A Fully-Automated Solver for Multiple Square Jigsaw Puzzles Using Hierarchical Clustering

October 27, 2016

Zayd Hammoudeh
hammoudeh@gmail.com

Department of Computer Science
San José State University

Primary Goal: Develop a solver that can assemble multiple jigsaw puzzles simultaneously, with performance that exceeds the state of the art.

Primary Goal: Develop a solver that can assemble multiple jigsaw puzzles simultaneously, with performance that exceeds the state of the art.

Additional Goals:

- Define the first metrics that quantify the quality of outputs from a multi-puzzle solver
- Design visualizations for viewing the errors (if any) in a solver output

- First jigsaw puzzle introduced in the 1760s. Modern jigsaw puzzles introduced in the 1930s.
- First computational jigsaw puzzle solver introduced in 1964
- Solving a jigsaw puzzle is NP-complete [1, 2]

- First jigsaw puzzle introduced in the 1760s. Modern jigsaw puzzles introduced in the 1930s.
- First computational jigsaw puzzle solver introduced in 1964
- Solving a jigsaw puzzle is NP-complete [1, 2]
- **Example Applications:** DNA fragment reassembly, shredded document reconstruction, speech descrambling, and image editing

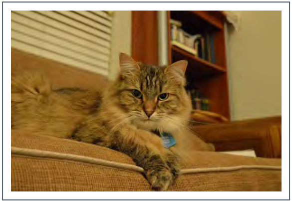
- First jigsaw puzzle introduced in the 1760s. Modern jigsaw puzzles introduced in the 1930s.
- First computational jigsaw puzzle solver introduced in 1964
- Solving a jigsaw puzzle is NP-complete [1, 2]
- **Example Applications:** DNA fragment reassembly, shredded document reconstruction, speech descrambling, and image editing
  - In most cases, the original, “**ground-truth**” image is unknown.

Jig Swap Puzzles – Variant of the traditional jigsaw puzzle

- All pieces are equal-sized squares
- Substantially more difficult

Jig Swap Puzzles – Variant of the traditional jigsaw puzzle

- All pieces are equal-sized squares
- Substantially more difficult

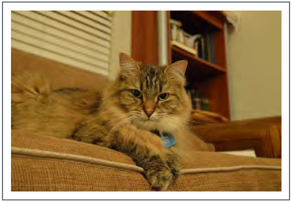


Ground-Truth Image

Dept. of Computer Science
San José State University

Jig Swap Puzzles – Variant of the traditional jigsaw puzzle

- All pieces are equal-sized squares
- Substantially more difficult



Ground-Truth Image

Randomized Jig Swap Puzzle

There are three primary jig swap puzzle types as formalized by [3]. In all cases, the "ground-truth" input is **unknown**.

There are three primary jig swap puzzle types as formalized by [3]. In all cases, the "ground-truth" input is **unknown**.

- **Type 1**: Puzzle dimensions and piece rotation are known. Only piece location is unknown.

There are three primary jig swap puzzle types as formalized by [3]. In all cases, the "ground-truth" input is **unknown**.

- **Type 1**: Puzzle dimensions and piece rotation are known. Only piece location is unknown.
- **Type 2**: All piece locations and rotations unknown. Puzzle dimensions may be known.

Introduction

Jig Swap Puzzle Types

There are three primary jig swap puzzle types as formalized by [3]. In all cases, the "ground-truth" input is **unknown**.

- **Type 1**: Puzzle dimensions and piece rotation are known. Only piece location is unknown.
- **Type 2**: All piece locations and rotations unknown. Puzzle dimensions may be known.
- **Mixed-Bag**: Pieces come from multiple puzzles.

There are three primary jig swap puzzle types as formalized by [3]. In all cases, the "ground-truth" input is **unknown**.

- **Type 1**: Puzzle dimensions and piece rotation are known. Only piece location is unknown.
- **Type 2**: All piece locations and rotations unknown. Puzzle dimensions may be known.
- **Mixed-Bag**: Pieces come from multiple puzzles.

Mixed-Bag puzzles are the focus of this thesis.

Paikin & Tal [4] – Current State of the Art

- Greedy, **kernel growing** solver
- Supports Type 1, Type 2, and Mixed-Bag puzzles
- Immune to missing pieces

Paikin & Tal [4] – Current State of the Art

- Greedy, **kernel growing** solver
- Supports Type 1, Type 2, and Mixed-Bag puzzles
- Immune to missing pieces

Limitations:

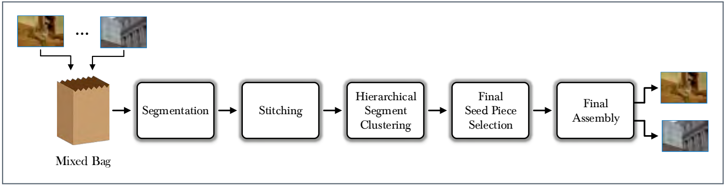
- **Poor Seed Selection**: All decisions are made at runtime using as few as 13 pieces
- **Externally Supplied Information**: The solver must be told the number of input puzzles

The Mixed-Bag Solver

SAN JOSÉ STATE UNIVERSITY 1857

Mixed-Bag Solver

Basic Structure



Paikin & Tal's Algorithm

- Begin each puzzle with a single piece
- Place all pieces around the expanding kernel

Paikin & Tal's Algorithm

- Begin each puzzle with a single piece
- Place all pieces around the expanding kernel

Alternate Jigsaw Puzzle Solving Strategy

- Correctly assemble small puzzle regions (i.e., segments)
- Iteratively merge smaller regions to form large ones

Paikin & Tal's Algorithm

- Begin each puzzle with a single piece
- Place all pieces around the expanding kernel

Alternate Jigsaw Puzzle Solving Strategy

- Correctly assemble small puzzle regions (i.e., segments)
- Iteratively merge smaller regions to form large ones
- **Advantages of this Approach**:
  - Reduces the size of the problem
  - Provides structure to the unordered set of puzzle pieces.

Dept. of Computer Science
San José State University

Paikin & Tal's Algorithm

- Begin each puzzle with a single piece
- Place all pieces around the expanding kernel

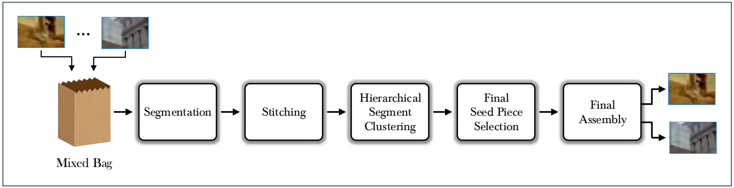
Alternate Jigsaw Puzzle Solving Strategy

- Correctly assemble small puzzle regions (i.e., segments)
- Iteratively merge smaller regions to form large ones
- **Advantages of this Approach**:
  - Reduces the size of the problem
  - Provides structure to the unordered set of puzzle pieces.

The alternate strategy is the basis of the **Mixed-Bag Solver**

Mixed-Bag Solver
Overview

- The Mixed-Bag Solver is fully-automated. It makes no assumptions concerning piece orientation, puzzle dimensions, or number of puzzles.
  - **Input**: A bag of puzzle pieces
  - **Output**: One or more disjoint, solved puzzles.
- The Mixed-Bag Solver consists of five distinct stages:



- **Role**: Place the individual pieces in the solved puzzle.
  - Mixed-Bag Solver is independent of the assembler used, giving the solver significant upgradability and flexibility.

Dept. of Computer Science
San José State University

Assembler

Mixed-Bag Solver Component

- **Role**: Place the individual pieces in the solved puzzle.
  - Mixed-Bag Solver is independent of the assembler used, giving the solver significant upgradability and flexibility.
- **Assembler Used in this Thesis**: Paikin & Tal
  - Current state of the art
  - Allows for more direct comparison of performance
  - Natively supports Mixed-Bag puzzles
- **Implementation**: Assembler re-implemented as part of this thesis based off the description in [4]
  - Written in Python and fully open source [5]

Segmentation

Mixed-Bag Solver Stage #1

- **Segment**: Partial puzzle assembly where this is a high degree of confidence pieces are placed correctly.
- **Role of Segmentation**: Provide structure to the set of puzzle pieces by partitioning them into disjoint segments
  - **Input**: A bag of puzzle pieces
  - **Output**: Set of saved segments
- **Relationship between Puzzle Pieces and Segments**:
  - Pieces from a single ground-truth input may be separated into multiple segments
  - A piece can be assigned to at most one segment

Segmentation

Algorithm Overview

- Iterative process consisting of one or more rounds
- In each round, all pieces not yet assigned to a segment are assembled as if all are from the same input image
- Segments of sufficient size are saved to be used in future Mixed-Bag Solver stages
- Pieces in a saved segment are not placed in future rounds.
- Segmentation terminates if all pieces are assigned to a saved segment or when no segment is larger than the minimum allowed size

Segmentation

Composition of a Segment

- **Starting a Segment**: Each segment is created iteratively starting with a single seed piece
- **Definition of Best Buddies**: Any pair of pieces that are more similar to each other than they are to any other piece.
- **Growing the Segment**: Add to the segment any piece that is a neighbor and best buddy of a segment member
- **Trimming the Segment**
  - **Articulation Point**: Any piece whose removal disconnects other pieces from the segment seed.
    - All articulation pieces pieces are removed from the segment.
  - After the removal of the articulation points, any pieces no longer connected to the seed are removed.

Segmentation

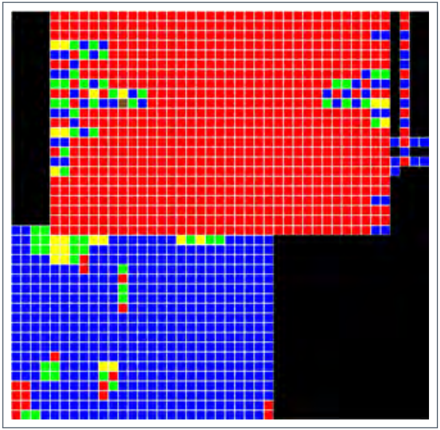
Example – Input Images

Image (a) – 805 Pieces [6]

Image (b) – 540 Pieces [7]

Segmentation

Example – First Segmentation Round Output Image



Stitching

Mixed-Bag Solver Stage #2

- **Role of Stitching**: Quantify the extent that any pair of segments is related.
  - **Input**: All puzzle pieces and the set of saved segments
  - **Output**: Segment overlap matrix

Dept. of Computer Science
San José State University

Stitching
Mixed-Bag Solver Stage #2

- **Role of Stitching**: Quantify the extent that any pair of segments is related.
  - **Input**: All puzzle pieces and the set of saved segments
  - **Output**: Segment overlap matrix

- **Theoretical Foundation**: If two segments are from the same ground-truth image, they would eventually **overlap** if one segment were to expand.
  - Segments should be allowed, but not forced, to expand in all directions.

- **Mini-assembly** (*MA*): Same as a standard assembly except only a fixed number of pieces are placed .
- **Stitching Piece** (ζ_x): A piece near the boundary of a segment that is used as the seed of a mini-assembly
- **Segment Overlap**: Maximum overlap between any mini-assembly for segment, Φ_i and another segment Φ_j.

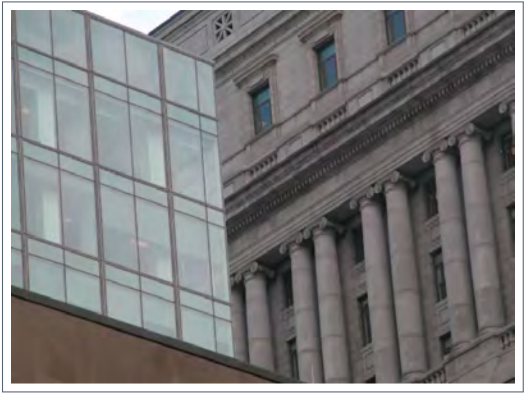
$$Overlap_{\Phi_i,\Phi_j} = \underset{\zeta_x \in \Phi_i}{\arg\max} \frac{|MA_{\zeta_x} \bigcap \Phi_j|}{\min(|MA_{\zeta_x}|, |\Phi_j|)} \tag{1}$$

- **Asymmetry**: In most cases:

$$Overlap_{\Phi_i,\Phi_j} \neq Overlap_{\Phi_j,\Phi_i} \tag{2}$$

Stitching

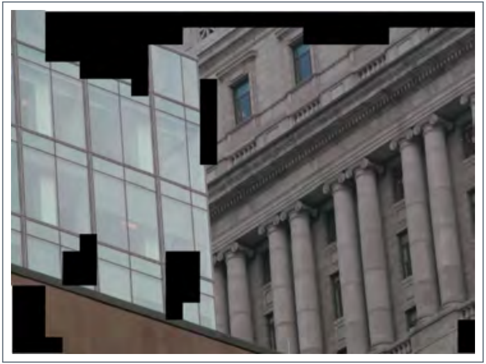
Example – Input Image



Stitching

Example – Two Segment Images

Segment #1

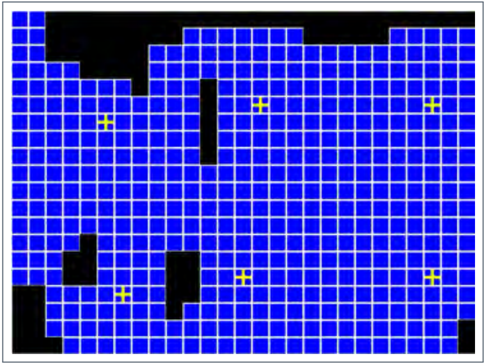


Segment #2

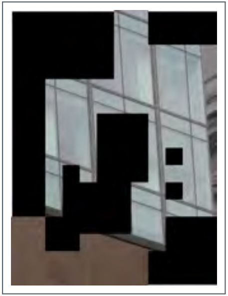
Stitching

Example – Stitching Piece Locations

Segment #1



Segment #2



Stitching Result from Segment #1

Segment Overlap:

$$Overlap_{\Phi_1,\Phi_2} = 0.83 \quad (3)$$

- A single ground-truth image may be comprised of multiple segments.

- **Role of Hierarchical Clustering**: Merge all segments from the same input image into a single segment cluster.
  - **Input**: All saved segments and the segment overlap matrix
  - **Output**: A set of **segment clusters**

Hierarchical Segment Clustering

Calculating the Initial Similarity Matrix

- **Segment Overlap Matrix**: A hollow matrix quantifying the relationship between each pair of segments.
- **Hierarchical Clustering Similarity Matrix**: A diagonal matrix quantifying the similarity between segment pairs.
- **Quantifying Similarity**: Given n segments, the similarity between segments Φ_i and Φ_j is:

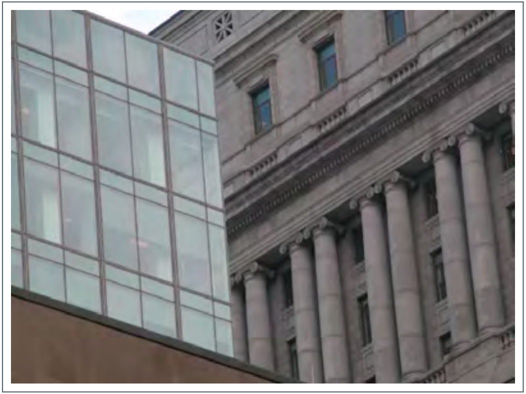
$$\omega_{i,j} = \frac{Overlap_{\Phi_i,\Phi_j} + Overlap_{\Phi_j,\Phi_i}}{2} \tag{4}$$

Hierarchical Segment Clustering

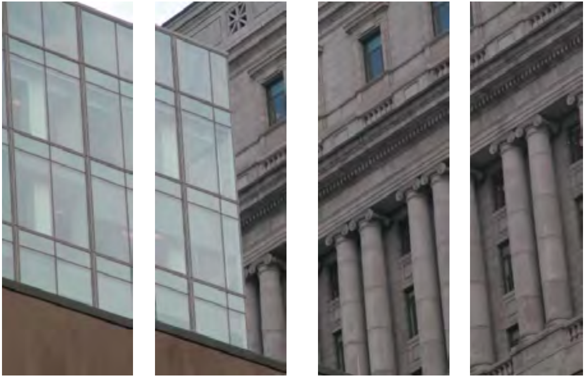
Merging Clusters

- After two clusters are combined, the similarity between the merged cluster and all other clusters must be recalculated.
- **Single Link Clustering**: The similarity between any two clusters is equal to the maximum similarity between any two members in the clusters [8]
- The Mixed-Bag Solver must use single link clustering as two clusters may only have two member segments that are adjacent.

Hierarchical Segment Clustering
Example – Single Linking



Hierarchical Segment Clustering
Example – Single Linking



Segment 1 Segment 2 Segment 3 Segment 4

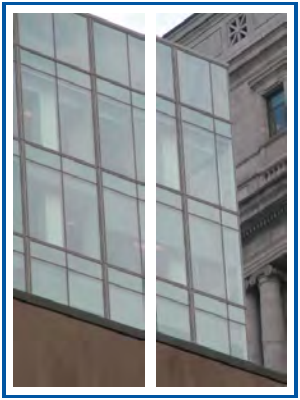
Hierarchical Segment Clustering

Example – Single Linking

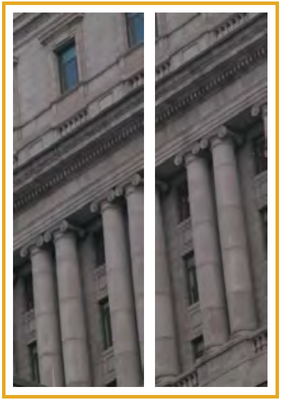
Segment Cluster 1 Segment 3 Segment 4

Hierarchical Segment Clustering

Example – Single Linking



Segment Cluster 1



Segment Cluster 2

- The solver continues merging segment clusters until one of two criteria is satisfied:
  - Only a single segment cluster remains
  - Maximum similarity between any segment clusters is below a predefined threshold
- All remaining segment clusters are passed to the next solver stage.

Mixed-Bag Solver

- **Role of Final Seed Selection**: Determine the pieces that will be used as the seed for the final output puzzles.
  - **Input**: Set of segment clusters
  - **Output**: Final seed pieces
- A single seed piece is selected from each segment cluster

Mixed-Bag Solver

- **Role of Final Seed Selection**: Determine the pieces that will be used as the seed for the final output puzzles.
  - **Input**: Set of segment clusters
  - **Output**: Final seed pieces
- A single seed piece is selected from each segment cluster

Paikin & Tal

- All puzzle seeds are selected greedily at run time, which often leads to poor decisions.

Final Assembly Stage

Mixed-Bag Solver Stage #5

- **Role of Final Assembly**: Generate the solved puzzles that are output by the Mixed-Bag Solver.
  - **Input**: Set of puzzle pieces with the seeds marked
  - **Output**: Final solved puzzles
- All pieces are placed around the seeds selected in the previous stage.
- Assembly proceeds in this stage normally without any custom modifications.

Quantifying Solver Quality

Quantifying Solver Quality

- Jigsaw puzzle solvers are not able to always correctly reconstruct the input puzzle(s)
  - Metrics compare the quality of solver outputs
- **Two Most Common Quality Metrics**:
  - Direct Accuracy
  - Neighbor Accuracy

Dept. of Computer Science
San José State University

Quantifying Solver Quality

- Jigsaw puzzle solvers are not able to always correctly reconstruct the input puzzle(s)
  - Metrics compare the quality of solver outputs
- **Two Most Common Quality Metrics**:
  - Direct Accuracy
  - Neighbor Accuracy
- **Disadvantages of Current Metrics**: Neither account for:
  - Pieces misplaced in different puzzles
  - Extra pieces from other puzzles
- **Goal**: Define new quality metrics for Mixed-Bag puzzles

- **Standard Direct Accuracy**: Fraction of pieces, c placed in the same location in both the ground-truth and solved image versus the total number of pieces, n

$$DA = \frac{c}{n} \tag{5}$$

- **Standard Direct Accuracy**: Fraction of pieces, c placed in the same location in both the ground-truth and solved image versus the total number of pieces, n

$$DA = \frac{c}{n} \tag{5}$$

- **Enhanced Direct Accuracy Score (EDAS)**: Modified direct accuracy that accounts for missing and extra pieces.

$$EDAS_{P_i} = \underset{S_j \in S}{\arg\max} \frac{c_{i,j}}{n_i + \sum_{k \neq i}(m_{k,j})} \tag{6}$$

Quantifying Solver Quality

Standard and Enhanced Direct Accuracy

- **Standard Direct Accuracy**: Fraction of pieces, c placed in the same location in both the ground-truth and solved image versus the total number of pieces, n

$$DA = \frac{c}{n} \tag{5}$$

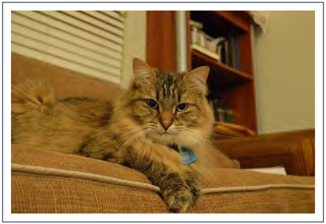
- **Enhanced Direct Accuracy Score (EDAS)**: Modified direct accuracy that accounts for missing and extra pieces.

$$EDAS_{P_i} = \underset{S_j \in S}{\arg\max} \frac{c_{i,j}}{n_i + \sum_{k \neq i}(m_{k,j})} \tag{6}$$

- **Direct Accuracy Range**: 0 to 1
- **Perfectly Reconstructed Image**: All pieces are placed in their original location ($DA = EDAS = 1$)

Problem: Direct accuracy is highly vulnerable to shifts, in particular when puzzle dimensions are not fixed

Problem: Direct accuracy is highly vulnerable to shifts, in particular when puzzle dimensions are not fixed

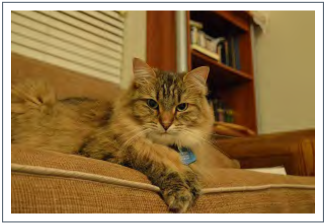


Ground-Truth Image

Direct Accuracy

Example – Effect of Shifts

Problem: Direct accuracy is highly vulnerable to shifts, in particular when puzzle dimensions are not fixed



Ground-Truth Image

Solver Output

Direct Accuracy

Example – Effect of Shifts

Problem: Direct accuracy is highly vulnerable to shifts, in particular when puzzle dimensions are not fixed

Ground-Truth Image

Solver Output

Conclusion: Direct accuracy can be overly punitive.

Direct Accuracy

Shiftable Enhanced Direct Accuracy Score (SEDAS)

- **Solution**: Allow the reference point for direct accuracy to **shift beyond the upper left corner** of the image
- **Shiftable Enhanced Direct Accuracy Score (SEDAS)**: Select the reference point, l, within radius d_{min} of the upper left corner of the solved puzzle
  - d_{min} – Manhattan distance between the upper left corner of the solved image and the nearest puzzle piece
- **Formal Definition of SEDAS**:

$$SEDAS_{P_i} = \arg\max_{l \in L} \left(\arg\max_{S_j \in S} \frac{c_{i,j,l}}{n_i + \sum_{k \neq i}(m_{k,j})} \right) \quad (7)$$

- **SEDAS Range**: 0 to 1

Direct Accuracy
Example – Shiftable Reference Point

Solver Output

Dept. of Computer Science
San José State University

Direct Accuracy

Example – Shiftable Reference Point

Solver Output

Direct Accuracy Reference Point

Direct Accuracy

Example – Shiftable Reference Point

Solver Output

SEDAS Reference Points

Quantifying Solver Quality

Standard and Enhanced Neighbor Accuracy

- **Standard Neighbor Accuracy**: Ratio of puzzle piece sides adjacent in both the original and solved images, a, versus the total number of sides, $n \cdot q$

$$NA = \frac{a}{n \cdot q} \tag{8}$$

Dept. of Computer Science
San José State University

- **Standard Neighbor Accuracy**: Ratio of puzzle piece sides adjacent in both the original and solved images, a, versus the total number of sides, $n \cdot q$

$$NA = \frac{a}{n \cdot q} \tag{8}$$

- **Enhanced Neighbor Accuracy Score (ENAS)**: Modified neighbor accuracy that accounts for missing and extra pieces.

$$ENAS_{P_i} = \underset{s_j \in S}{\arg\max} \frac{a_{i,j}}{q(n_i + \sum_{k \neq i}(m_{k,j}))} \tag{9}$$

Quantifying Solver Quality

Standard and Enhanced Neighbor Accuracy

- **Standard Neighbor Accuracy**: Ratio of puzzle piece sides adjacent in both the original and solved images, a, versus the total number of sides, $n \cdot q$

$$NA = \frac{a}{n \cdot q} \tag{8}$$

- **Enhanced Neighbor Accuracy Score (ENAS)**: Modified neighbor accuracy that accounts for missing and extra pieces.

$$ENAS_{P_i} = \underset{S_j \in S}{\arg\max} \frac{a_{i,j}}{q(n_i + \sum_{k \neq i}(m_{k,j}))} \tag{9}$$

- **Neighbor Accuracy Range**: 0 to 1
- **Advantage of Neighbor Accuracy**: Less vulnerable to shifts than direct accuracy

Experimental Results

Experimental Results

Experimental Results

- Paikin & Tal's algorithm is the current state of the art and was used as the reference for performance comparison

Experimental Results

- Paikin & Tal's algorithm is the current state of the art and was used as the reference for performance comparison
- **Standard Test Conditions**:
  - **Puzzle Type**: 2
  - **Dimensions Fixed**: No
  - **Piece Width**: 28 pixels
  - **Benchmark**: Twenty 805 piece images [6]

Experimental Results

- Paikin & Tal's algorithm is the current state of the art and was used as the reference for performance comparison
- **Standard Test Conditions**:
  - **Puzzle Type**: 2
  - **Dimensions Fixed**: No
  - **Piece Width**: 28 pixels
  - **Benchmark**: Twenty 805 piece images [6]
- **Number of Ground-Truth Inputs**: 1 to 5

# Puzzles	1	2	3	4	5
# Iterations	20	55	25	8	5

Experimental Results

- Paikin & Tal's algorithm is the current state of the art and was used as the reference for performance comparison
- **Standard Test Conditions**:
  - **Puzzle Type**: 2
  - **Dimensions Fixed**: No
  - **Piece Width**: 28 pixels
  - **Benchmark**: Twenty 805 piece images [6]
- **Number of Ground-Truth Inputs**: 1 to 5

# Puzzles	1	2	3	4	5
# Iterations	20	55	25	8	5

- **Test Condition Variation**: Only Paikin & Tal's algorithm was provided the number of input puzzles.

- **Goal**: Measure the Mixed-Bag Solver's accuracy determining the number of input puzzles
  - **Importance** – The Mixed-Bag Solver must estimate this accurately to provide meaningful outputs.

Experimental Results

Determining Input Puzzle Count

- **Goal**: Measure the Mixed-Bag Solver's accuracy determining the number of input puzzles
  - **Importance** – The Mixed-Bag Solver must estimate this accurately to provide meaningful outputs.
- **Single Puzzle Accuracy** – Represents the solver's performance ceiling
- **Multiple Puzzle Accuracy** – A more general estimate of the solver's performance

Determining Input Puzzle Count

Single Input Puzzle Results

- **Summary**: 17 out of the 20 images were correctly identified as a single ground-truth input
- **Misclassified Images**: 3 out of the 20 images misclassified as if they were two images.
  - All three images have large areas with little variation (e.g., a blue sky, smooth water)
  - The solver's poor performance on these puzzles is due to the assembler as noted in [4]

Determining Input Puzzle Count
Single Input Puzzle Results

- **Summary**: 17 out of the 20 images were correctly identified as a single ground-truth input
- **Misclassified Images**: 3 out of the 20 images misclassified as if they were two images.
  - All three images have large areas with little variation (e.g., a blue sky, smooth water)
  - The solver's poor performance on these puzzles is due to the assembler as noted in [4]
- **Note**: 85% (17/20) represents the accuracy ceiling when solving multiple puzzles.

Visual Comparison of a Misclassified Image

Perfectly Reconstructed Image (a)

Misclassified Image (b)

Determining Input Puzzle Count

Multiple Input Puzzles

- **Goal**: Measure the Mixed-Bag Solver's accuracy determining the input puzzle count for multiple images
- **Procedure**: Randomly select a specified number of images (between 2 and 5) from the 20 image data set.

Determining Input Puzzle Count

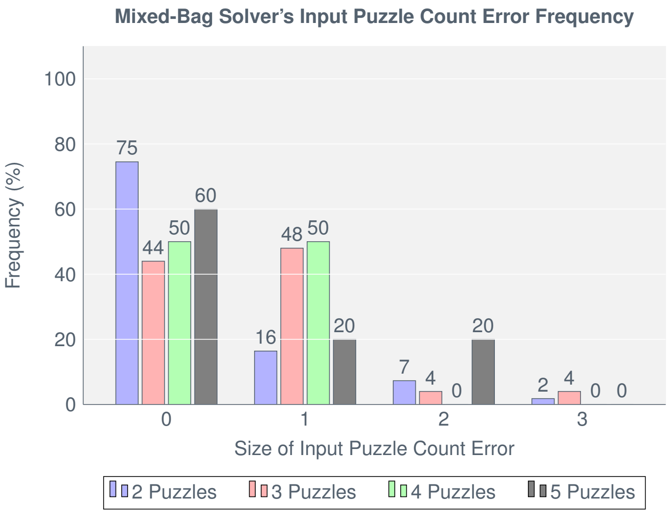
Multiple Input Puzzles

- **Goal**: Measure the Mixed-Bag Solver's accuracy determining the input puzzle count for multiple images
- **Procedure**: Randomly select a specified number of images (between 2 and 5) from the 20 image data set.
- **Input Puzzle Count Error**: Difference between the actual number of input puzzles and the number determined by the Mixed-Bag Solver.
  - **Example**: If 3 images were supplied to the solver but it determined there were 4, the error would be 1.

Dept. of Computer Science
San José State University

Determining Input Puzzle Count

Multiple Input Puzzles – Results



Determining Input Puzzle Count

Multiple Input Puzzles – Results Summary

- **Overall Accuracy**: 65%
- **Iterations with Error Greater than One**: 8%
- Accuracy did not significantly degrade as the number of input puzzles increased.

Determining Input Puzzle Count

Multiple Input Puzzles – Results Summary

- **Overall Accuracy**: 65%
- **Iterations with Error Greater than One**: 8%
- Accuracy did not significantly degrade as the number of input puzzles increased.
- **Over-Rejection of Cluster Mergers**: The Mixed-Bag Solver never underestimated the number of input puzzles.
  - Performance may be improved by reducing the minimum clustering similarity threshold or minimum segment size

- **Goal**: Compare the performance of the Mixed-Bag Solver (**MBS**) and Paikin & Tal's algorithm
- **Procedure:** Randomly select a specified number of images and input them into both solvers.
- **Quality Metrics Used**:
  - Shiftable Enhanced Direct Accuracy Score (SEDAS)
  - Enhanced Neighbor Accuracy Score (ENAS)
  - Perfect Reconstruction Percentage

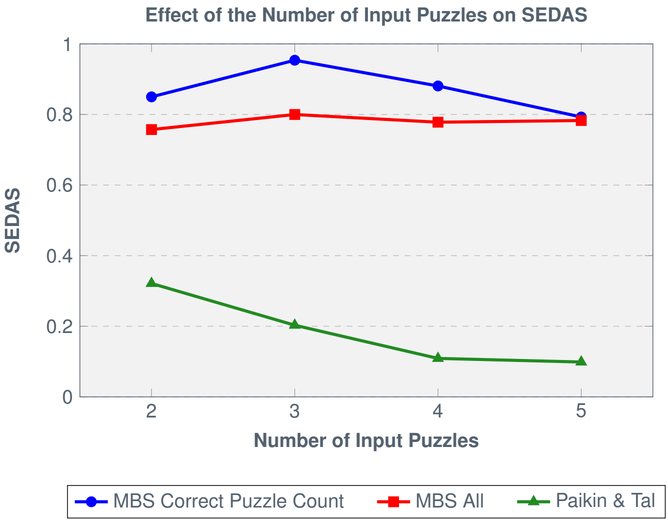
Experimental Results

Performance on Multiple Input Puzzles

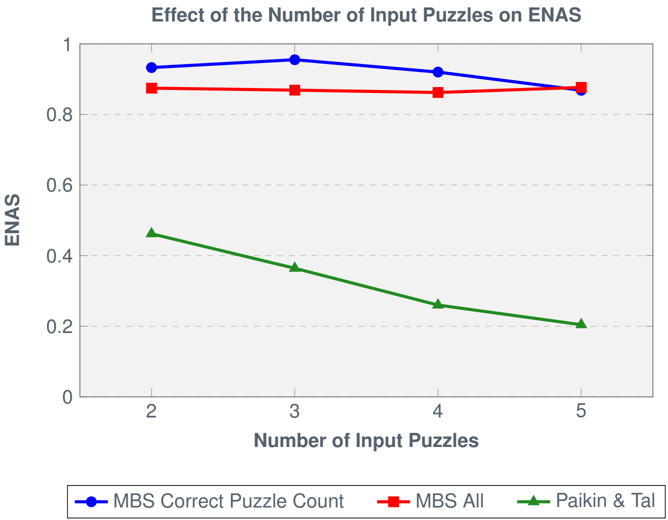
- **Goal**: Compare the performance of the Mixed-Bag Solver (**MBS**) and Paikin & Tal's algorithm
- **Procedure:** Randomly select a specified number of images and input them into both solvers.
- **Quality Metrics Used**:
  - Shiftable Enhanced Direct Accuracy Score (SEDAS)
  - Enhanced Neighbor Accuracy Score (ENAS)
  - Perfect Reconstruction Percentage
- **Note**: The results include the Mixed-Bag Solver's performance when it correctly estimated the puzzle count.
  - This represents the performance ceiling for optimal hierarchical clustering.

Performance on Multiple Input Puzzles

Shiftable Enhanced Direct Accuracy Score (SEDAS)

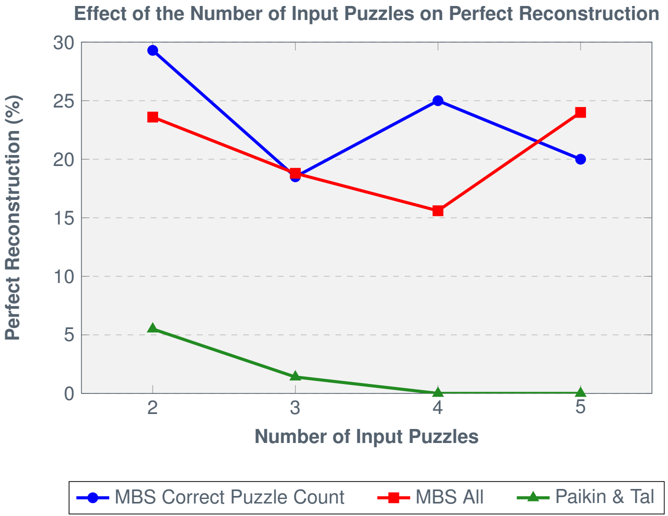


Performance on Multiple Input Puzzles
Enhanced Neighbor Accuracy Score (ENAS)



Performance on Multiple Input Puzzles

Perfect Reconstruction Percentage



Performance on Multiple Input Puzzles

Results Summary

- **Summary**: The Mixed-Bag Solver significantly outperformed Paikin & Tal's algorithm across all metrics.
  - This is notwithstanding that only their algorithm was supplied with the number of input puzzles.

Performance on Multiple Input Puzzles

Results Summary

- **Summary**: The Mixed-Bag Solver significantly outperformed Paikin & Tal's algorithm across all metrics.
  - This is notwithstanding that only their algorithm was supplied with the number of input puzzles.

- **Puzzle Input Count**: Unlike Paikin & Tal's algorithm, the Mixed-Bag Solver saw no significant decrease in performance with additional input puzzles

Performance on Multiple Input Puzzles

Results Summary

- **Summary**: The Mixed-Bag Solver significantly outperformed Paikin & Tal's algorithm across all metrics.
  - This is notwithstanding that only their algorithm was supplied with the number of input puzzles.
- **Puzzle Input Count**: Unlike Paikin & Tal's algorithm, the Mixed-Bag Solver saw no significant decrease in performance with additional input puzzles
- **Effect of Clustering Errors**: Performance only decreased slightly when incorrectly estimated input puzzle count.
  - Many of the extra puzzles were relatively insignificant in size

Conclusions

Conclusions

Conclusions

- This thesis presented a fully-automated solver for Mixed-Bag puzzles.
- Mixed-Bag Solver significantly outperforms the current state of the art while receiving no externally supplied information.
- Introduced the first set of solver quality metrics for Mixed-Bag puzzles.

Summary of Topics

Appendix

- Improved Assembler
  - Prioritize placement using multiple best buddies
  - Address placement performance in regions with low best buddy density

Conclusions

Future Work

- Improved Assembler
  - Prioritize placement using multiple best buddies
  - Address placement performance in regions with low best buddy density
- Dynamic determination of the segment clustering threshold

- Improved Assembler
  - Prioritize placement using multiple best buddies
  - Address placement performance in regions with low best buddy density
- Dynamic determination of the segment clustering threshold
- Expanded stitching piece selection

List of References I

[1] T. Altman, "Solving the jigsaw puzzle problem in linear time," *Applied Artificial Intelligence*, vol. 3, pp. 453–462, Jan. 1990.

[2] E. D. Demaine and M. L. Demaine, "Jigsaw puzzles, edge matching, and polyomino packing: Connections and complexity," *Graphs and Combinatorics*, vol. 23 (Supplement), pp. 195–208, June 2007.

[3] A. C. Gallagher, "Jigsaw puzzles with pieces of unknown orientation," in *Proceedings of the 2012 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, CVPR '12, pp. 382–389, IEEE Computer Society, 2012.

[4] G. Paikin and A. Tal, "Solving multiple square jigsaw puzzles with missing pieces," in *Proceedings of the 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, CVPR '15, IEEE Computer Society, 2015.

[5] Z. Hammoudeh, "Github - puzzle solver thesis repository." `https://github.com/ZaydH/Thesis`. (Accessed on 10/23/2016).

[6] D. Pomeranz, M. Shemesh, and O. Ben-Shahar, "Computational jigsaw puzzle solving." `https://www.cs.bgu.ac.il/~icvl/icvl_projects/automatic-jigsaw-puzzle-solving/`, 2011. (Accessed on 05/01/2016).

[7] A. Olmos and F. A. A. Kingdom, "McGill calibrated colour image database." `http://tabby.vision.mcgill.ca/`, 2005. (Accessed on 05/01/2016).

[8] P.-N. Tan, M. Steinbach, and V. Kumar, *Introduction to Data Mining, (First Edition)*. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 2005.

[9] T. S. Cho, S. Avidan, and W. T. Freeman, "A probabilistic image jigsaw puzzle solver," in *Proceedings of the 2010 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, CVPR '10, pp. 183–190, IEEE Computer Society, 2010.

[10] D. Pomeranz, M. Shemesh, and O. Ben-Shahar, "A fully automated greedy square jigsaw puzzle solver," in *Proceedings of the 2011 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, CVPR '11, pp. 9–16, IEEE Computer Society, 2011.

[11] H. Braxmeier and S. Steinberger, "Pixabay." `https://pixabay.com/`. (Accessed on 05/15/2016).

There are four primary jig swap puzzle types as formalized by [3]. In all cases, the “ground-truth” input is unknown.

55

There are four primary jig swap puzzle types as formalized by [3]. In all cases, the "ground-truth" input is unknown.

- **Type 1**: Puzzle dimensions and piece rotation are known. May have "anchor" piece(s) fixed in the correct location(s).

There are four primary jig swap puzzle types as formalized by [3]. In all cases, the "ground-truth" input is unknown.

- **Type 1**: Puzzle dimensions and piece rotation are known. May have "anchor" piece(s) fixed in the correct location(s).
- **Type 2**: All piece locations and rotations unknown. Puzzle dimensions may be known.

55

There are four primary jig swap puzzle types as formalized by [3]. In all cases, the "ground-truth" input is unknown.

- **Type 1**: Puzzle dimensions and piece rotation are known. May have "anchor" piece(s) fixed in the correct location(s).
- **Type 2**: All piece locations and rotations unknown. Puzzle dimensions may be known.
- **Type 3**: All piece locations are known. Only rotation is unknown.

55

There are four primary jig swap puzzle types as formalized by [3]. In all cases, the “ground-truth” input is unknown.

- **Type 1**: Puzzle dimensions and piece rotation are known. May have “anchor” piece(s) fixed in the correct location(s).
- **Type 2**: All piece locations and rotations unknown. Puzzle dimensions may be known.
- **Type 3**: All piece locations are known. Only rotation is unknown.
- **Mixed-Bag**: Pieces come from multiple puzzles

There are four primary jig swap puzzle types as formalized by [3]. In all cases, the "ground-truth" input is unknown.

- **Type 1**: Puzzle dimensions and piece rotation are known. May have "anchor" piece(s) fixed in the correct location(s).
- **Type 2**: All piece locations and rotations unknown. Puzzle dimensions may be known.
- **Type 3**: All piece locations are known. Only rotation is unknown.
- **Mixed-Bag**: Pieces come from multiple puzzles

Mixed-Bag puzzles are the focus of this thesis.

Previous Work

- **Cho *et al.*** [9] – Introduced the first modern jig swap puzzle solver
  - Graphical model-based Type 1 solver
  - Puzzle dimensions are known
  - Used one or more anchor pieces
  - Defined quality metrics for Type 1 and Type 2 puzzles
  - Established the standard comparative test conditions

Previous Work

- **Cho *et al.*** [9] – Introduced the first modern jig swap puzzle solver
  - Graphical model-based Type 1 solver
  - Puzzle dimensions are known
  - Used one or more anchor pieces
  - Defined quality metrics for Type 1 and Type 2 puzzles
  - Established the standard comparative test conditions
- **Pomeranz *et al.*** [10] – Iterative, greedy Type 1 puzzle solver
  - Eliminated the use of anchor pieces
  - Created multiple solver benchmarks of various sizes

- **Basis of all Modern Jig Swap Solvers**: The more compatible two pieces are, the more likely they are to be adjacent.
- **Best Buddies**: A pair of puzzles pieces that are more compatible with each other on their respective sides than they are to any other piece [10]
  - **Note**: Not all puzzle pieces will have a best buddy.

$$\forall p_k \forall s_z, C(p_i, s_x, p_j, s_y) \geq C(p_i, s_x, p_k, s_z)$$

$$\text{and} \tag{10}$$

$$\forall p_k \forall s_z, C(p_j, s_y, p_i, s_x) \geq C(p_j, s_y, p_k, s_z)$$

- **Importance of Best Buddies**: Key adjacency indicator

58

- **Best Buddy Density** (BBD): A metric for quantifying the best buddy profile of an image that is independent of image size.

$$BBD = \frac{b}{n \cdot q} \tag{11}$$

- A greater BBD means the pieces are more differentiated making the puzzle easier to solve.

Visualizing Best Buddy Density

- Transform each puzzle piece into a square consisting of four isosceles triangles.
- Color each triangle according to whether the adjacent piece is a best buddy. The scheme used in this thesis:

No Best Buddy	Non-Adjacent Best Buddy	Adjacent Best Buddy	No Piece Present
(white)	(red)	(green)	(black)

- Areas with higher best buddy density will have more green triangles.

Best Buddy Density

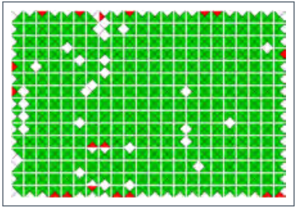
Visualization Example

Original Image [11]

Best Buddy Density

Visualization Example

Original Image [11]



Best Buddy Visualization

Determining Input Puzzle Count

Comparison of Best Buddy Density for Misclassified Images

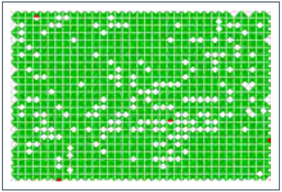
Perfectly Reconstructed Image (a)

Misclassified Image (b)

Determining Input Puzzle Count

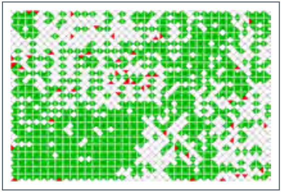
Comparison of Best Buddy Density for Misclassified Images

Perfectly Reconstructed Image (a)



Best Buddy Visualization (a)

Misclassified Image (b)



Best Buddy Visualization (b)

- As the number of puzzles increases, the difficulty of simultaneously reconstructing them also increases.

- As the number of puzzles increases, the difficulty of simultaneously reconstructing them also increases.

- **Current State of the Art**: Paikin & Tal [4] solved up to five puzzles simultaneously.

- As the number of puzzles increases, the difficulty of simultaneously reconstructing them also increases.
- **Current State of the Art**: Paikin & Tal [4] solved up to five puzzles simultaneously.
- **Goal**: Compare the performance of the Mixed-Bag Solver and Paikin & Tal's algorithm on 10 puzzles.

- **Paikin & Tal**
  - Seed of nine images came from just three input images
  - SEDAS and EDAS greater than 0.9 for only one image
  - No perfectly reconstructed images

Ten Puzzle Results

Summary

- **Paikin & Tal**
  - Seed of nine images came from just three input images
  - SEDAS and EDAS greater than 0.9 for only one image
  - No perfectly reconstructed images
- **Mixed-Bag Solver**
  - SEDAS and EDAS greater than 0.9 for all images
  - Four images perfectly reconstructed
  - Results comparable to Paikin & Tal's algorithm solving each puzzle individually