BEN-GURION UNIVERSITY OF THE NEGEV

THE FACULTY OF NATURAL SCIENCES

DEPARTMENT OF COMPUTER SCIENCE

SOLVING THE SQUARE JIGSAW PROBLEM

THESIS SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS

FOR THE MASTER OF SCIENCES DEGREE

DOLEV POMERANZ

UNDER THE SUPERVISION OF PROF. OHAD BEN-SHAHAR

OCTOBER 2012

BEN-GURION UNIVERSITY OF THE NEGEV

THE FACULTY OF NATURAL SCIENCES

DEPARTMENT OF COMPUTER SCIENCE

SOLVING THE SQUARE JIGSAW PROBLEM

THESIS SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS

FOR THE MASTER OF SCIENCES DEGREE

DOLEV POMERANZ

UNDER THE SUPERVISION OF PROF. OHAD BEN-SHAHAR

Signature of student: _____ Date: _____

Signature of supervisor: _____ Date: _____

Signature of chairperson

Of the committee for graduate studies: _____ Date: _____

OCTOBER 2012

Abstract

In the square jigsaw puzzle problem one is required to reconstruct the complete image from a set of non-overlapping, unordered, square puzzle parts. Here we propose a fully automatic solver for this problem, where unlike some previous work, it assumes *no* clues regarding parts' location and requires no prior knowledge about the original image or its simplified (e.g., lower resolution) versions. To do so, we introduce a greedy solver which combines both informed piece placement and rearrangement of puzzle segments to find the final solution. Among our other contributions are new compatibility metrics which better predict the chances of two given parts to be neighbors, and a novel estimation measure which evaluates the quality of puzzle solutions without the need for ground-truth information. Incorporating these contributions, our approach facilitates solutions that surpass state-of-the-art solvers on puzzles of larger sizes than ever before attempted.

Acknowledgments

I wish to thank my advisor, Prof. Ohad Ben-Shahar, for his guidance and support throughout my studies. He is always patient and understanding, which made me appreciate him even more.

I would like to thank my family, especially Karen my wife, for their endless support.

A special thank you is dedicated to Michal Shemesh who had an active part in this research. Michal, working with you was always a pleasure. Thank you for your insightful thoughts, and good luck with your Ph.D.

Finally, I wish to thank my friends and colleagues from the Interdisciplinary Computational Vision Lab (ICVL) at Ben-Gurion University: Michal Shemesh, Yair Adato, Ilan Kadar, Alik Mokeichev, Guy Ben-Yosef, Liana Diesendruck, Rotem Mairon, Ehud Barnea, and Boaz Arad. You are all good friends from whom I have learned a lot. Thank you for making these past years an enjoyable experience.

Contents

List of Figures

Chapter 1

Introduction

1.1 The Problem

Jigsaw puzzles first appeared during the 1760's as an educational game for children [31]. John Spilsbury, an engraver and map maker from London [31], attached a map to a wooden board. Using a saw, he divided the map into countries, enabling children to learn geography through playing. Today, jigsaw puzzles continue to challenge people, and in recent years computers as well.

Given n jigsaw puzzle parts of an object, reconstruct the complete object. This simple definition holds the essence of the jigsaw puzzle problem. Though it sounds simple, Demaine *et al.* [8] proved that this problem is NP-complete. There exists many variations that fine tune this definition. Each has different attributes that researchers exploit in their efforts for creating better jigsaw solvers. This diversity in variations lead to many solvers which are not easily comparable. However, there are several metrics which scientists use to differentiate between related work, usually awarding the number of parts as the most crucial attribute.

1.2 Applications

A most common question regarding the scientific research on jigsaw puzzles is: "What is it good for?". In this section we will address this question by discussing several applications which derive common tools and methods from the jigsaw puzzles research. That said, it is also important to note that the captivating character of the jigsaw problem, which sparked the imagination of so many throughout centuries, is, from our point of view, a sufficient motivation for researching the problem (as previously noted by [12]).

Although phrased as a game, this problem serves a platform for many applications. Many similar characteristics can be found in the archaeological [3, 14, 22] (e.g., Fig. 1.1), and torn and shredded documents contexts [16]. Usually, torn documents or even some broken pottery are required to be reconstructed out of many given pieces. The enormous diversity of such reconstruction problems (e.g., 2D vs. 3D) led to many different algorithms and approaches [13]. One example of the popularity of the torn and shredded documents could be associated with the DARPA Shredder Challenge [7] in which a prize was rewarded to the first team that successfully reconstructed 5 torn documents.

Other than those mainstream fields, similarities to the jigsaw problem could also be found in other more surprising fields. Cho *et al.* [4] showed the connection to image editing. They introduce the *patch transform* which enables operations like: image reorganization, object removal, image retargeting, photomontage, and more. For example, setting a constraint which marks a region that encapsulates an object to be removed, will result with using existing patches from other parts of the image to fill that gap. Since it uses existing patches this method is fast and

Figure 1.1: Reconstructed wall painting (image taken from [22])

usually achieves appealing results (see Fig. 1.2). When no constraints are imposed on the image the patch transform reduces to solving a jigsaw puzzle.

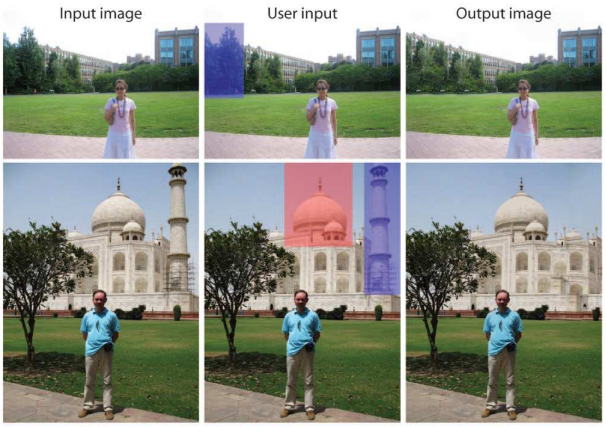


Figure 1.2: Two object removal examples. The inverse patch transform reconstructs an image discarding the user-specified patches, red areas are fixed while blue areas are marked to be removed (image taken from [4])

Another application which comes from the image editing domain is image alignment and mosaicing. Given several images we wish to compose them into a panoramic mosaic of a scene [28]. The images usually overlap, however, in case

they do not Poleg *et al.* [24] commented that jigsaw puzzles are a relatively simple set of instances in which the relative placements, the gap widths, and the canvas size are all known.

Zhao *et al.* [35] showed that jigsaw puzzles are also related to speech descrambling. An analog scrambler scrambles a speech signal to prevent eavesdroppers from gaining secret information. This can be done in either the time domain or the frequency domain. The resulting spectrogram is in fact a permutation of square patches to the original spectrogram, since the first axis represents time and the second represents frequency (see Fig. 1.3). Thus, a jigsaw solver actually performs as an analog speech descrambler.

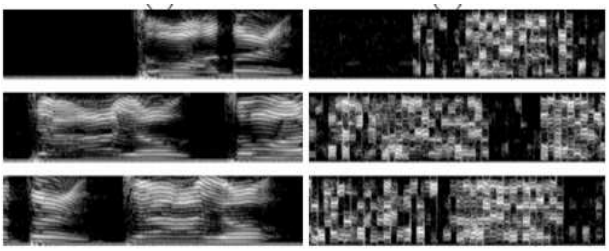


Figure 1.3: Two sound signal spectrograms. The left is the original signal's spectrogram, while the right belongs to the scrambled signal (image taken from [35])

The image reassembly problem which is the problem of restoring deleted files from a scattered set of fragments is also related to the jigsaw problem [18]. Here, fragments of memory holding pieces of different images are to be composed back to the original images. Deleting a file actually manipulates the file system's data structures. Since files are likely to be scattered rather than continuous, these data structures point to different locations in the disk. Loosing the file's meta data, meant we actually lost the order in which the fragments should reside in the file and the affiliation of each fragment to its file.

This variety in applications further supports the importance of the jigsaw problem. Researching new and improved algorithms that are robust, accurate, and efficient would certainly (and have already) aid in those neighboring fields.

1.3 Related Work

The traditional jigsaw puzzle problem assumes pieces of various shapes, and indeed earlier computational work considered the problem in this form. The first jigsaw solver proposed by Freeman and Garder in 1964 [9] was designed to solve "apictorial" puzzles (i.e., the only available information is the shape of the pieces) and was able to handle nine-piece problems. By the end of the century, shaped-based solvers were already able to reconstruct puzzles of more than 200 pieces [12]. The use of appearance and chromatic information began only three decades after the work of Freeman and Garder (e.g., [15, 6, 30, 34, 32, 17, 20]).

A major building block in most appearance-based puzzle solvers evaluate the similarity between two parts, and the strategies for doing so are typically divided into two groups. One approach compares the appearance of the abutting boundaries while using a formal distance measure between these two vectors (e.g., [15, 6, 30, 34, 17, 20]). Alternatively, it was suggested to use the entire part and measure its statistical properties in order to group similar parts together [20] or as a means to define pairwise similarity measures [15].

Once the similarity between two parts is estimated, different classes of algorithms are employed to solve the puzzle. These algorithms are affected mostly by their specific problem variation. In this thesis we are interested in the square jigsaw puzzle variation, in which we have only the pieces' texture while their shape

is largely irrelevant. In this variation, pieces, also known as patches are square. Thus, their shape gives us no useful information in obtaining the desired solution. Given n square patches where each patch orientation is known and the puzzle's size is known, there are $n!$ possible arrangements. This implies that solving square puzzles even with small n values might be challenging.

To the best of our knowledge, the first work addressing this variation was done by Toyama *et al.* [30] in 2002. They considered a square jigsaw problem where images have a binary pixel value, and in which the patches' orientation is known. They proposed a genetic algorithm that tries to minimize the total difference of neighboring border pixels. To do so, they interestingly define the genetic elements such as: fitness function, crossover operators, and selection operator. Their algorithm could successfully handle 8×8 patch size puzzles, where each patch contains 62×62 pixels.

Work which followed [30] addressed color images. These unlike binary images, had much more information and thus enabled solving much bigger puzzles. Zhao *et al.* [35] proposed an ant colony optimization algorithm (ACO) on color images. This probabilistic technique, was tested on small puzzles of up to 4×8 patches.

Murakami *et al.* [19] showed in 2008 that their algorithm can solve 16×12 (192) patches puzzles, where each patch is of 80×80 pixels size. Their approach generates blocks of patches in different sizes and shapes, and then aspires to generate bigger blocks by connecting smaller blocks. They continue until reaching one block, which is their output solution.

In 2008 another work handled square patches. Nielsen *et al.* [20] introduced the "singlepiece" algorithm. It assumed the border pieces are known (e.g., are

given by an oracle), and starts with solving the puzzle's border. For that purpose they employ the assignment problem heuristic. Once the puzzle's frame is set they use a greedy algorithm to fill up the frame. Applying their algorithm, they were able to accurately solve a puzzle of 16 (320) patches.

An approach introduced by Alajlan [1] in 2009 solves gray scale images. It is essentially a greedy algorithm. However, in each greedy step the well-known Hungarian procedure is employed to determine all the neighbors of a selected patch. The algorithm manages to correctly assemble puzzles up to 8×8 patches.

Recently, Cho *et al.* [5] presented a probabilistic solver which achieves approximated puzzle reconstruction via a graphical model and a probability function that is maximized via loopy belief propagation. Since they lack a local evidence term required for their graphical model, they employed two strategies that exploit prior knowledge - either a dense-and-noisy evidence which estimates the low resolution image from a bag of parts, or a sparse-and-accurate evidence which assumes that a few parts, called "anchor patches", are given by an oracle (e.g., a human observer) at their correct location in the puzzle. While only semi-automatic, their approach was seminal in its ability to handle puzzles with over 400 pieces.

The 2011 work by Yang *et al.* [33] also formulates the problem as maximizing a label probability function. Generalizing the known particle filter inference framework, they are able to improve Cho *et al.* [5] results on puzzles of 108 patches, where each patch is 56×56 pixels. However, their algorithm requires one anchor patch to be initialized in its correct location. Specifically, the top left corner patch.

1.4 Overview and Contributions

In this thesis we challenge the state-of-the-art in jigsaw puzzle solving in several ways. We suggest a computational framework that can handle within reasonable time square jigsaw puzzles of a size larger than ever before attempted. However, we completely exclude the use of clues, oracles, or human intervention, as well as any use of prior knowledge about the original image or simplified (e.g., lower resolution) versions of it. Despite these significant restrictions, our approach achieves better than state-of-the-art performance, and frequently succeeds in providing completely accurate puzzle solutions.

Our puzzle solving framework is based on a greedy solver, which works in several phases. First a *compatibility function* is computed to measure the affinity between two neighboring parts (as often done in other solvers as well). Then the solver solves three sub-problems: the placement problem, the segmentation problem, and the shifting problem. The placement module places all parts on the board in an informed fashion, the segmentation module identifies regions which are likely to be assembled correctly, and the shifting module relocates regions and parts to produce the final result.

Our main contribution is a fully automated solver which uses no clues, hints, or other prior knowledge whatsoever. In addition, our contributions also include the proposal of new and better compatibility metrics, as well as the introduction of the concept of an *estimation metric* which evaluates the quality of a given solution without any reference to the original, ground-truth image. As we show, these metrics become a critical tool that facilitates self-evaluation when no clues or prior knowledge are available. We also introduce a technique which *globally*

improves a given compatibility metric, thus resulting with state-of-the-art compatibility results. In what follows we first discuss these, as well as the rest of the metrics involved in the solver.

Chapter 2

Metrics

2.1 Compatibility Metrics

A compatibility metric is at the foundation of every jigsaw solver. Given two puzzle parts x_i, x_j and a possible spatial relation R between them, the compatibility function predicts the likelihood that these two parts are indeed placed as neighbors with relation R in the correct solution. With $R \in \{l, r, u, d\}$, we use $C(x_i, x_j, R)$ to denote the compatibility that part x_j is placed on either of the left, right, up, or down side of part x_i, respectively.

Optimally, one would wish to find a metric C such that $C(x_i, x_j, R) = 1$ *iff* part x_j is located to the R side of x_i in the original image and 0 otherwise. If such a function exists, the jigsaw puzzle problem could be solved in polynomial time by a greedy algorithm [8]. Motivated by the above, we first discuss how one can measure the accuracy of a compatibility function and then seek a compatibility function C which is as accurate and discriminative as possible.

2.1.1 Measuring Compatibility Accuracy

In order to compare between compatibility metrics, we denote the classification criterion as the ratio between correct placements and the total number of possible placements. Similar to Cho *et al.* [5], we define the *correct placement of parts* x_i *and* x_j *according to the compatibility metric* C if part x_j is in relation R to part x_i in the original image and if

$$\forall x_k \in Parts, \quad C(x_i, x_j, R) \geq C(x_i, x_k, R) \ . \tag{2.1}$$

Recently, Cho *et al.* [5] evaluated five compatibility metrics, among which the *dissimilarity-based* compatibility metric showed to be the most discriminative. Inspired by both the dissimilarity-based metric and the characteristics of natural images, we propose two new types of compatibility metrics, which will be described shortly after we review the dissimilarity-based compatibility metric.

2.1.2 Dissimilarity-based Compatibility

The dissimilarity between two parts x_i, x_j can be measured by summing up the squared color differences of the pixels along the parts' abutting boundaries [5]. For example, if we represent each color image part in normalized Lab space by a $K \times K \times 3$ matrix (where *K* is the part width/height in pixels) then the dissimilarity between parts x_i and x_j, where x_j is to the right of x_i, can be defined as

$$D(x_i, x_j, r) = \sum_{k=1}^{K} \sum_{d=1}^{3} (x_i(k, K, d) - x_j(k, 1, d))^2 \ . \tag{2.2}$$

We do emphasize that dissimilarity is not a distance measure, i.e., $D(x_j, x_i, R)$ is not necessarily the same as (and almost always different than) $D(x_i, x_j, R)$.

Although the dissimilarity-based metric was shown to be the most discriminative among the tested metrics in Cho *et al.* [5], the observation that it is related to the L_2 norm of the boundaries' difference vector suggests that other norms of the same difference vector could behave even better. Inspired by the use of non Euclidean norms in image operations such as noise removal [27] or diffusion [29], and by the observation that the L_2 norm penalizes large boundary differences very severely even though such large differences *do* exist in natural images, we evaluated other L_p norms as well. The average accuracy empirical results for the dissimilarity metric based on various L_p norms are shown in Fig. 2.1 and indicate clearly that the L_2 is suboptimal, and that the best results may be obtained with L_p norms with $p \approx 0.3$. It would be interesting to know why does the L_p norm accuracy behave as a concave function, and why does $p \approx 0.3$ appear to provide optimal performance.

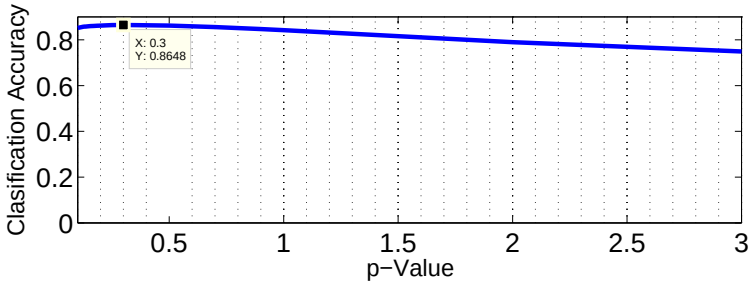


Figure 2.1: Comparing dissimilarity-based metrics using different L_p norms. For this test we used a database of 20 test images [5], and analyzed each for the portion of correct part pairs that received the highest compatibility score among other candidates. The plot depicts the average classification accuracy of 20 images and shows how performance peaks at $p \approx 0.3$, with a value of 86%.

2.1.3 $(L_p)^q$ Compatibility

As mentioned before, the optimal compatibility function should also be as discriminative as possible. In order to reflect the scattering of the dissimilarity values when obtaining their compatibility measures,we also experimented with different powers q of the L_p norms. For example, the $(L_p)^q$ compatibility for parts x_i, x_j where x_j is to the right of x_i is defined as:

$$D_{p,q}(x_i, x_j, r) = \left(\sum_{k=1}^{K} \sum_{d=1}^{3} (|x_i(k, K, d) - x_j(k, 1, d)|)^p \right)^{\frac{q}{p}} . \tag{2.3}$$

Having these dissimilarity values, we also propose to obtain its compatibility measure slightly differently than done previously. In particular, we define

$$C(x_i, x_j, R) \propto \exp\left(- \frac{D_{p,q}(x_i, x_j, R)}{quartile(i, R)} \right) , \tag{2.4}$$

where *quartile(i,R)* is the quartile of the dissimilarity between all other parts in relation R to part x_i. The quartile normalization gives us valuable information about the scattering of the compatibility function values, and in particular emphasizes discriminative scattering.

Although the value of q does not affect the dissimilarity classification accuracy, it does have a significant effect on our solver's performance. While Cho *et al.* [5] used $p = 2$ and $q = 2$ in their dissimilarity metric (Eq. 2.2), we found that optimal results are achieved with power value $q = 1/16$ and hence prefer to use a $(L_{p=3/10})^{q=1/16}$, i.e.,

$$D_{p,q}(x_i, x_j, r) = \left(\sum_{k=1}^{K} \sum_{d=1}^{3} (|x_i(k, K, d) - x_j(k, 1, d)|)^{\frac{3}{10}} \right)^{\frac{5}{24}} . \tag{2.5}$$

2.1.4 Prediction-based Compatibility

While dissimilarity-based metrics may be improved greatly by employing $(L_p)^q$ affinities, other possibilities may be promising as well. In particular, as opposed to measure differences, one may attempt to quantify how well one can *predict* the boundary content of one part based on the boundary content of the other. The better the prediction, the higher the compatibility of the two parts, a measure we call a *Prediction-based Compatibility*.

Naturally, predictions over the boundaries can be made using Taylor's expansion. A first order prediction would need an estimation of the derivative of each part at its boundary, a computation that can be done numerically in one of several ways. For example, employing backward differences estimation using the last two pixels in each row near the boundary, one can obtain a prediction of the first pixel in the neighboring part. The quality of this prediction can then be verified against the actual pixel value from the second part by using any desired norm, and in general, using $(L_p)^q$ for preselected p and q as discussed above in the dissimilarity measure. Importantly, we repeat this computation for both parts in a symmetric fashion. For example, the prediction metric for two parts x_i, x_j with relation r when using $(L_p)^q$ with $p = 3/10$ and $q = 1/16$ can be denoted by the following expression, and obtain its compatibility measure as in Eq. 2.4:

$$\begin{aligned} &Pred(x_i, x_j, r) = \textstyle\sum_{k=1}^{K} \sum_{d=1}^{3} \\ &\Big[([2x_i(k, K, d) - x_i(k, K-1, d)] - x_j(k, 1, d))^{\frac{3}{10}} + \\ &\quad ([2x_j(k, 1, d) - x_j(k, 2, d)] - x_i(k, K, d))^{\frac{3}{10}} \Big]^{\frac{5}{24}} \end{aligned} \tag{2.6}$$

2.2 Comparison of Compatibility Metrics

Fig. 2.2 shows the comparison between compatibility metrics' accuracy on three different databases. Note that even though dissimilarity-based and $(L_p)^q$ metrics are directly related by definition, $(L_{3/10})^{1/16}$ has achieved a significant improvement of 7% over the dissimilarity metric (86% vs. 79%). As shown, the prediction-based compatibility performs very similarly, but further evaluation on additional puzzle databases (see Sec 4) clearly indicates its superiority.

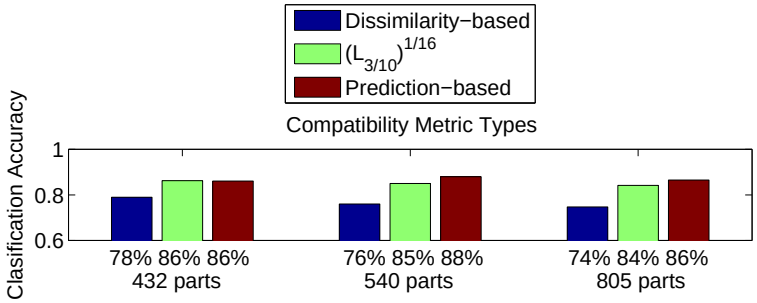


Figure 2.2: Comparing compatibility metrics' accuracy. For each image in the test sets consisting of 20 images each, we find the portion of correct part pairs that received the highest compatibility score among other candidates. We show the average classification accuracy of 20 images for each test set. Note how the $(L_{3/10})^{1/16}$ and the prediction-based compatibilities perform better than the dissimilarity-based compatibility in all three sets.

Before we proceed to describe our jigsaw solver, we discuss two additional types of metrics. The first, the *performance metric*, evaluates the performance of an approximated solution by comparison to the ground truth [5]. The second, a novel metric which we term the *estimation metric*, evaluates the quality of an approximated solution *without* prior knowledge of the original image. The mo-

tivation for this kind of metric is driven from the need to assess the quality of an approximated solution in the process of computing it, when no access to a ground-truth solution is available.

2.3 Performance Metrics

Performance metrics are used to evaluate the performance accuracy of an approximated solution by comparison to the correct (i.e., desired) solution. Cho *et al.* [5] introduced three performance metrics, two of which are relevant to our algorithm and are reviewed below.

Direct Comparison Metric: The direct comparison metric calculates the ratio between the number of parts in the approximated solution which are placed in their correct location and the total number of parts [5].

Neighbor Comparison Metric: The neighbor comparison metric calculates the ratio between the number of correct neighbors placement for each part and the total number of neighbors [5].

By construction, the direct comparison metric does not tolerate parts which are not placed in their correct location, and hence grants lower scores even to solutions that are assembled correctly but are slightly shifted. Performance metrics' accuracy results of Cho *et al.* and our solver are shown in Fig. 2.3.

2.4 Estimation Metrics

While the performance metrics evaluate the quality of solutions in the presence of a ground truth image, the *estimation metric* is designed to provide such quality

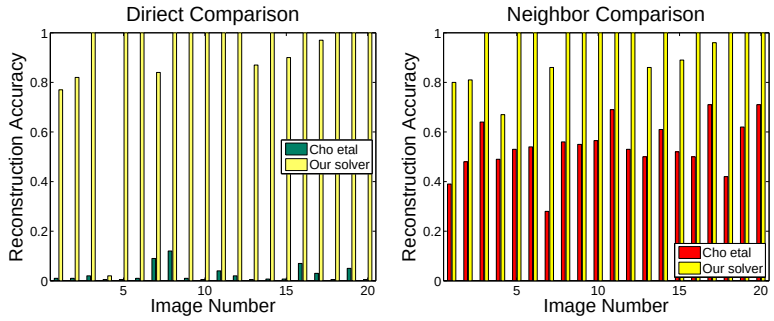


Figure 2.3: Performance metrics' accuracy comparison. We compare our results with those reported by Cho *et al.* [5] in both the direct measure (upper graph) and neighbor measure (lower graph). Under the direct comparison measure Cho *et al.* average reconstruction accuracy is lower than 10% while ours is 94%. Under the neighbor comparison measure Cho *et al.* average reconstruction accuracy is 55% while ours is 95%.

scores without knowing the correct solution whatsoever. Given the fact that we use no clues or prior information about the original image, this is an essential tool which can be used during the reconstruction process in order to improve the solver's final approximated solution. In this thesis we introduce this notion and suggest one estimation metric that is put to use in our proposed puzzle solver.

The "Best Buddies" metric: Two parts x_i, x_j, their relation R_1 and opposite relation R_2 are said to be *best buddies iff* the following holds:

$$\forall x_k \in Parts, \quad C(x_i, x_j, R_1) \geq C(x_i, x_k, R_1)$$

$$\text{and} \tag{2.7}$$

$$\forall x_p \in Parts, \quad C(x_j, x_i, R_2) \geq C(x_j, x_p, R_2)\ .$$

Intuitively, two parts are best buddies if both "agree" that the other part is their

most likely neighbor in a certain spatial relation. The best buddies *metric* for a given approximated solution represents the ratio between the number of neighbors who are said to be best buddies and the total number of neighbors.

Testing the correlation between the best buddies estimation metric and the performance metrics reveals no correlation to the direct performance metric (which is not based on pairwise similarities) but a strong correlation to the neighbor performance metric. Fig. 2.4 depicts this relationship as a scatter plot of the best buddies estimation metric vs. the neighbor performance metric. Every dot in the graph represents a single puzzle solution generated by our solver (our solver is discussed in Section 3), where the x-coordinates and the y-coordinate denote the performance accuracy and estimation accuracy, respectively. Each image in the set was tested 10 times, totaling 200 sample points.

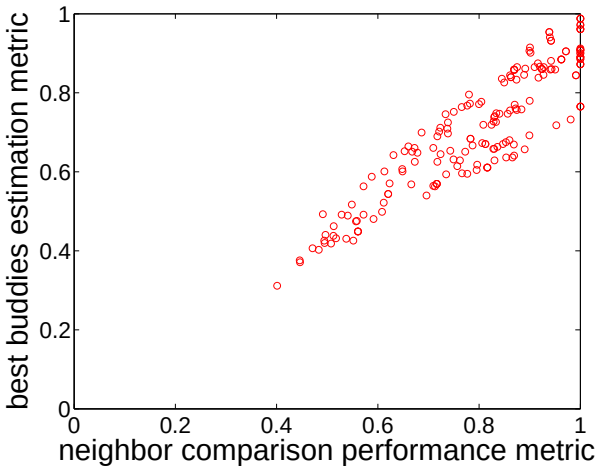


Figure 2.4: Correlation between the best buddies estimation metric and the neighbor comparison performance metric. Every dot in this 200 point graph represents a single puzzle solver solution, where the x and y coordinates denote its performance value and estimation value, respectively.

2.5 Global Improvements to Compatibility Metrics

To the best of our knowledge, all previous solvers compute a compatibility metric and then directly proceed with solving the puzzle. They do so knowing the compatibility metric probably contains errors, and thus try to avoid them in their solver's pursuit toward an accurate reconstruction. We ask, can those errors be detected prior to the solver's activation? Can some of these errors be detected using the compatibility metric alone? In this thesis we introduce a novel approach of detecting and fixing errors in a given compatibility metric, based solely on the compatibility metric values. Thus, hopefully, raising its accuracy performance. We refer to this method as a global method, since it uses all of the compatibility metric information to overcome local mistakes.

While there could be many ways in which one could try to cope with such a challenge, our method is inspired by the *Stable marriage* problem, and the Gale-Shapley algorithm [10] which addresses it. First, let us recap the definitions of the stable marriage problem. We are given a set of n men denoted by M, and a set of n women denoted by W, where each member of those sets had strictly ranked each member of the opposite sex with respect to her or his marriage preferences.

Definition 1 A match of men to women will be called *unstable* if there are two men a and b who are assigned to women A and B, respectively, although b prefers A to B and A prefers b to a. Otherwise, if there are none, the match will be called *stable*.

Given an instance of the problem, one is required to find a stable match. D. Gale and L. S. Shapley [10] proved that according to these definitions the following holds:

Theorem 1 There always exists a stable set of marriages.

And since there could be several stable matches to a given problem instance we wish to define a criterion for a preferred (i.e., optimal) stable match.

Definition 2 A stable match is said to be *optimal* for set A, if each person in A gets the highest match possible among all other stable matches.

Gale and Shapley provide an algorithm (a.k.a., the Gale-Shapley algorithm) that returns the optimal match for either men or women (depending which group is set as *suitors*). It can also be shown that if the optimal match for men is identical to the optimal match for women, then that match is the only stable match which exists for that particular instance of the problem.

Using this theory, we can now suggest a method in which we could detect errors in any compatibility metric. In our method, given a compatibility metric that was applied on the input puzzle's parts, we generate two instances of the stable marriage problem. The first is the *left-right* instance and the second is the *up-down* instance. In each of them, both the M and W sets contain all of the puzzle's parts, and matching preferences to all other parts are determined by the compatibility metric values. The compatibility metric values are ordered from highest to lowest, and this order is used to determine the preferences. More specifically, in the left-right instance, we set the preferences of part x_i from the men set M to be the ordered list of parts that could be attached to the *left* of it. For part x_i from the women set W, we set the preferences to be the ordered list of parts that could be attached to the *right* of it. The up-down instance is defined in a similar manner. We note that in these newly generated problem instances each part appears in both the men and women sets. This might lead to a stable match in which a part is

matched to itself. Clearly, this a mistake since a part could not neighbor itself in any puzzle. To avoid this obstacle we set a part preference to itself as the lowest among other preferences.

Applying the stable marriage algorithm on those sets we get for each part its matched neighbor in each direction. Thus, we can now define the accuracy of the proposed method as the percentage of correct matches. Testing this novel method on several compatibility metrics yields promising results, as can be seen in Figure 2.5.

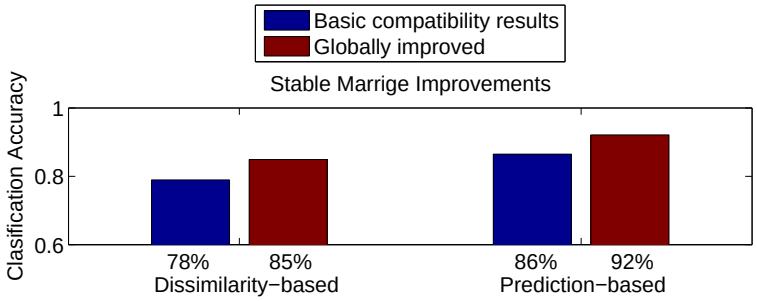


Figure 2.5: Globally improved compatibility metrics' accuracy. We tested two compatibility metrics on Cho's *et al.* [5] image database of 20 images. Each metric was globally improved using our proposed method. Note the substantial improvement for both metrics.

The method was tested on 20 images from Cho's *et al.* [5] database. Surprisingly, for each of the tested images, we found that the optimal match for men was always identical to the optimal match for women. This is true for both the left-right and up-down instances, and for the dissimilarity and prediction based compatibility metrics as well. Thus, as we mentioned before, we conclude for those examples that there is only one stable match. This means that we were able to achieve an optimal match.

Chapter 3

Our Solver

Equipped with the different types of metrics discussed above, we are now ready to describe our puzzle solver. At its higher level, this solver divides the puzzle reconstruction problem into three sub-problems, and solves each of them separately.

- 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.

In what follows we describe algorithms for all these problems.

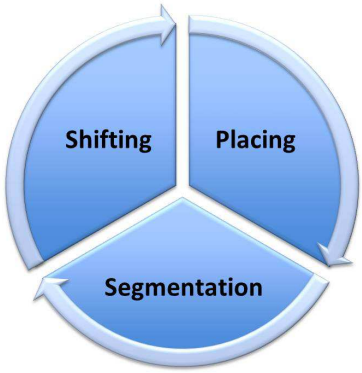


Figure 3.1: The algorithm's overview scheme. Iteratively solving the three sub-problems.

3.1 The Placer

As the first solver module, the *placer* is a greedy algorithm that is given a single part or a partially constructed puzzle, and places the remaining parts on the board in an informed fashion. More specifically, given a greedy criterion and a "seed" part (or a partial constructed seed puzzle), the placer reconstructs the remaining puzzle parts around the seed by iteratively applying the greedy criterion. Such greedy criteria are discussed below.

Technically, the greedy procedure is affected by the board's dimensions such that piece placement is not allowed to exceed the puzzle's width or height which are assumed given. Furthermore, our greedy criteria are also endowed with the heuristics that *empty* puzzle slots with a larger number of *occupied* neighboring slots (i.e., slots in which parts had been allocated), are more informative, and hence should be allocated a puzzle part sooner than others. Note that using this heuristic while starting from a single seed part limits empty slots to have at most two occupied neighbor slots, and hence that the evolving constructed puzzle will

preserve a roughly rectangular shape. If the seed is a partially constructed puzzle, the placer would first complete it to a rectangular shape by placing new parts in empty slots with more than two occupied neighbor slots.

What greedy criteria can one use for placing parts in slots? Given the potential of the best buddies estimation metric (see Sec. 2.4), we propose to use it as a greedy placement criterion as well. By using it, in each step the placer chooses the empty slot and unallocated part which "agree" to be best buddies among all empty slots and remaining unallocated parts. If there is more than one possibility, or none at all, the algorithm then chooses the part with the highest compatibility metric to its candidate neighbors[1].

The results of using the best buddies metric as a greedy criterion for the placer is shown in Figs. 4.1a and 4.4a and clearly demonstrate that placements tend to contain regions which are assembled correctly, but are often misplaced relative to each other. This is expected, since a compatibility function can only predict the likelihood of immediate neighbors, rather than global arrangements or true locations on the puzzle. This is also the motivation for the two subsequent phases of our solver, as described next.

Another possible greedy criterion relies upon the stable marriages data structures which were discussed in Section 2.5. In each greedy step we place the *best* possible match out of the left-right and up-down marriages. The best possible match could be interpreted in one of several ways. For example, since each part ranked all other parts from 1 (the most preferred) to n (the least preferred),we interpret it as the match with the lowest average between the matched pair pref-

[1]In this case, if a candidate part is considered to an empty slot with more than one allocated neighbor, its compatibility is computed by averaging its individual compatibilities to each of these neighbors.

erences of each other (e.g., a best buddies matched pair which ranked each other as first, would have the lowest possible average of 1). If there are no matches to select from in a given greedy step, then we revert to the previously discussed greedy criterion.

Before we move to do so, one more comment is in place. As opposed to previous work, our placement procedure is guaranteed to place each puzzle part exactly once, rather than more than once or even not at all, as could happen in Cho *et al.* [5]. This is due to the fact that in their graphical model the exclusion term in the maximized probability discourages parts from being used more than once, but does not prevent that entirely, therefore allowing the repetition or the complete dropping of some of the parts.

3.2 The Segmenter

Given a complete placement of all parts by the placer, we now look for regions which are assembled correctly, which we denote as *segments*. Clearly, although this would be easier if the ground truth solution image was available, here we care to consider again the stricter version of the problem where no such solution is available for inspection. Under such conditions we attempt to find these segments using a region growing segmentation algorithm [23] with random seeds and a segmentation predicate based on the best buddies metric (cf. Sec. 2.3). This means that two neighbor parts will be in the same segment only if they are best buddies. We have tested the segments defined by the *segmenter* and found that 99.7% of the neighbors in each segment are genuine neighbors in the original image. The results of this second solver module are shown in Figs. 4.1b and 4.4b.

3.3 The Shifter

Once coherent solution segments are computed, we finally relocate the parts on the board such that a better puzzle solution is reconstructed. Towards that goal, the *shifter* iteratively repeats the following steps:

1. It uses the current largest segment (S_{max}) in the most recent reconstructed puzzle as a new seed while executing the greedy placer once again. Note that since the puzzle grows around the seed with no reference to a particular position, this is equivalent to shifting S_{max} to some better place as determined by the greedy placement procedure.

2. Once the placer completes a new puzzle reconstruction, it performs the segmentation again.

This loop is repeated until the evaluation of the best buddies metric reaches a local maximum (see Fig. 3.1).

Chapter 4

Experimental Results

The following benchmark shows our solver performance results. To obtain truly comparative results to the state-of-the-art, we used the same database of 20 images from Cho *et al.* [5], where each puzzle problem consists of 432 parts of 28×28 pixels. For these experiments we used a solver with the prediction-based compatibility metric, without globally improving it. For each puzzle problem the solver is executed 10 times, each with a random seed, and the solution selected is the one having the highest best buddies estimation metric score. The average performance metrics accuracy results for all 20 images are 94% accuracy under the *Direct* comparison, and 95% under the *Neighbor* comparison, which is a significant improvement over past methods (lower than 10% and 55%, respectively, in Cho *et al.* [5]). Importantly, we were able to obtain 100% accurate solutions for 65% of the images. The contribution of the shifting module was significant, since the direct and neighbor accuracy scores were 35% and 74% after the placer's initial reconstruction and prior to the shifting phase. On average, it took the solver roughly 1.2 minutes (on a desktop with a 3.20 GHz CPU with 4 GB RAM) and

6 iterations to generate a solution. Fig. 4.1 shows our solver's results for two images. In addition to the final result, this figure also shows the results of the initial placement by the placer, the initial segmentation by the segmenter, and the final result after the shifter.

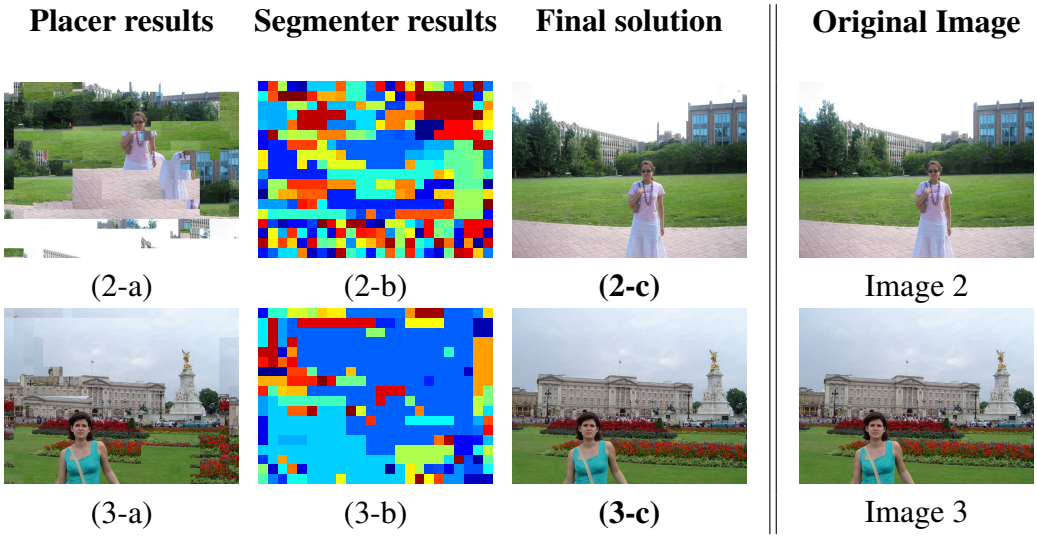
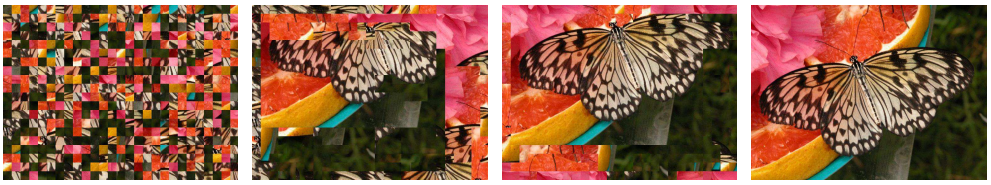


Figure 4.1: Examples of results by our solver with the same images used by Cho *et al.* [5]. For each image i (as numbered in the original collection), panel $i-a$ shows the initial placement results of the greedy placer. Panel $i-b$ shows the segments of panels $i-a$ (depicted by different colors). Panel $i-c$ shows the final result of the shifter. In both cases the results are similar to the original image (shown on the right). Image 2 direct and neighbor reconstruction accuracy are 82% and 81%, respectively. Image 3 was constructed with 100% accuracy.

In Fig. 4.2, you can see the 4 iterations that the solver required to reconstruct the butterfly image. Notice that the third and fourth iterations yielded the same solution, and thus resulted with the same estimation value which insinuated convergence.

As shown in Fig. 4.3, the final results of our solver are superior to the state-of-the-art (Cho *et al.*[5]) even though it *never uses externally provided clues*. Note in particular how some puzzle problems are not solved perfectly by previous meth-



Input image **First iteration** **Second iteration** **Third and fourth iterations**

Figure 4.2: This example shows the 4 iterations required to converge on a 540 parts puzzle, taken from McGill image database [21]. As you can see, the butterfly is gradually shifted toward its correct location.

ods even after being provided with 10 anchors, while our approach does so successfully with no externally provided prior whatsoever. Clearly, comparing these two methods on equal grounds (shown in columns a,d of Fig. 4.3) shows significant qualitative improvement.

Applying global improvements to the prediction-based compatibility metric, and using the corresponding greedy criterion tends to further improve results. Since our algorithm is not deterministic, due to the fact that it depends on the initial seed, it might not always demonstrate such improvement. Nevertheless, an improvement could be observed in the following test. For each of the 20 images from Cho *et al.* [5], we invoke our algorithm with each of the 432 possible seeds and pick the solution having the highest best buddies estimation metric score. The average performance metrics accuracy results without globally improving are 95.2% accuracy under the direct comparison, and 95.4% under the neighbor comparison. When applying global improvements we received 95.3% accuracy under the direct comparison, and 95.9% under the neighbor comparison.

Fig. 4.4 shows our solver's results on puzzle problems of various part numbers, including sizes never before attempted. We used two additional databases of

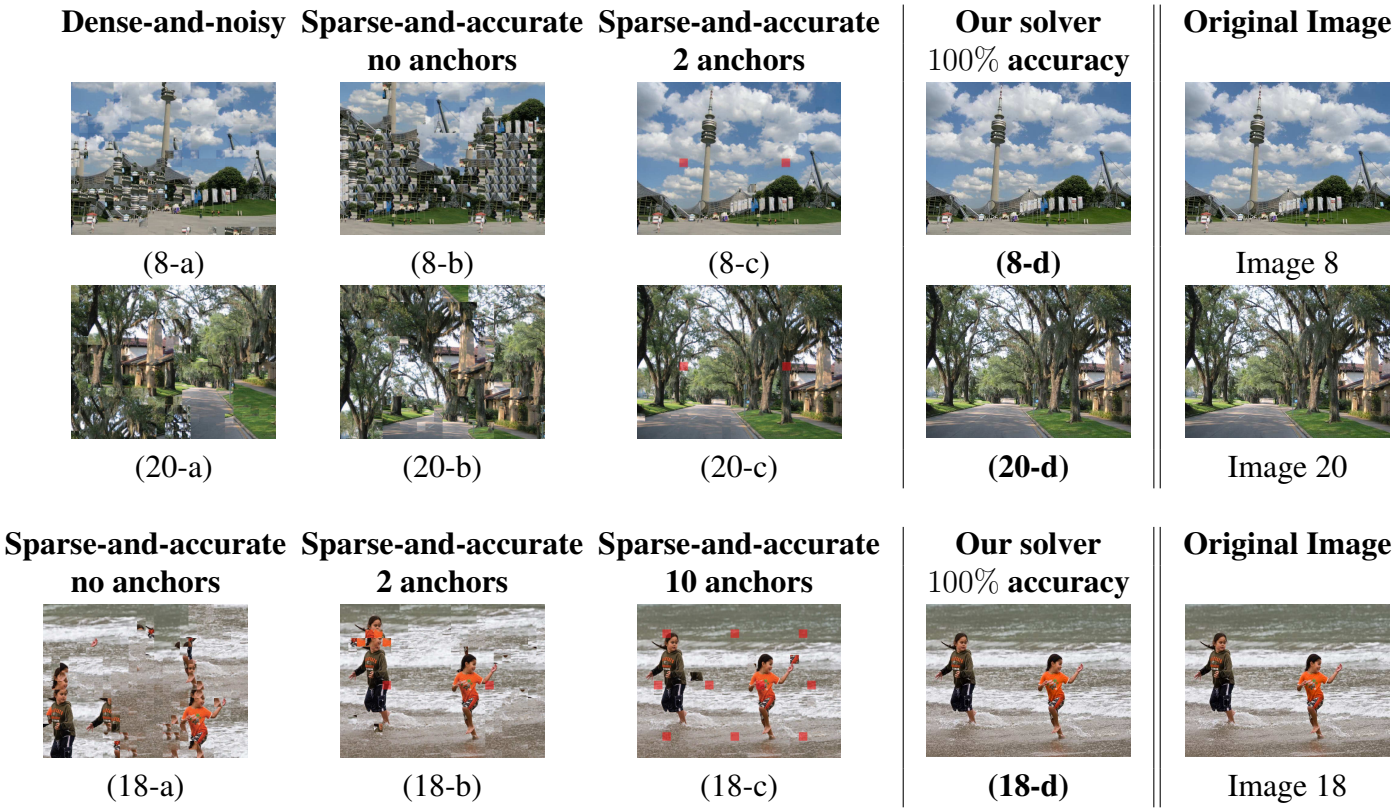


Figure 4.3: Comparison of our results to the state-of-the-art. Shown are the results of Cho *et al.* [5] for both their dense-and-noisy local evidence, and sparse-and-accurate local evidence with various numbers of anchors. Note that in some cases, even 10 anchors provide inferior results to those obtained by our solver (with no externally provided prior whatsoever). Clearly, comparing these two methods on equal grounds (i.e., with no externally provided clues, shown in columns a,d), demonstrates significant qualitative improvement.

20 images, each divided into parts of 28×28 pixels each. The first, divided into 540 parts, was taken from the McGill image database [21]. The second, divided into 805 parts, was self-created (all images can be inspected in [26]). The average reconstruction accuracy for the direct and neighbor comparison is 83% and 91% respectively for the 540 parts database, while 50% of the images are constructed with 100% accuracy. Here it took an average of 1.9 minutes and 6.5 iterations to

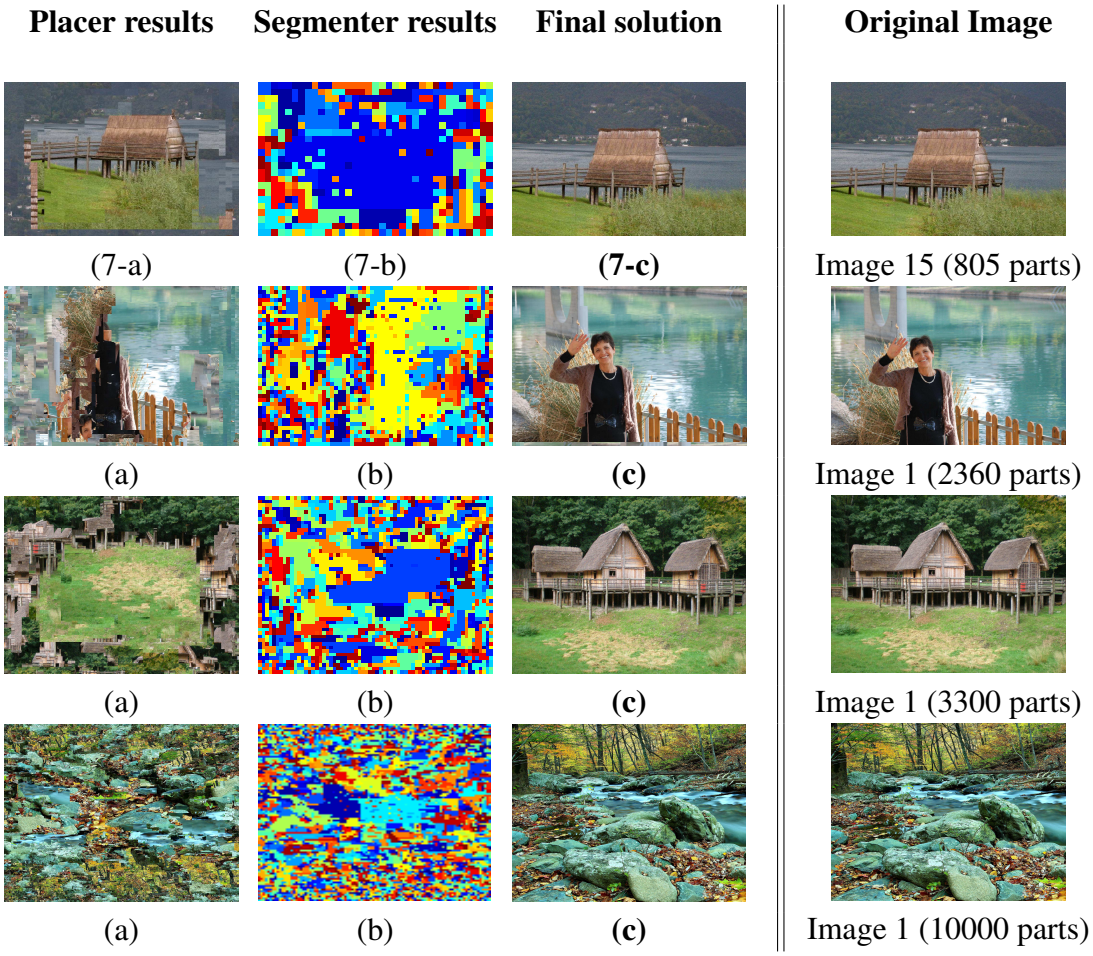


Figure 4.4: Selected results of our solver for larger puzzle problems. The first row shows a result for a puzzle containing 805 parts, solved with 100% accuracy. Please refer to the text for average solution performance on the entire database of 540 and 805 parts puzzles. The second and third examples show a single solution for puzzle of 2360 and 3300 parts, respectively, (each 28×28 pixels in size), where performance was 0.21% and 95% accurate in terms of the direct and neighbor metrics for the second example and 100% for the third. The last example shows a solution for a puzzle comprised of 10000 parts which was perfectly assembled.

generate a solution. The average reconstruction accuracy for the direct and neighbor comparison is 80% and 90% respectively for the 805 parts database, while 35% of the images are constructed with 100% accuracy. In this case solutions were generated on average after 5.1 minutes and 8 iterations. While performance

is expectedly decreasing, note how it surpasses previously reported performance (e.g., [5]) for problems of size as large as 432 parts. In addition, Fig. 4.4 shows three examples of puzzles whose size is an order of magnitude larger. We note that due to large demand of both memory and time, we have not tested the algorithm's limits on even bigger instances. We further note that the jigsaw solver code and images are available for the use of the community [26].

Chapter 5

Conclusions and Future Work

5.1 Conclusions

We presented a new greedy solver for the (square) jigsaw puzzle problem by employing part placement, region segmentation, and shifting phases, all informed by a novel part compatibility and solution estimation measures. Unlike previous work, our solver uses no clues or priors about the original image or the problem solution, and it requires no manual interaction, yet it still exhibits much improved results compared to the state-of-the-art. Arguably, the ability of our solver to provide successful solutions for puzzles whose size is an order of magnitude larger than ever before attempted despite using no clues, hints, or other priors, implies that this problem may be simpler than previously thought. Furthermore, in general, humans are considered superior to computers at solving vision problems. A small example which demonstrates this is a *captcha* test which can usually detect if a human or a machine answered it. However, computers may now be seen as better at solving visual puzzles.

5.2 Possible Future Work

Several main issues still deserve further investigation in the context of our solver and jigsaw puzzle solving in general. For example, formulating compatibility metrics that can handle typical failure cases (e.g., around roughly homogeneous image regions) would clearly achieve improved results. Furthermore, a critical issue for consideration is the choice of initial seed, which has a great effect over the final solution. Identifying criteria that successfully *predict* a good seed (as opposed to repeated random selections) could help reduce the solver's computational cost. Another approach might seek to completely remove the need for a seed, thus creating a deterministic algorithm. Finally, while the degradation effect of increasing the number of parts has been explored empirically in the puzzle solving community, much less insight has been obtained about the effect of part *size*, or how uncertainty regarding parts' appearance (e.g., due to noise) could affect the solution. We believe that studying such issues will improve solvers' accuracy significantly, and will contribute greatly to the problem of jigsaw puzzles in general.

At the time this thesis was wrapped up, some new and interesting work which further progress the current research was published. As most of our research was published in the past year [25], the new research develops on top of that new and improved methods and ideas. Such can be seen in the work by Gallagher [11] and by Andaló *et al.* [2]. We hope it not only shows the impact of our own work but also the interest in the puzzle problem in years to come.

Bibliography

[1] Naif Alajlan. Solving square jigsaw puzzles using dynamic programming and the hungarian procedure. *American Journal of Applied Sciences*, 11:1942–1948, 6 2009.

[2] Fernanda A. Andaló, Gabriel Taubin, and Siome Goldenstein. Solving image puzzles with a simple quadratic programming formulation. In *Proceedings of the Conference on Graphics, Patterns and Images*, 2012.

[3] Benedict J. Brown, Corey Toler-Franklin, Diego Nehab, Michael Burns, David Dobkin, Andreas Vlachopoulos, Christos Doumas, Szymon Rusinkiewicz, and Tim Weyrich. A system for high-volume acquisition and matching of fresco fragments: Reassembling Theran wall paintings. *ACM Transactions on Graphics*, 27(3), 2008.

[4] Taeg Sang Cho, Shai Avidan, and William T. Freeman. The patch transform. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(8):1489–1501, 2010.

[5] Taeg Sang Cho, Shai Avidan, and William T. Freeman. A probabilistic image jigsaw puzzle solver. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 183–190, 2010.

[6] Min Gyo Chung, Margaret M. Fleck, and David A. Forsyth. Jigsaw puzzle solver using shape and color. In *In Proceedings of the International Conference on Signal Processing*, 1998.

[7] DARPA. The DARPA Shredder Challenge. http://archive.darpa.mil/shredderchallenge/, 2012.

[8] Erik Demaine and Martin Demaine. Jigsaw puzzles, edge matching, and polyomino packing: Connections and complexity. *Graphs and Combinatorics*, 23:195–208, 2007.

[9] H. Freeman and L. Garder. Apictorial jigsaw puzzles: the computer solution of a problem in pattern recognition. *Electronic Computers, IEEE Transactions on*, 13:118–127, 1964.

[10] D. Gale and L. S. Shapley. College admissions and the stability of marriage. *American Mathematical Monthly*, 69:9–14, 1962.

[11] Andrew C. Gallagher. Jigsaw puzzles with pieces of unknown orientation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 382–389, 2012.

[12] David Goldberg, Christopher Malon, and Marshall W. Bern. A global approach to automatic solution of jigsaw puzzles. In *Symposium on Computational Geometry*, pages 82–87, 2002.

[13] Florian Kleber and Robert Sablatnig. A survey of techniques for document and archaeology artefact reconstruction. In *Proceedings of the Inter-*

national Conference on Document Analysis and Recognition, pages 1061–1065, 2009.

[14] D. Koller and M. Levoy. Computer-aided reconstruction and new matches in the forma urbis romae. *Bullettino Della Commissione Archeologica Comunale di Roma*, 15:103–125, 2006.

[15] D.A. Kosiba, P.M. Devaux, S. Balasubramanian, T.L. Gandhi, and K. Kasturi. An automatic jigsaw puzzle solver. In *Pattern Recognition*, volume 1, pages 616–618, 1994.

[16] Huei-Yung Lin and Wen-Cheng Fan-Chiang. Reconstruction of shredded document based on image feature matching. *Expert Syst. Appl.*, 39(3):3324–3332, 2012.

[17] M. Makridis and N. Papamarkos. A new technique for solving a jigsaw puzzle. In *Proceedings of the IEEE International Conference on Image Processing*, pages 2001–2004, 2006.

[18] Nasir Memon and Anandabrata Pal. Automated reassembly of file fragmented images using greedy algorithms. *IEEE Transactions on Image Processing*, 15:385–393, 2006.

[19] Takenori Murakami, Fubito Toyama, Kenji Shoji, and Juichi Miyamichi. Assembly of puzzles by connecting between blocks. In *Proceedings of the IEEE International Conference on Pattern Recognition*, pages 1–4, 2008.

[20] Ture R. Nielsen, Peter Drewsen, and Klaus Hansen. Solving jigsaw puzzles using image features. *Pattern Recognition Letters*, 29(14):1924–1933, 2008.

[21] A. Olmos and F. A. A. Kingdom. McGill calibrated colour image database. http://tabby.vision.mcgill.ca., 2005.

[22] Constantin Papaodysseus, Thanasis Panagopoulos, Michael Exarhos, Constantin Triantafillou, Dimitrios Fragoulis, and Christos Doumas. Contour-shape based reconstruction of fragmented, 1600 b.c. wall paintings. In *IEEE Transactions on Signal Processing*, volume 50, pages 1277–1288, 2002.

[23] Ioannis Pitas. *Digital Image Processing Algorithms and Applications*. John Wiley and sons, INC., 2000.

[24] Yair Poleg and Shmuel Peleg. Alignment and mosaicing of non-overlapping images. In *Proceedings of the IEEE International Conference on Computational Photography*, pages 1–8, 2012.

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

[26] D. Pomeranz, M. Shemesh, and O. Ben-Shahar. A fully automated greedy square jigsaw puzzle solver MATLAB code and images. http://www.cs.bgu.ac.il/%7Eicvl/projects/project-jigsaw.html, 2012.

[27] L.I. Rudin, S. Osher, and E. Fatemi. Nonlinear total variation based boise removal algorithms. *Physica D*, 60:259–268, 1992.

[28] Richard Szeliski. Image alignment and stitching: A tutorial. *Foundations and Trends in Computer Graphics and Vision*, 2:1–104, 2006.

[29] B. Tang, G. Sapiro, and V. Caselles. Diffusion of general data on non-flat manifolds via harmonic maps theory: The direction diffusion case. *International Journal of Computer Vision*, 36(2):149–161, 2000.

[30] Fubito Toyama, Yukihiro Fujiki, Kenji Shoji, and Juichi Miyamichi. Assembly of puzzles using a genetic algorithm. In *Proceedings of the IEEE International Conference on Pattern Recognition*, pages 389–392, 2002.

[31] Robert Tybon. *Generating Solutions to the Jigsaw Puzzle Problem*. PhD thesis, Griffith University, 2004.

[32] Miri Weiss-Cohen and Yoram Halevi. Knowledge retrieval for automatic solving of jigsaw puzzles. In *Computational Intelligence for Modelling, Control and Automation, 2005 and International Conference on Intelligent Agents, Web Technologies and Internet Commerce, International Conference on*, volume 2, pages 379–383, 2005.

[33] Xingwei Yang, Nagesh Adluru, and Longin Jan Latecki. Particle filter with state permutations for solving image jigsaw puzzles. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2873–2880, 2011.

[34] Feng-Hui Yao and Gui-Feng Shao. A shape and image merging technique to solve jigsaw puzzles. *Pattern Recognition Letters*, 24(12):1819–1835, 2003.

[35] Yu-Xiang Zhao, Mu-Chun Su, Zhong-Lie Chou, and Jonathan Lee. A puzzle solver and its application in speech descrambling. In *Proceedings of the 2007 annual Conference on International Conference on Computer Engineering and Applications*, pages 171–176, 2007.

תקציר

בעית הפאזלים הריבועיים הינה בעיה בה אנו נדרשים לשחזר תמונה בהנתן קבוצה לא חופפת של חלקי תמונה ריבועיים, לא מסודרים. בתזה זאת אנו מציעים פותר אוטומטי לבעיה זאת, אשר בניגוד לפותרים קודמים אינו עושה שימוש כלל ברמזים לגבי מיקומי החלקים ואינו זקוק למידע כלשהו על התמונה המקורית. לשם כך, אנו מציגים פותר חמדן אשר משלב בין הצבה מושכלת של חלקי הפאזל לבין סידור מקטעים מורכבים לשם קבלת התמונה המשוחזרת. בין התרומות של עבודה זאת גם מטריקות חדשות אשר מאפשרות לחזות בצורה טובה יותר את ההסתברות ששני חלקים נתונים הינם אכן שכנים בתמונה המקורית, ומטריקות הערכה חדשנית אשר מצליחה לחזות את איכות פתרון מוצע ללא צורך בתמונה המקורית. ברתימת תרומות אלו הפותר שלנו מגיע לתוצאות המתועדות הטובות ביותר, ואף מצליח להתמודד בהצלחה עם פאזלים גדולים מכפי שנוסה אי פעם.

אוניברסיטת בן-גוריון בנגב

הפקולטה למדעי הטבע

המחלקה למדעי המחשב

פתרון פאזלים ריבועיים

חיבור זה מהווה חלק מהדרישות לקבלת התואר "מוסמך למדעי הטבע" (M.Sc.)

מאת: דולב פומרנץ

מנחה: פרופ. אוהד בן-שחר

חתימת הסטודנט: _____ תאריך: _____

חתימת המנחה: _____ תאריך: _____

חתימת יו"ר הועדה המחלקתית: _____ תאריך: _____

אוקטובר 2012

אוניברסיטת בן-גוריון בנגב

הפקולטה למדעי הטבע

המחלקה למדעי המחשב

פתרון פאזלים ריבועיים

חיבור זה מהווה חלק מהדרישות לקבלת התואר "מוסמך למדעי הטבע" (M.Sc.)

מאת: דולב פומרנץ

מנחה: פרופ. אוהד בן-שחר

חשון ה'תשע"ג אוקטובר 2012