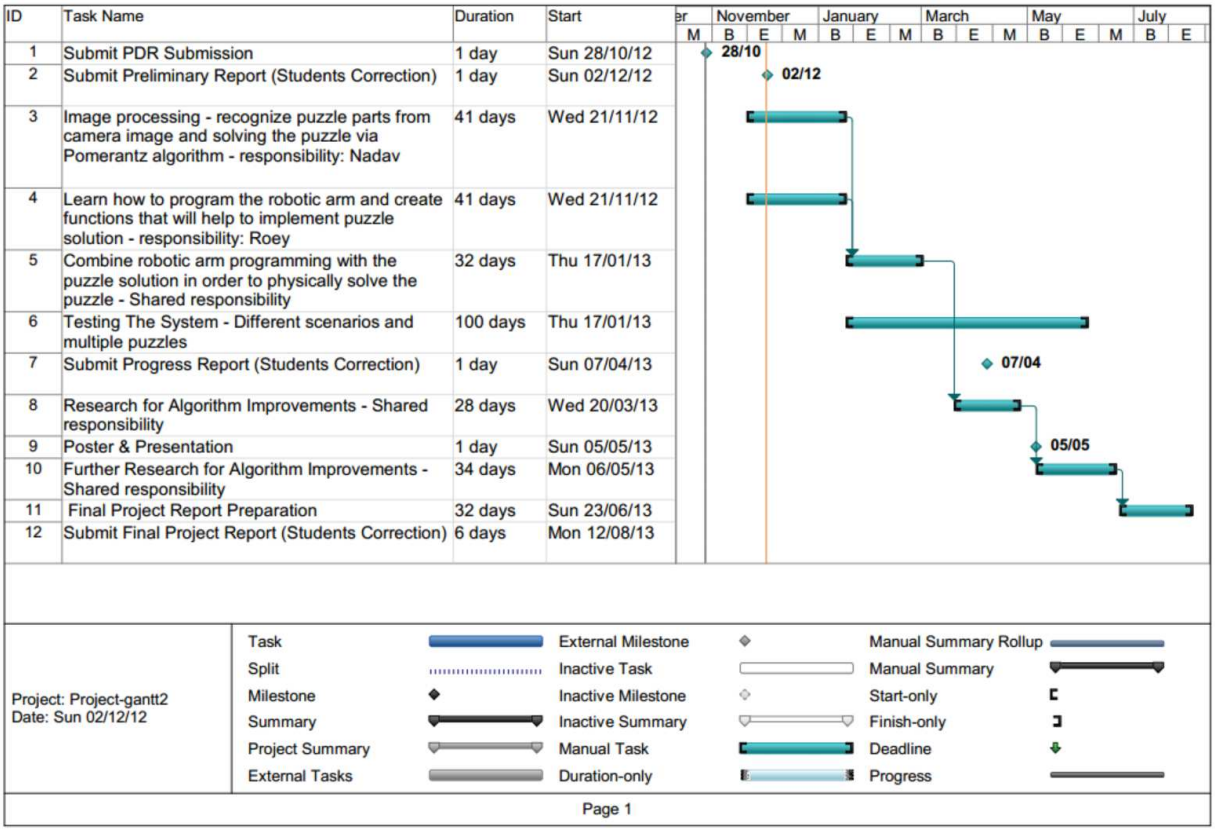

Figure 5: Project Gantt

Table 3 presents the full schedule for report submissions and corrections, as specified by the EE department.

Report name	Date
PDR Submission	28-Oct-2012
PDR-Adviser Correction	28-Oct-2012
PDR-Students Correction	28-Oct-2012
PDR Evaluation	28-Oct-2012
Preliminary Submission	11-Nov-2012
Preliminary-Adviser Correction	18-Nov-2012
Preliminary-Students Correction	02-Dec-2012
Preliminary Evaluation	09-Dec-2012
Progress Submission	17-Mar-2013
Progress-Adviser Correction	20-Mar-2013
Progress-Students Correction	07-Apr-2013
Progress Evaluation	14-Apr-2013
Poster	05-May-2013
Presentation	05-May-2013
Final Submission	04-Aug-2013
Final-Adviser Correction	11-Aug-2013
Final-Students Correction	18-Aug-2013
Final Evaluation	25-Aug-2013

Table 3: Reports submission dates

References

[1] B.J. Brown, C. Toler-Franklin, D. Nehab, M. Burns, D. Dobkin, A. Vlachopoulos, C. Doumas, S. Rusinkiewicz, and T. Weyrich. A system for high-volume acquisition and matching of fresco fragments: Reassembling theran wall paintings. In *ACM Transactions on Graphics (TOG)*, volume 27, page 84. ACM, 2008.

[2] T.S. Cho, S. Avidan, and W.T. Freeman. The patch transform. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 32(8):1489–1501, 2010.

[3] T.S. Cho, S. Avidan, and W.T. Freeman. A probabilistic image jigsaw puzzle solver. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pages 183–190. IEEE, 2010.

[4] E.D. Demaine and M.L. Demaine. Jigsaw puzzles, edge matching, and polyomino packing: Connections and complexity. *Graphs and Combinatorics*, 23:195–208, 2007.

[5] H. Freeman and L. Garder. Apictorial jigsaw puzzles: The computer solution of a problem in pattern recognition. *Electronic Computers, IEEE Transactions on*, (2):118–127, 1964.

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

[7] K. Hori, M. Imai, and T. Ogasawara. Joint detection for potsherds of broken earthenware. In *Computer Vision and Pattern Recognition, 1999. IEEE Computer Society Conference on.*, volume 2. IEEE, 1999.

[8] D. Koller and M. Levoy. Computer-aided reconstruction and new matches in the forma urbis romae. *Bullettino Della Commissione Archeologica Comunale di Roma*, 2, 2006.

[9] H.Y. Lin and W.C. Fan-Chiang. Reconstruction of shredded document based on image feature matching. *Expert Systems with Applications*, 39(3):3324–3332, 2012.

[10] M.A.O. Marques and C.O.A. Freitas. Reconstructing strip-shredded documents using color as feature matching. In *Proceedings of the 2009 ACM symposium on Applied Computing*, pages 893–894. ACM, 2009.

[11] D. Pomeranz, M. Shemesh, and O. Ben-Shahar. A fully automated greedy square jigsaw puzzle solver. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, pages 9–16. IEEE, 2011.

[12] G.M. Radack and N.I. Badler. Jigsaw puzzle matching using a boundary-centered polar encoding. *Computer Graphics and Image Processing*, 19(1):1–17, 1982.

[13] H. Wolfson, E. Schonberg, A. Kalvin, and Y. Lamdan. Solving jigsaw puzzles by computer. *Annals of Operations Research*, 12(1):51–64, 1988.

[14] Y.X. Zhao, M.C. Su, Z.L. Chou, and J. Lee. A puzzle solver and its application in speech descrambling. In *Proc. 2007 WSEAS Int. Conf. Computer Engineering and Applications*, pages 171–176, 2007.

Appendices

1. Experimental results of the greedy puzzle solver

These results are taken directly from the supplemental material provided with the paper by Pomerantz *et al.* [11]. They are included here for quick reference, and give an indication of the algorithm's current performance.

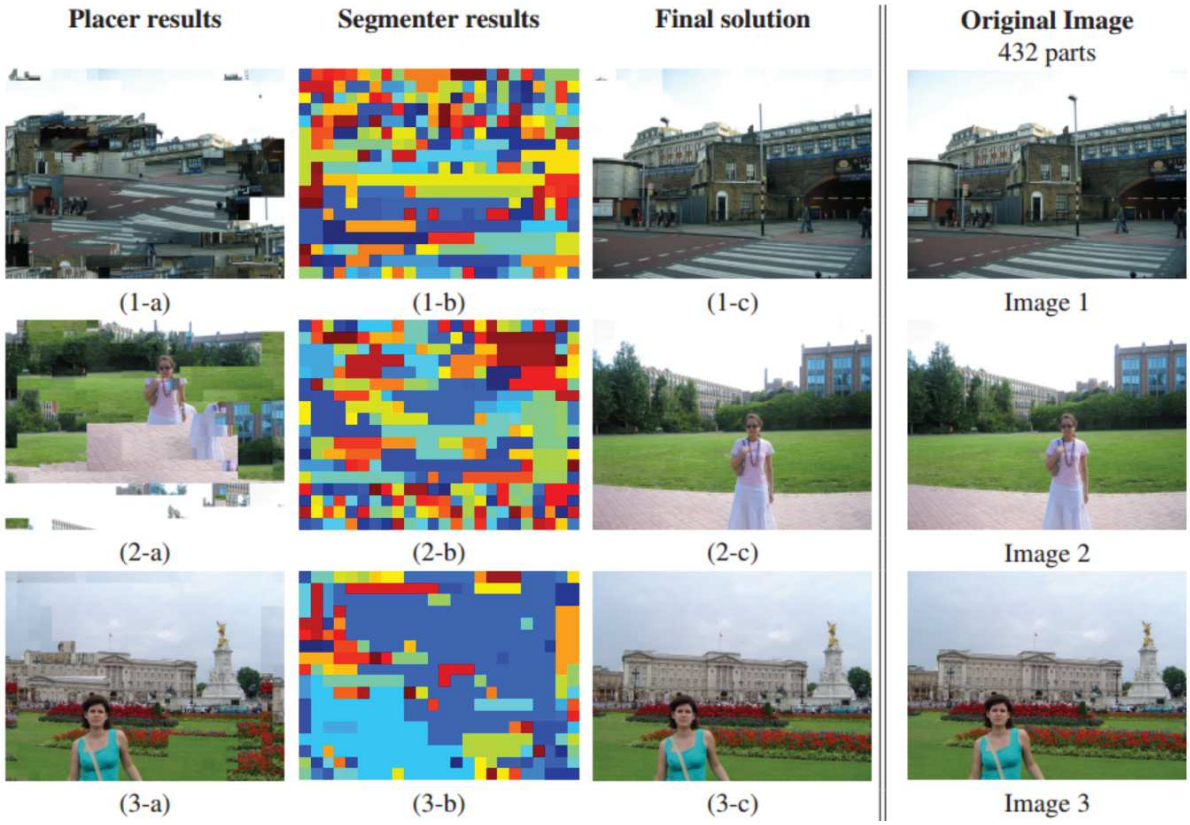

Figure 6: Results of our solver with the same images used by Cho *et al.* [3]. For each image i (as numbered in the original collection), panel $i{-}a$ shows the initial placement results of the greedy placer. Panel $i{-}b$ shows the segments of panels $i{-}a$, chosen according to the best buddies criterion and depicted by different colors. Panel $i{-}c$ shows the final result of the shifter. Each of the above is the best result chosen by the solver (out of the 10 runs with random seeds).
The performance metrics accuracy results under the Direct comparison and under the Neighbor comparison for each image are:
Image 1: direct 77%, neighbor: 80%.
Image 2: direct 82%, neighbor: 81%.
Image 3: direct 100%, neighbor: 100%.

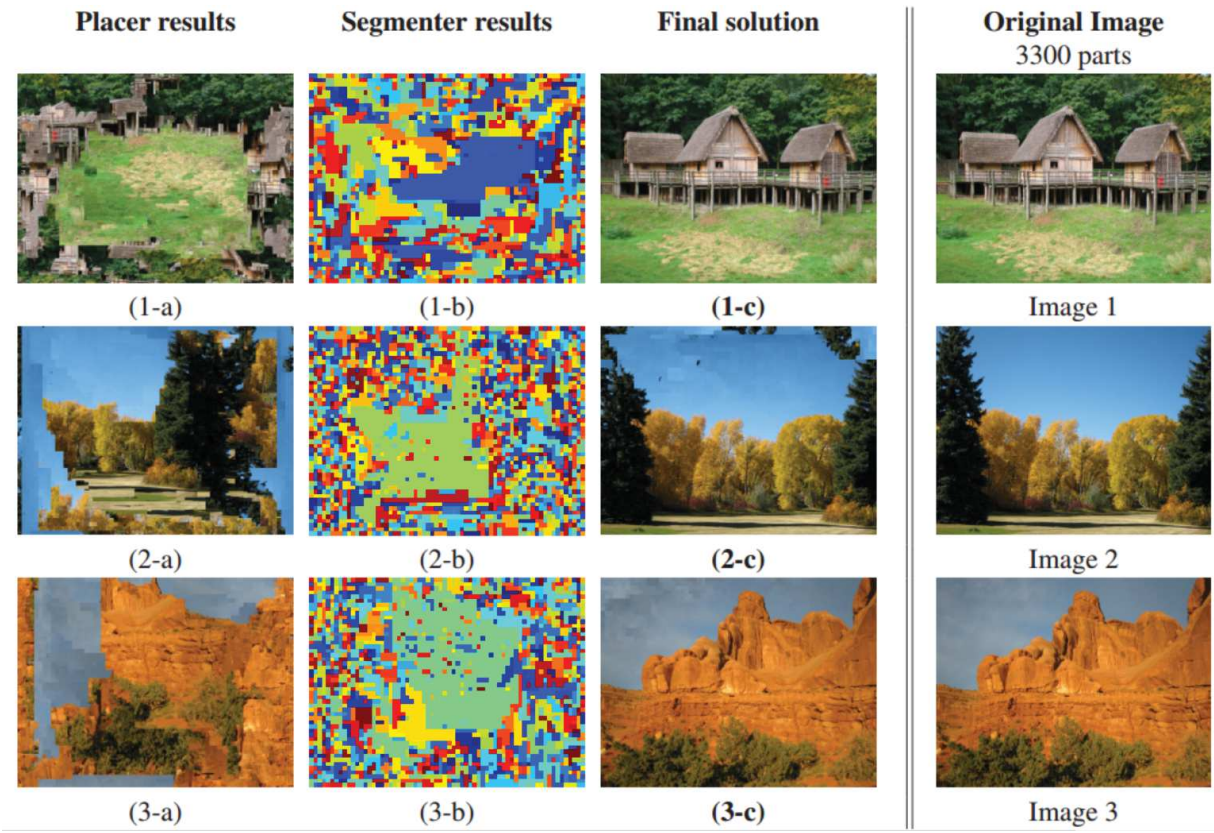

Figure 7: Three solutions for puzzles consist of 3300 parts, (each 28 $\times$ 28 pixels in size). For each image i (as numbered in the original collection), panel $i{-}a$ shows the initial placement results of the greedy placer. Panel $i{-}b$ shows the segments of panels $i{-}a$, chosen according to the best buddies criterion and depicted by different colors. Panel $i{-}c$ shows the final result of the shifter. Each of the above is the best result chosen by the solver (out of the 10 runs with random seeds).
The performance metrics accuracy results under the Direct comparison and under the Neighbor comparison for each image are:
Image 1: direct 100%, neighbor: 100%.
Image 2: direct 56%, neighbor: 65%.
Image 3: direct 86%, neighbor: 90%.