SJSU SAN JOSÉ STATE UNIVERSITY

Square Jigsaw Puzzle Solver Literature Review

Prepared by: Zayd Hammoudeh
(zayd.hammoudeh@sjsu.edu)

Introduction

- "Jigsaw Puzzle Problem"
  - **Problem Statement:** Reconstruct an image from a set of image patches
  - **Problem Complexity:** NP-Complete (via the set partition problem) when pairwise affinity of pieces is unreliable [1]

- **Problem Formulation**: Set of square, non-overlapping patches
  - This specific type of puzzle is known as "*jig swap*" [7] or "*Type 1*" puzzles [19]
  - *Type 2 Puzzles*: Allow/disallow patch rotation [19]

- **A Key Difference with Standard Jigsaw Puzzle Solving:** The source image you are trying to reconstruct is unknown.

Square Jigsaw Puzzle Example

- Source image (left) is divided into 81 (9x9) uniform, square patches (center). The goal is to organize the patches to reconstruct the source image (right).

SJSU SAN JOSÉ STATE UNIVERSITY

Jigsaw Puzzle Solver Applicability

- Possible and existing applications of the jigsaw puzzle problem include:
  - **Computer Forensics**: Reconstructing deleted JPEG, block-based images [2]
  - **Document Investigation**: Reconstruct shredded documents [3]
  - **Bioinformatics**: DNA/RNA modelling and reconstruction [4]
  - **Archeology**: Reconstruction of damaged relics [5]
  - **Audio Processing**: Voice descrambling [6]

SJSU SAN JOSÉ STATE UNIVERSITY

Additional Variants of Problem

[9] proposes a list of additional variants to the jigsaw puzzle problem including:

- Missing piece(s)
- Extra piece(s)
- Unknown piece orientation (type 2 puzzle)
- Three dimensional puzzles
- Unknown puzzle dimension

Pairwise Affinity

- **Definition:** Quantifies the similarity/compatibility between two patches.

- Between two pieces x_i and x_j, there are 4 pairwise affinity values when rotation is not allowed and 16 when rotation is allowed.

- Metrics of particular interest in the literature are divided into two categories.
  - Boundary/Edge Based:
    - Normalized and Unnormalized Dissimilarity-based Compatibility
    - Prediction-based Compatibility

  - Statistical based using the entire patch and its statistical properties [14]

SJSU SAN JOSÉ STATE UNIVERSITY

Dissimilarity-Based Compatibility

- Proposed in Cho *et. al.* [7]

- Uses the LAB (lightness, and a/b color opponent dimensions), which is three (3) dimensions.

- Given two patches x_i and x_j that are size K pixels by K pixels, then left-right (LR) dissimilarity (where x_j is to the right of x_i) is:

$$D_{LR}(x_i, x_j) = \sum_{l=1}^{K} \sum_{d=1}^{3} \left(x_i(l, K, d) - x_j(l, 1, d)\right)^2$$

Where $x_m(r, c, d)$ is value for the pixel in row r and column c of patch x_m at dimension d.

- **Disadvantage of this Approach:**
  - Severely penalizes boundary differences between patches which *do* occur in actual images [10].
  - It is common that actual image does **not** the minimum dissimilarity. Hence, this "*better than perfect score*" where the solved solution has a lower score than the original is a type of overfitting [9].

$\left(L_p\right)^q$ Dissimilarity-Based Compatibility

- Proposed by Pomeranz *et. al*. in [10]. *Generalizes* the dissimilarity metric from [7] with the L_p norm.

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

Hence, [7]'s metric is essentially the $(L_2)^2$ norm.

- While q has no effect on the patch pairwise classification accuracy, [10] observed it had an effect on their solver's performance

Prediction-Based Compatibility

- The dissimilarity based approach measured the difference between two pieces.
  - Prediction based attempts to predict the boundary pixel value of the neighboring patch.

- **First-Order Example**:
  - Use the last two pixels of each patch to predict the neighboring piece's value.
  - Gradient between two right edge pixels for patch x_i in row l for dimension d:

$$x_i(l, K, d) - x_i(l, K - 1, d)$$

  - Gradient between two left edge pixels for patch x_j row l for dimension d:

$$x_j(l, 1, d) - x_i(l, 2, d)$$

Prediction-Based Compatibility (Continued)

- The two pixel gradient can be combined with the dissimilarity-based compatibility as shown below for patch x_i's right edge:

$$\big(x_i(l,K,d) - x_j(l,1,d)\big) + \big(x_i(l,K,d) - x_i(l,K-1,d)\big)$$

which is equivalent to:

$$\big(2 * x_i(l,K,d) - x_i(l,K-1,d)\big) - x_j(l,1,d)$$

- If the $\left(L_p\right)^q$ dissimilarity is used, the entire prediction based compatibility for the left-right boundary of x_i and x_j is:

$$\sum_{l=1}^{K}\sum_{d=1}^{3}\left(\left|\big(2 * x_i(l,K,d) - x_i(l,K-1,d)\big) - x_j(l,1,d)\right|^p + \left|\big(2 * x_j(l,1,d) - x_j(l,2,d)\big) - x_i(l,K,d)\right|^p\right)^{\frac{q}{p}}$$

- **Advantage of this Approach:** Incorporates a predictor of the pairwise change which may help detect expected pixel pairwise differences.

Accuracy Comparison of the Compatibility Metrics

- Pomeranz *et. al.* in [10] compared the accuracy of the three compatibility metrics on 20 images in a test dataset.

- Using the $(L_p)^q$ norm resulted in a 7% to 10% improvement in selecting the correct neighbor.

- The impact of using the prediction-technique varied from no change up to a 3% improvement.

Puzzle Size	Dissimilarity-Based	$(L_{3/10})^{1/16}$	Prediction-Based
432 Patches	78%	**86%**	**86%**
540 Patches	76%	85%	**88%**
805 Patches	74%	84%	**86%**

Comparison of Pairwise Similarity Metric Accuracy

Asymmetric Dissimilarity

- The previous definitions of pairwise affinity have been symmetrically similar such that:

$$D\left(p_i, p_j, right\right) = D(p_j, p_i, left)$$

- [20] proposes using an asymmetric dissimilarity such that equality in the above equation does not hold.

- One sided, L_1 version of Pomeranz *et. al.*'s prediction based distance as shown below:

$$D\left(x_i, x_j, right\right) = \sum_{l=1}^{K} \sum_{d=1}^{3} \left\| \left(2 * x_i(l, K, d) - x_i(l, K-1, d)\right) - x_j(l, 1, d) \right\|$$

Benefits of Asymmetric Dissimilarity

- Three times faster due to the elimination of the exponent (80% of runtime is in distance calculations)
  - Additional speedup can be gained if when the asymmetric dissimilarity is sufficiently large (i.e. no chance of a pairing), the calculation is stopped and the distance set to infinity.
- Number of correct "best buddies" increased
- Number of incorrect decreased
- Using the benchmark in [17], the number of correctly solved puzzles increased from 25 to 37.

Performance Metrics

- **Summary:** Evaluate the performance of a jigsaw puzzle solver against the original (correct) image.

- Cho *et. al.* proposed three performance metrics:
  - *Direct Comparison Method:* Most naïve approach. The ratio of the number of patches in their correct locations versus the total number of patches.
    - *Disadvantage:* Susceptible to shifts
  - *Neighbor Comparison Method:* For each patch, calculate the fraction of the four neighbors that are correct. The total accuracy is the average neighbor accuracy of all patches.

"Best Buddies"

Definition: Two patches are *best buddies* if they are more similar to each other on their respective sides than they are two any other pieces [10].

Hence, two patches, x_i and x_j, are said to be "best buddies" for a spatial relationship R. if and only if, two conditions hold:

$$\forall x_k \in \{Patches\}, C\left(x_i, x_j, R_1\right) \geq C(x_i, x_k, R_1)$$

$$\forall x_k \in \{Patches\}, C\left(x_j, x_i, R_2\right) \geq C(x_j, x_k, R_2)$$

Where:

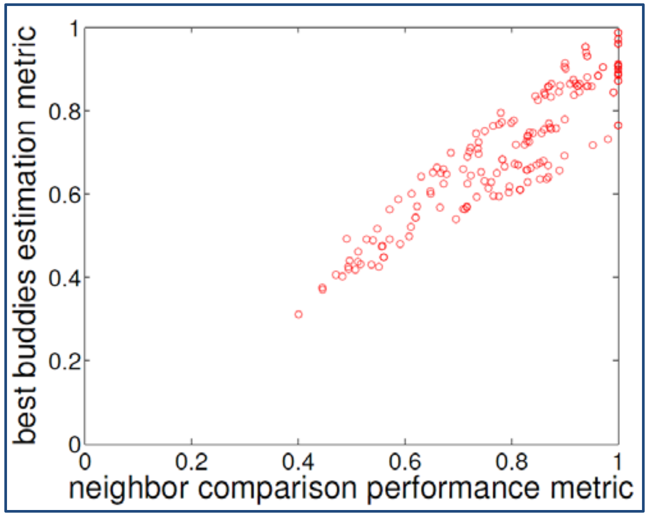
- $C(x_i, x_j, R_1)$ – Compatibility between patches x_i and x_j on side R_i of x_i
- $\{Patches\}$ – Set of all patches in the puzzle
- R_1 – One of the four sides (e.g. top, bottom, left, right) of x_i where x_j will be placed assuming no rotation.
- R_2 - Given x_i and R_1, this represents the complementary side of x_j. For example if R_1 is "left", then R_2 would be "right"

"Best Buddies" Estimation Metric

- **Definition:** Ratio of the number of neighbors who are said to be "best buddies" to the total number of neighbors [10].

- Correlation between the "Best Buddies" Estimation Metric and Cho *et. al.*'s two performance metrics:
  - *Direct Comparison Metric*: Little to no correlation since direct comparison method is not based on pairwise accuracy.

  - *Neighbor Comparison Metric*: Stronger correlation Graph below is for 20 images tested 10 times each (for 200 total points)



Scatterplot of "Best Buddy" Metric versus Neighbor Comparison Metric

Existing Square Jigsaw Puzzle Approaches

- Dynamic Programming and the "Hungarian" Procedure [13]
- Patch Transform using a Low Resolution "Solution Image" [8]
- "Dense and Noisy" or "Sparse and Accurate" with Loopy Belief Propagation [7]
- Particle Filter-Based Solver [11]
- Greedy Algorithm [10]
- Genetic Algorithm [9]

- **Variations of the Problem**
  - Unknown orientation (i.e. rotation) and puzzle dimensions [12]

Patch Transform

- Introduced by Cho *et. al.* in [8]

- **Overview of the Patch Transform**: Segment a source image into a set of non-overlapping "patches" and rearrange these patches and reorganize the image in the "patch" domain.
  - *Intended Usage*: Image editing

- **"Inverse" Patch Transform**: Reconstruct an image from a set of patches. This requires two components:
  - A patch compatibility function
  - An algorithm that places all patches

- Uses a provided low resolution image as part of the patch placement algorithm.

Markov Random Field

- Use a Markov Random Field (MRF) to enforce three rules:
  - Adjacent pieces should fit plausibly together
  - A patch should "*never*" (or in the loosened case "*seldomly*") be reused.
  - User constraints (e.g. board size) on patch placement.

- Consider each possible patch location as a node in the MRF. The key notation definitions:
  - x_i – Undetermined state for the node i^{th} in the MRF.
  - $\psi_{i,j}(k, l)$ – Compatibility between patches k and l at adjacent MRF locations i and j
  - X – Vector of N determined patch indices, x_i
  - Y – Low resolution version of the original image.

Maximizing the Patch Assignment Probability

For a given patch assignment X, the probability of that assignment is defined as:

$$P(X) = \frac{1}{Z}\prod_{i}\phi_i(x_i)\prod_{j\in\zeta(i)}\left(\psi_{ij}\left(x_i, x_j\right) * E(x)\right)$$

- i : i^{th} node in the MRF/board
- N : Number of nodes in the MRF/board.
- $\phi_i(x_i)$: User constraints (e.g. board size)
- $\psi_{ij}\left(x_i, x_j\right)$: Patch to patch compatibility
- $\zeta(i)$: Markov blanket of node i
- $E(X)$: Exclusion term that discourages patches being used more than once.
- Z : Normalization term to ensure $\int P(X)\, dX = 1$

Loopy Belief Propagation Solver

- Maximizes the preceding probability function using loopy belief propagation.
- Susceptible to local maxima so random restarts may be performed.
- **Question:** What if I do not have access to a low resolution version of the original image? Can I make one or use a substitute?

“Dense and Noisy” Estimation

- Proposed by Cho *et. al.* in [7] in 2010.

- **Review:** In Cho *et. al.*’s work in [8], they assumed access to a correct, low resolution version of the original image.
  – In many real world applications, such a low resolution image is not available.

- **Solution:** Estimate a low resolution image from a “bag of patches.” The simplified procedure is:
  – Creating a histogram of the bag of patches
  – “Estimate” a low resolution version by comparing the histogram to a set of K centroids with predefined low resolution images.

“Dense and Noisy” Clustering and Histogram Generation

- **Training Set:** 8.5M patches from 15,000 images.
  - *Patch Size:* 7px by 7px by 3 (LAB) for 147 total, original dimensions. This dimensionality is reduced via PCA.

- **Clustering the Patches**
  - *Step #1:* Cluster each image’s patches into L (e.g. 20) centroids.
  - *Step #2*: Re-cluster L centroids from all images into N (e.g. 200) centroids.

- **Creating the Histogram:** For a given image, assign each patch to its closest centroid.

"Dense and Noisy" – Generating the Low Res. Image

- **Theoretical Motivation:** Different colors are more likely to be at different places in an image.
  - *Example*: Blue (sky) is more likely to be towards the top of the image while brown (soil) tends to be in the image foreground.

- **Mapping Bins to the Image**: Use the training set to generate probability density maps for each histogram bin.

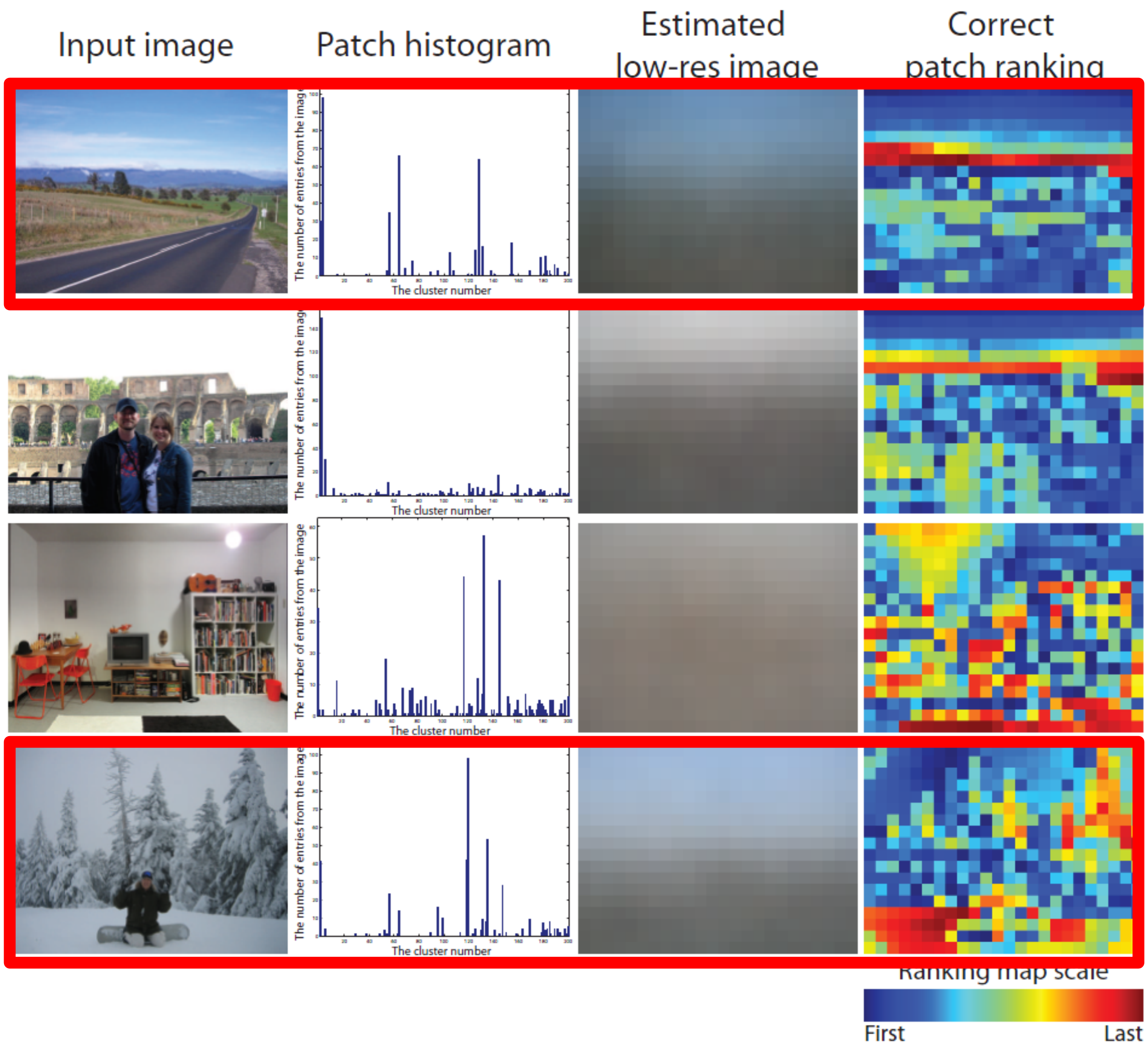
- **Using the Histogram to Create the Low Resolution Image**: Use a trained, linear regression function to map a bag of pieces histogram to the training images (i.e. use prior knowledge).

“Dense and Noisy” Results

- **Summary:** Patch histogram can “coarsely predict” a low resolution of the original image.
  - *Possible Explanation*: There is enough “structural regularity” in images that a bag of patches proves spatial information.

- **Patch Rank Map:** For each pixel in the low resolution images, patches are ranked from least likely to most likely to reside in that location.
  - *Ideal Case*: The set of patches that map to the low resolution will have the best rank (i.e. 1)
  - *Worst Case*: The matching set of patches will have rank N, where N is the number of patches in the image.

"Dense and Noisy" End to End Example



Best Results

Worst Results

Confused snow for sky

“Sparse and Accurate”

- Proposed by Cho *et. al.* in [7]

- **Common Human Approach to Solving Puzzles**: “Outside-in”
  - Find the puzzle’s four corner pieces.
  - Build from the corner pieces until all four sections converge.

- “Sparse and accurate” is based off the “outside-in” technique.
  - **Definition on an “Anchor Patch”**: A puzzle patch that is placed in its correct location and orientation.
  - **Summary of the Approach:** Place a set of N anchor patches and then solve the puzzle.

- Two most important criteria of anchor patches
  - Quantity
  - Uniform Spatial Distribution

Generalized Greedy Algorithm

- Proposed by Pomeranz *et. al.* in [10] in 2011.
- **Goal:** Provide a computational framework for handling square jigsaw puzzles in reasonable time that does not rely on any prior knowledge or human intervention.
- Solver divides the puzzle reconstruction into three subproblems:
  - *Placement*: Given a single patch or partially-placed set of patches, place the remaining patches.
  - *Segmentation*: Given a fully-placed board, segment the board into subcomponents that are *believed* to be placed correctly.
  - *Shifting*: Given a set of trusted segments, relocate entire segments and individual patches to improve solution quality.

Overview of the Greedy Placement Phase

- Given a partially assembled board (either a single patches or set of patches), continue applying the greedy choice until all patches are placed.
- **Overview of the Greedy Choice**:
  - Board dimensions are known in advance and fixed
  - Board locations with a higher number of occupied neighbors are preferred as the choice of the next piece is more informed.
  - Patch selection criteria:
    - *Primary Criteria*: Prefer a “best buddy” first.
    - *Secondary Criteria*: If no or multiple patches satisfy the primary criteria, select the patch with the highest compatibility score.
- **Question:** Why is a placer not enough?
- **Answer:** It works solely on local information. To get the best results, we must also look at the entire global solution.

Segmenter Phase

- **Definition of “Segments”**: Areas of the puzzle that are (or “are believed to be”) assembled correctly.

- **Procedure**: Using random seeds and a segmentation predicate based on the “best buddies” metric, grow the segments via “region growing segmentation algorithm” described in [15].

- **Accuracy of the Segmenter**: 99.7%

Pomeranz's Complete Algorithm

- ***Step #1***: Select a single puzzle patch as the seed to placement phase.
- ***Step #2***: Perform the placement phase around the seed.
- ***Step #3***: Use the segmenter to partition the board.
- ***Step #4***: Calculate the "best buddies" ratio. If you are at a local maximum, stop.
- ***Step #5***: Select the largest segment from step #3 and use it as the seed of the placement phase. Return to step #2.
  - Performing this step is similar to shifting the largest segment.

Genetic Algorithm (GA) Solver

- Proposed by Sholomon *et. al.* in 2013 [9].
  - A genetic algorithm puzzle solver was first proposed in [16] in 2002.

- **Genetic Algorithm Review**
  - Based off the biological theory of natural selection.
  - Divided into a series of stages
    - Random generation of initial population
    - Successor selection
    - Reproduction
    - Mutation
  - Requires a "fitness function" that measures solution quality.

Sholomon's GA Implementation

- **Puzzle Type:** 1 (patches have known orientation)
- **Chromosome (Solution) Representation:** N by M matrix where each cell represents one patchin the puzzle.
- **Population Size:** 1,000
- **Number of Generations:** 100
- **Number of Restarts:** 10
- **Successor Selection Algorithm:** Roulette Wheel
- ***Elitism***: Always pass *four* best solutions to the next generation
- **Culling:** None
- **Mutation Rate:** 5%
- **Fitness Function:** Sum of the L_2 of all pieces in the puzzle
- **Color Space:** LAB

GA Crossover

- Takes two "*highly fit*" parents and returns one child.
  - Non-trivial as the crossover must ensure there are no duplicate/missing patches in the solution.

- Correctly assembled segments may be at incorrect absolute locations. Hence, the crossover must allow for "*position independence*", which is the ability to shift segments.

- **Sholomon *et. al.*'s Approach:** Kernel-growing.

Sholomon’s Kernel Growing Algorithm

- Start with a single puzzle patch that is “floating” in the board such that the puzzle can grow in any direction.
  - Boundary size (i.e. length by width) is fixed and known.
- **Patch Placement Algorithm:** When deciding on the next patch to place, the algorithm iterates through up to three phases.
  - *Phase #1*: In an available boundary location, place the piece where both parents agree on the neighbor.
  - *Phase #2*: Place a “best buddy” that *exists in one of the parents*.
  - *Phase #3*: Select a location randomly and pick the piece with the best pairwise affinity.
  - If in any phase there is a tie, the tie is broken randomly.
  - After a piece is placed, the placement algorithm returns to phase #1 for the next piece.
  - Once a piece is placed, it can never be reused.

Kernel Growing with Mutation

- Mutations in genetic algorithms are used to improve the quality of the final solution via increased population diversity.

- **Sholomon's Mutation Strategy:** During the first and third phase of placement, place a piece at random with some low probability (e.g. 5%)

A Possible Benchmark

- Sholomon *et. al.* provide three large puzzle datasets as well as their results for comparative benchmarking [17].
  - *Dataset Puzzle Sizes*: 5,015, 10,375, and 22,834

- Unfortunately the website seems to no longer exist. I will separately send an email to the authors about why the removed the content.

- Used as a benchmark in [20].

Measuring Solution Quality

- **Problem Statement:** There is no uniform technique for grading the final output of a square jigsaw puzzle solver.

- **Two Divergent Approaches:**
  - *Performance Metrics:* Use the original image to grade solution quality.
    - Direct Comparison [7]
    - Neighbor Comparison [7]
  - *Estimation Metrics:* Evaluates the quality of a solution without reference to the original image [10].
    - “Best Buddies” Ratio

Algorithm Runtime Comparison

To improve execution time, Sholomon *et. al.* precompute and store all pairwise dissimilarity values.

# of Pieces	Sholomon *et. al.*	Pomeranz *et. al.*
432	48.3s	1.2min
540	64.1s	1.9min
805	116.2s	5.1min
2,360	17.60min	N/A
3,300	30.24min	N/A
5,015	61.06min	N/A
10,375	3.21hr	N/A
22,834	13.19hr	N/A

Comparison of the Algorithm Execution Time
for Sholomon *et. al.* and Pomeranz *et. al.*

Managing Missing Pieces and Multiple Puzzles

- Proposed by Paikin and Tal in [20].

- Inspired by Pomeranz *et. al.*'s greedy algorithm [10] with **three additional requirements**:
  - *New Requirement #1*: A modified compatibility function
  - *New Requirement #2*: Superior initial seed selection.
  - *New Requirement #3*: Rather than making the "best"/ "closest matching" selection at each round, make the selection with the lowest chance of erring regardless of location.
    - This makes their algorithm deterministic eliminating the need for restarts.

- **Accuracy:** 97.7% on dataset in [17]

Puzzle Problem Requirements

Paikan's & Tal's jigsaw puzzle problem definition is the most difficult presented to date. It is described below:

- Size of the puzzle is unknown and may be different
- Orientation of the patches is unknown
- Patches may missing
- Input may contain pieces from multiple puzzles

Comparison of Patch Sizes

Reference	Patch Size
Cho *et. al.* (2010)	7px by 7px
Pomeranz *et. al.* (2010)	28px by 28px
Sholomon *et. al.* (2013)	28px by 28px
Wu (SJSU Thesis) [20]	25px by 25px

SJSU SAN JOSÉ STATE UNIVERSITY

List of References

[1] Erik D. Demaine and Martin L. Demaine, “Jigsaw Puzzles, Edge Matching, and Polyomino Packing: Connections and Complexity”, Graphs and Combinatorics, volume 23 (Supplement), June 2007, pages 195–208.

[2] Simson L. Garfinkel. 2010. Digital forensics research: The next 10 years. *Digital Investigation* 7 (August 2010), S64-S73.

[3] Liangjia Zhu, Zongtan Zhou, and Dewen Hu. 2008. Globally Consistent Reconstruction of Ripped-Up Documents. *IEEE Trans. Pattern Anal. Mach. Intell.* 30, 1 (January 2008), 1-13.

[4] Marande, W., and Burger, G. 2007. Mitochondrial DNA as a genomic jigsaw puzzle. *Science* 318-415.

[5] Benedict J. Brown, Corey Toler-Franklin, Diego Nehab, Michael Burns, David Dobkin, Andreas Vlachopoulos, Christos Doumas, Szymon Rusinkiewicz, and Tim Weyrich. 2008. A system for high-volume acquisition and matching of fresco fragments: reassembling Theran wall paintings. In *ACM SIGGRAPH 2008 papers* (SIGGRAPH '08).

List of References (Continued)

[6] Yu-Xiang Zhao, Mu-Chun Su, Zhong-Lie Chou, and Jonathan Lee. 2007. A puzzle solver and its application in speech descrambling. In *Proceedings of the 2007 annual Conference on International Conference on Computer Engineering and Applications* (CEA'07), 171-176.

[7] Cho, Taeg Sang, Avidan, Shai and Freeman, William T. "A probabilistic image jigsaw puzzle solver." *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, 2010.

[8] Cho, Taeg Sang, Avidan, Shai and Freeman, William T. "The Patch Transform and Its Applications to Image Editing," *Proc. IEEE Conf. Computer Vision and Pattern Recognition,* 2008.

[9] Sholomon, D.; David, O. E.; and Netanyahu, "A genetic algorithm-based solver for very large jigsaw puzzles". *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, 2013.

[10] Pomeranz, D.; Shemesh, M. & Ben-Shahar, O "A fully automated greedy square jigsaw puzzle solver," *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, 2011.

SJSU SAN JOSÉ STATE UNIVERSITY

List of References (Continued)

[11] Xingwei Yang, N. Adluru, and L. J. Latecki. 2011. Particle filter with state permutations for solving image jigsaw puzzles. In *Proceedings of the 2011 IEEE Conference on Computer Vision and Pattern Recognition* (CVPR '11). 2873-2880.

[12] A. Gallagher, "Jigsaw Puzzles with Pieces of Unknown Orientation," IEEE Conference on *Computer Vision and Pattern Recognition 2012*.

[13] N. Alajlan. Solving square jigsaw puzzles using dynamic programming and the Hungarian procedure. *American Journal of Applied Sciences*, 2009

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

[15] Ioannis Pitas. 2000. *Digital Image Processing Algorithms and Applications* (1st ed.). John Wiley & Sons, Inc., New York, NY, USA.

List of References (Continued)

[16] F. Toyama, Y. Fujiki, K. Shoji, and J. Miyamichi. Assembly of puzzles using a genetic algorithm. In IEEE Int. Conf. on Pattern Recognition, volume 4, pages 389–392, 2002.

[17] D. Sholomon, O. David, and N. Netanyahu. Datasets of larger images and GA-based solver's results on these and other sets. http://www.cs.biu.ac.il/∼nathan/Jigsaw.

[18] Wu, Fengjiao, "Using Probabilistic Graphical Models to Solve NP-complete Puzzle Problems" (2015). *Master's Projects.* Paper 389.

[19] Kilho Son, James Hays, David B. Cooper. Solving Square Jigsaw Puzzles with Loop Constraints. ECCV (6) 2014: 32-46. 2013.

[20] Genady Paikin, Ayellet Tal. Solving multiple square jigsaw puzzles with missing pieces. CVPR 2015: 4832-4839