Consistent Correspondences for Shape and Image Problems

Taiwei Wang

Submitted to Swansea University in fulfilment

of the requirements for the Degree of Doctor of Philosophy

Swansea University

Prifysgol Abertawe

Department of Computer Science

Swansea University

October 12, 2020

Declaration

This work has not been previously accepted in substance for any degree and is not being concurrently submitted in candidature for any degree.

Signed ..... (candidate)

Date .....

Statement 1

This thesis is the result of my own investigations, except where otherwise stated. Other sources are acknowledged by footnotes giving explicit references. A bibliography is appended.

Signed ..... (candidate)

Date .....

Statement 2

I hereby give consent for my thesis, if accepted, to be available for photocopying and for inter-library loans after expiry of a bar on access approved by the Swansea University.

Signed ..... (candidate)

Date .....

Abstract

Establish consistent correspondences between different objects is a classic problem in computer science/vision. It helps to match highly similar objects in both 3D and 2D domain. In the 3D domain, finding consistent correspondences has been studying for more than 20 years and it is still a hot topic. In 2D domain, consistent correspondences can also help in puzzle solving. However, only a few works are focused on this approach. In this thesis, we focus on finding consistent correspondences and extend to develop robust matching techniques in both 3D shape segments and 2D puzzle solving. In the 3D domain, segment-wise matching is an important research problem that supports higher-level understanding of shapes in geometry processing. Many existing segment-wise matching techniques assume perfect input segmentation and would suffer from imperfect or over-segmented input. To handle this shortcoming, we propose multi-layer graphs (MLGs) to represent possible arrangements of partially merged segments of input shapes. We then adapt the diffusion pruning technique on the MLGs to find consistent segment-wise matching. To obtain high-quality matching, we develop our own voting step which is able to remove inconsistent results, for finding hierarchically consistent correspondences as final output. We evaluate our technique with both quantitative and qualitative experiments on both man-made and deformable shapes. Experimental results demonstrate the effectiveness of our technique when compared to two state-of-art methods. In the 2D domain, solving jigsaw puzzles is also a classic problem in computer vision with various applications. Over the past decades, many useful approaches have been introduced. Most existing works use edge-wise similarity measures for assembling puzzles with square pieces of the same size,

and recent work innovates to use the loop constraint to improve efficiency and accuracy. We observe that most existing techniques cannot be easily extended to puzzles with rectangular pieces of arbitrary sizes, and no existing loop constraints can be used to model such challenging scenarios. We propose new matching approaches based on sub-edges/corners, modelled using the MatchLift or diffusion framework to solve square puzzles with cycle consistency. We demonstrate the robustness of our approaches by comparing our methods with state-of-art methods. We also show how puzzles with rectangular pieces of arbitrary sizes, or puzzles with triangular and square pieces can be solved by our techniques.

Acknowledgements

I would like to thank my supervisor Dr. Gary K.L. Tam for guiding me through my Doctor of Philosophy and Master of Science studying. When I started my study, I have limited background in computer science. Thanks to Gary who have been mentoring and teaching me how to do academic research. I also want to thank Dr. Xianghua Xie, Dr. David George, Dr. Yukun-Lai, and Mr. Kristiyan Vladimirov, for providing help during my study. Most importantly, I thank my parents for supporting my living costs in these years.

Acronyms

DP Diffusion Pruning

EMD Earth Mover's Distance

HKS Heat Kernel Signature

HSV colour space of Hue Saturation Value

ICP Iterative Closest Point

LAB colour space defined by International Commission on Illumination

LFD Light Field Descriptor

MGC Mahalanobis Gradient Compatibility

MLG Multi-layer Graph

MRI Magnetic Resonance Imaging **MST** Minimum Spanning Tree

PCA Principle Component Analysis

PSD Positive Semi-definite

QP Quadratic Programming

RGB colour space of Red Green Blue

SDP Semi-definite Programming

SFM Structure From Motion

SHED Shape Edit Distance

SIFT Scale-invariant Feature Transform

SSD Sum of Squared Distance

Publications

Taiwei Wang, David George, Yu-Kun Lai, Xianghua Xie, and Gary Tam. Consistent segment-wise matching with multi-layer graphs. In *Computer Graphics and Visual Computing (CGVC)*. The Eurographics Association, September 2018. Computer Graphics Visual Computing (CGVC) 2018 ; Conference date: 13-09-2018 Through 14-09-2018.

Taiwei Wang, David George, Yu-Kun Lai, Xianghua Xie, and Gary K.L. Tam. Consistent segment-wise matching with multi-layer graphs. *Computer Aided Geometric Design*, 70:31 – 45, 2019.

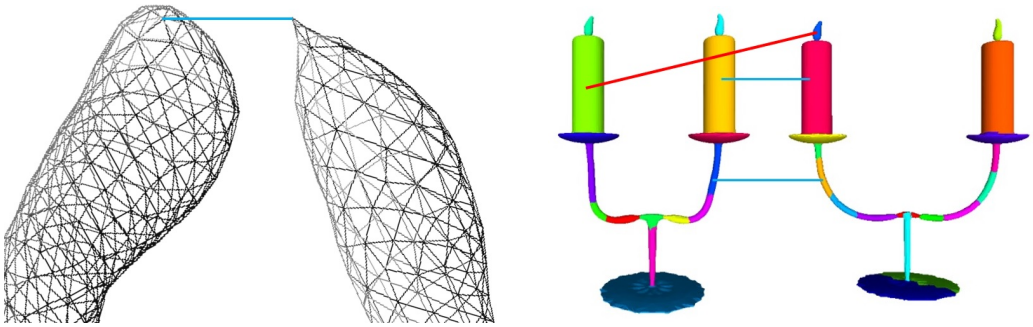
TAIWEI WANG, Kristiyan Vladimirov, Shu Yu Goh, Yukun Lai, Xianghua Xie, and Gary K. L. Tam. Robust and Flexible Puzzle Solving with Corner-based Cycle Consistent Correspondences. In Franck P. Vidal, Gary K. L. Tam, and Jonathan C. Roberts, editors, *Computer Graphics and Visual Computing (CGVC)*. The Eurographics Association, 2019.

Table of Contents

Chapter 1

Introduction

Contents



(a) Vertex/point-to-vertex/point correspondence

(b) Region/segment-to-region/segment correspondences

Figure 1.1: An example of correct/good/meaningful correspondences: the two vertices/segments been matched are located at the similar positions or regions on the two shapes/meshes. In (a), two vertices are located at the top of the objects, which is a correct correspondence (shown in the blue line). In (b), candle body is matched to the flame, which is a incorrect correspondence (shown in the red line).

Finding correspondences is a classic problem in 2D/3D analysis domain[1]. As shown in Figure 1.1a, a correspondence can be visualised as a line that indicates the matching information between two objects. Consistent correspondences show the similar topological/geometrical information from the two objects (the blue lines in Figure 1.1b), and a bad/inconsistent one indicates wrong matched information from objects (the red line in 1.1b).

Based on the objects that are linked/matched, we can category the finding of correspondences into two sections: point-wise registration and segment-wise shape matching. The registration techniques are focused on the point-wise matching of individual objects, such as a pair of vertices. The segment-wise matching techniques are computing regions, such as the matching between a set of vertices to another set of vertices.

1.1 Point-wise Registration

Registration techniques are based on correspondences to align different objects under the same coordinate system. The objects that have aligned can be in whether 3D or 2D. The goal of 3D registration is to compute how to align and overlap one 3D object (such as 3D model or mesh)

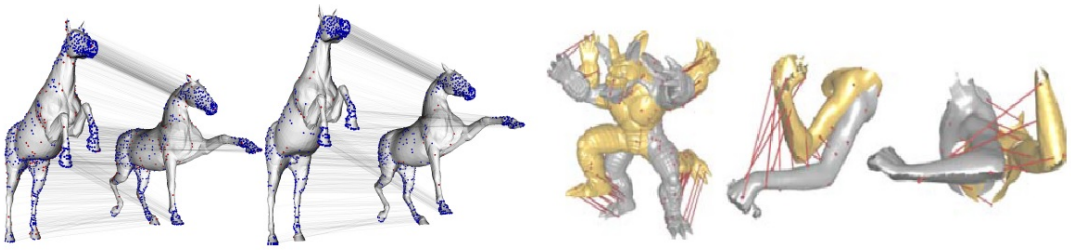


Figure 1.2: The goal of 3D registration is to find the alignment between different 3D models [2, 3]. The models can be point cloud, polygon mesh, or other kinds of format. Recent works are focused on deformable models alignment.

on another one. This alignment can be in both rigid (no deformation between 3D objects) or non-rigid (deformable 3D objects). The goal of 2D registration is to combine different 2D objects (such as a set of images) to construct a scene, such as MRI (Magnetic Resonance Imaging) scans.

1.1.1 3D Surfaces Registration

In the past 20 years, various 3D surfaces registration techniques have been developed. [4, 5] introduce the classic ICP (Iterative Closest Point). The iteratively computed closest point is the constraint for correspondence alignment between two rigid surfaces. Variant works followed this approach with improved performance in speed or accuracy [6, 7]. Later, [8] introduces non-rigid ICP for non-rigid surface by adding in a registration error metric. Other works, such as [9] have followed this approach. [10] introduced the classic spectral analysis technique. It uses pairwise information of input correspondences to build affinity matrix and then run eigen-decomposition to find eigenvectors. The values in the largest eigenvector are used to evaluate the confidence of each input correspondence. Input correspondence with high confidence will be selected as output matching. This approach has been widely used in 3D non-rigid registration [2, 11].

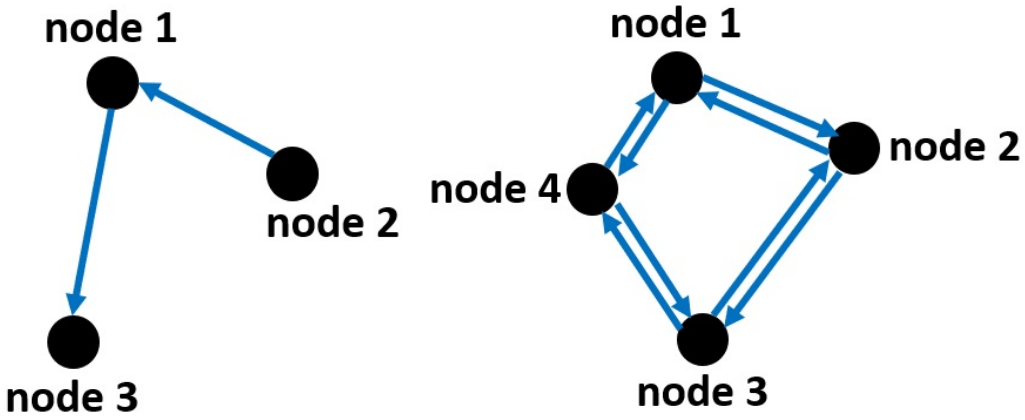
Divert from spectral analysis diffusion analysis is also based on eigenvectors and is used for computing correspondences. It has good local-consistency preservation which allows us to infer the globally consistent correspondences. [3] demonstrated good performance of using

diffusion pruning in point-based shape matching. It involved local-consistency constraint in affinity matrix construction. Furthermore, there is a designated pruning algorithm with local-consistency to ensure good outputs. During the pruning, the new candidate correspondence will only be accepted if it does not conflict with each already accepted correspondences, and if it fits local-consistency constraint. [12, 9] have followed diffusion pruning and show good performance in solving the point-wise matching problem.

Different from local consistency, cycle-consistency is also an important constraint in correspondence computing. Figure 1.3 shows two different definitions of cycle-consistency. First, cycle-consistency can be formed as a multi-way incomplete flow of correspondences between all input objects. In Figure 1.3a there is no direct correspondences between node 2 and node 3. However, they have been linked/matched through node 1. Second, the closed cycle-consistency is defined by a complete dual-flow between all input objects. The dual-flow can be considered as a forward-backwards matching between each pair of objects. In Figure 1.3b, each pair of nodes have been matched by two correspondences in a forward-backwards manner. Closed loops ensure stronger cycle-consistency than open loops. The cycle-consistency computation requires a collection of correspondences as input and a designated objective function to evaluate the displacement between input correspondences and form cycle-consistency outputs [13, 14]. Another approach is using mathematical modelling to find cycle-consistent correspondences from inputs [15, 16].

1.1.2 2D Images Registration

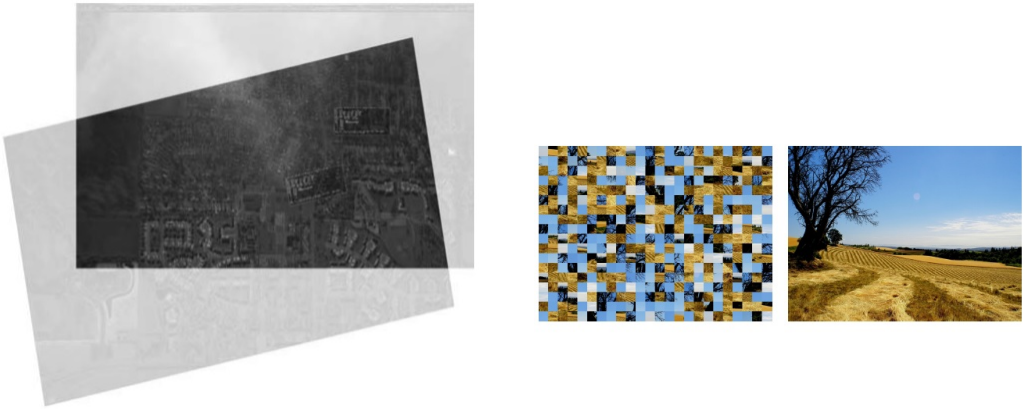
2D Image registration is a popular research domain. The goal of 2D registration is to combine/group different images into one complete image or scene by computing the correspondences between images (for example, the registration of multimodal images). In the medical industry, MRI and CT (Computed tomography) scans are essential for doctors' diagnosis. The scans output a set of images that are required to reconstruct to form a complete image/scene of the object, which is supported by computing 2D registration between each scanned image



(a) Multi-way cycle-consistency with open loops

(b) Two-way cycle-consistency with closed loops

Figure 1.3: An example of cycle-consistency.



(a) Align/merge two images into one for finding a complete scene of the view [19].

(b) Align image pieces to form the original image [20].

Figure 1.4: The goal of 2D image registration is to use the alignment of different images to construct a new scene/image.

[17]. In the past 20 years, the usage rate of 2D image registration has significantly increased in clinical practice [18]. Multimodal images registration also benefits other domains, such as combine aerial photos to form a map [19], and mosaic imaging.

Register images can also help in solving jigsaw puzzles. In 1964, [21] introduces the first work that is using a computer to solve the jigsaw puzzle problem. It computes the 2D registration between puzzle pieces (each puzzle piece can be considered as an image) to find the best alignment and solve the puzzle. In the early stage, works of using a computer to the

solve jigsaw puzzle problem are using the shape as the only constraint. There was no pattern on the puzzle pieces. Today, techniques for solving jigsaw puzzle are focused on how to use shredded image pieces to recover the original image/pattern. These works use the colour (the colour of each pixel on each puzzle piece) as the constraint to find the similarity between puzzle pieces. Then use a designated grouping/placement technique to find the correct placement of each puzzle piece to solve the puzzle.

There are two types of placement techniques: the global approach [22] and the greedy approach [20]. The global approach places the pieces by a global optimisation function which based on the similarity between each pair of pieces. The greedy approach based on the similarity between pieces to group the most similar pieces, and then group the next most similar pieces until all pieces are placed. The recent works of solving puzzles are focused on greedy approaches.

1.1.3 Applications

Point-wise registration techniques have been widely applied in both 3D and 2D to solve problems of object recognition, model reconstruction, multimodal registration.

Object recognition is essential in many industries, such as the processing of medical data, facial recognition, and object retrieval. Registration techniques output high accurate images to help object recognition in medical imaging and benefit doctors' diagnosis, such as better understanding of patient's organ [23, 24] and better imaging of MRI scans [25]. Facial recognition can detect and recognise a human face from different angle of views and variant illumination environment. It can be used to identify a person with interest from camera footage, cognitive assistance, and security problems [26, 27]. Object retrieval can identify a certain object (such as a 3D model or a semantic definition) from a collection of various similar objects. It can help the understanding of 3D scenes such as identify a tree from a landscape photo [28], and find a 3D model from data sets to achieve 3D models classification [29].

Model reconstruction is based on incomplete information (such as partially 3D scanned

data of a 3D model) to generate a complete 3D model. It is useful in the field of cultural heritage, engineering. The model reconstruction helps the understanding of real-world architecture and archaeology, and the recover ancient archaeology relics[30]. In the engineering domain, model reconstruction can help reverse engineering [31] and rapid manufacturing/prototyping [32] in industries production.

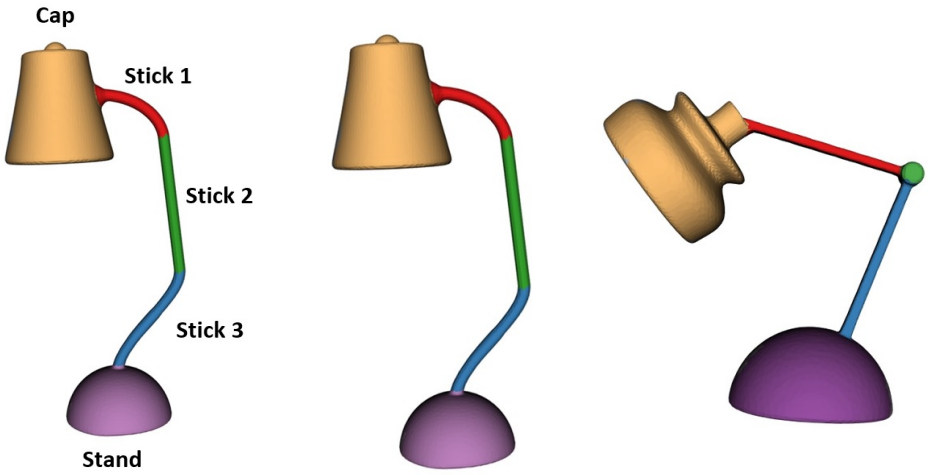
Multimodal registration can combine different image to form a complete object. As mentioned in the section 1.1.2 Multimodal registration is important and useful in processing medical data such as MRI and CT scans, map generation, and mosaic imaging. Multimodal registration can also help in solving jigsaw puzzle, which is important in archaeology research [33]. The broken ancient relics can be quickly restored by using puzzle solving techniques. Still, puzzle solving techniques can also be applied in recovering the shredded files or evidence for the police procedural.

1.2 Segment-wise Shape Matching

Segment-wise matching techniques are designed for understanding objects by matching parts of objects. Psychological research shows that people are using the shape of parts and their structural/spatial information to recognise and understand a 3D object [34]. For example, as shown in Figure 1.5 a lamp can be decomposed into a stand, several sticks, and a cap. We use the shape of these decomposed parts to identify their functions, and then we can infer it is a lamp. Another example, the function of military camouflage is to split the object into irregular shapes, to confuse enemy reconnaissance.

1.2.1 Registration Approach

Segment-wise matching techniques are based on point-wise registration methods. As mentioned in the section 1.1.1, spectral analysis [10] is a classic technique in solving the registration problem, and it is also popular in solving the segment-wise matching problem. [35] takes symmetric shape segments (such as a pair of hands or arms) with their spatial/structural in-



(a) A lamp can be decomposed into different parts in human understanding.

(b) Matching similar parts can help the recognition and understanding between different shapes.

Figure 1.5: The shape recognition and understanding of mankind are based on shape decomposition.

formation as input, then uses spectral analysis to find consistent correspondences between the symmetric segments. [36] introduces SHED, which is also using spectral analysis for finding consistent segment-wise matching between input shapes. They use pre-segmented shapes with segments spatial/structural information as input and then use a designated iteratively approach to find the most consistent and reliable correspondences. Based on these confident correspondences, they can further compute the classification of input shapes.

The techniques introduced above are heavily based on spatial/structure information of input segments, which is an important constraint in the segment-wise matching domain. In the 3D registration domain, the usage of point-wise distance (such as the shortest path between a pair of vertices of a mesh) is a popular constraint for finding consistent matching between objects. Segment-wise matching also requires distance to evaluate the consistency between a pair of segments. Since it is difficult to define the distance between a set of vertices and another set of vertices, the spatial/structural information between segments can be used to evaluate the consistency between them.

1.2.2 Hierarchical Approach

In recent works, since spatial/structural information is an essential constraint in the segment-wise matching domain, the hierarchical approach has become popular in solving segment-wise matching problem. [37] shows that they based on the spatial/structural to recursively decompose the input shapes into parts, and then how to use deformation to compute the segment-wise matching. [38] based on the deformation between segments to compute their spatial/structural information, and then use them to compute the topological variants for finding consistent segment-wise matching. In these works, the spatial/structural information can be considered as a constraint that has limited a specific searching space of finding segment-wise matching.

1.2.3 Applications

Segment-wise matching is useful in shape analysis, which includes shape segmentation, shape matching, shape retrieval and classification.

Segment-wise matching is useful in archaeology research. [39] shows the matching between pieces of ancient artefacts can indicate the relationship with the existing ones for further study. Still, segment-wise matching can also handle broken objects, which means it can be applied in ancient artefacts and documents recovery. Similar applications can also help the research of palaeontology domain.

Segment-wise matching can improve the performance of segmentation techniques, which can further help in other applications in the different industries (such as the processing of medical data). For example, [40] introduces a segmentation technique with cycle-consistency segment-wise matching to find globally confident segments. Good segments can help medical imaging and benefits doctors' diagnosis.

Another application of segment-wise matching is shape classification and retrieval. [36] introduces a shape classification technique by using segment-wise matching to determine the distance/similarity between input shapes. [41] introduces how to use segment-wise matching and spatial/structural information to retrieval shapes. Shape classification and retrieval are

important for the management of nowadays online 3D model warehouse.

1.3 Motivation

1.3.1 Problem Observations

In this section, we are going to discuss the problems we have observed from existing works and what hypothesis/research questions we have made. We found three problems in both 3D and 2D domain. We use our hypothesis to answer each problem that we have found. For the first problem and its corresponding hypothesis, we show a detailed solution in Chapter 3. In Chapter 4 and 5 we show detailed solution for our second and third problem/hypothesis.

1.3.2 Problem 1: Over/Imperfect Segmentation

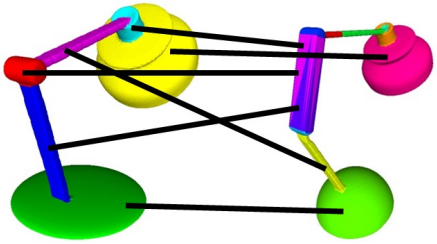
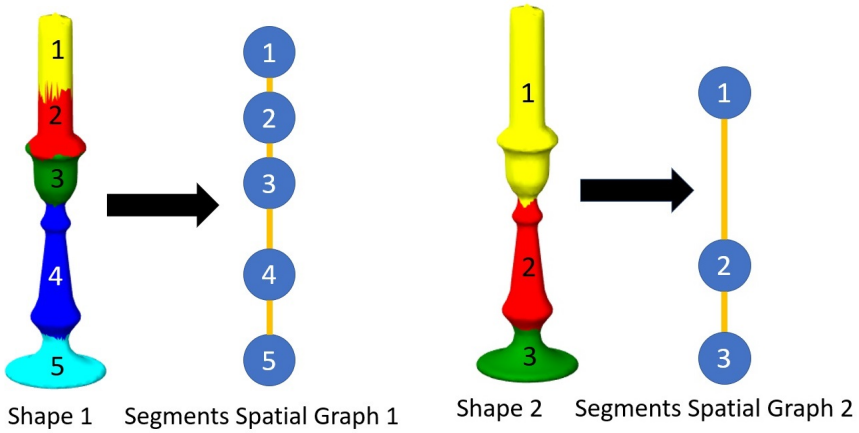


Figure 1.6: Matching results from combining [36] and [13]. Only the bases of lamps are correctly matched.

The first problem we have observed is that imperfect/over-segmentation is challenging existing shape matching techniques. In our early experiments we have combined [36] and [13] to test cycle-consistent segment-wise matching. [36] shows good performance of segment-wise matching and [13] has demonstrated cycle-consistent point-wise correspondences can be found based on reasonable input matching. We replaced the point-wise matching part in [13] by using the segment-wise matching technique from [36]. We hope to find high-quality cycle-consistent correspondences between shape segments from our modified technique of [13]. However, the results are not consistent, see Figure 1.6.

We realised that the graph distance caused incorrect results. In [36], it takes shape segments as input and builds a graph to represent the spatial relation of the original shape. Divert from the



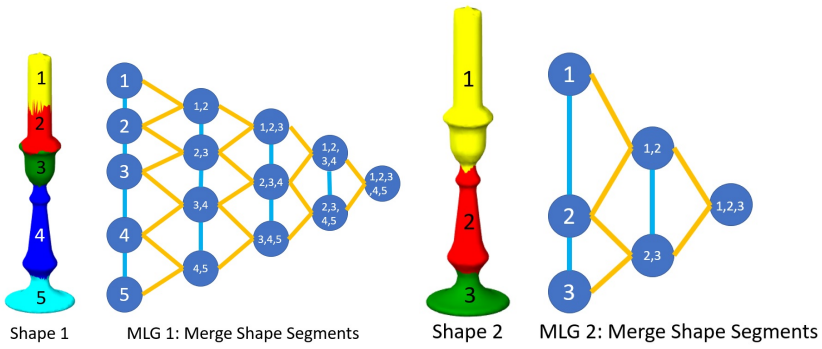
(a) Segments and spatial graph of shape 1 (b) Segments and spatial graph of shape 2

Figure 1.7: An example of over/imperfect segmentation. The more segments lead to larger spatial graph of segments.

point-wise matching in [13] segment-wise matching has a low range of distance distribution. It leads to high sensitivity of correspondences computing in [13]. Still, over/imperfect segmentation also leads to large graph distance variation between a pair of similar inputs. For example, in Figure 1.7, the shape 1 has been over segmented (two shapes are identical). In Figure 1.7a the shape has more segments and will generate a graph that larger than the graph in Figure 1.7b. In Figure 1.7a, the graph distance/shortest path between cap and base is significantly longer than the distance between cap and base in Figure 1.7b. Around the over segmented regions in Figure 1.7a the graph distances are also more multiplex than Figure 1.7b. As a result, these unreliable graph distance caused correct matching results.

We make our first hypothesis to handle this issue.

- Can a technique that handles moderate topological changes in the underlying segment graphs improve matching results?
- Can merged segments help improve the accuracy of segment-wise matching with inconsistent (over-/imperfectly) segmented inputs?
- How can we develop a representation that facilitates matching of merged segments, and



(a) Merge segments from shape 1 (b) Merge segments from shape 2

Figure 1.8: An example of merging: by combining input segments we can generate new merged segments. The merged segments help in finding the meaningful matching with a hierarchical understanding between over/imperfect segmentation regions. For example, in (a) the merged segment (1,2,3) can match to segment 1 in (b).

a technique for robust segment-wise matching?

We use a merging procedure to generate MLGs (multi-layer graph) for solving over/imperfect segmentation problem in shape matching. As shown in Figure 1.8, based on the spatial graph of input segments, and we merge segments to generate new layers that contain new 3D segments/shape parts. The merged segments are either spatially adjacent segments or segments that have shared faces (two segments are touching each other). To match consistent regions in MLGs, we can use diffusion pruning from [3] since it has good local-consistency in matching problems. The detailed method explanation and experiment results are shown in Chapter 3.

1.3.3 Problem 2: Variously Shaped Jigsaw Puzzle Pieces

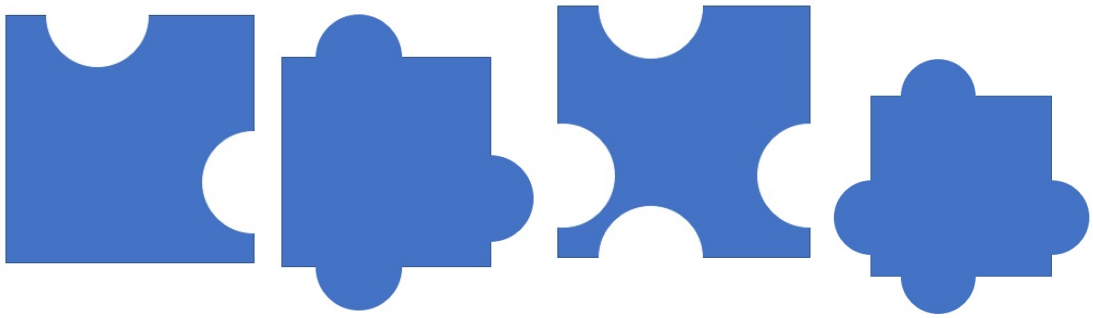
Solving jigsaw puzzle is also an old problem in computer science/vision since. Table 1.1 shows a brief survey of existing puzzle solving works. In 1964 [42] shows the first work that uses a computer to solve a nine pieces jigsaw puzzle. Since then, variously works have been introduced to solve jigsaw puzzles. In the early stage, researchers were computing how to assemble pieces (the assembly stage) since there is no pattern on each puzzle piece. They use the shape as the constraint to compute the similarity between input puzzle pieces. By using the geometry information of the edge on each piece, the placement information can be computed.

Paper	Year	Shape	Colour	Square Pieces	Unknown Rotation	Loop Constraint	Puzzle Pieces
[42]	1964	Yes	No	No	No	No	9
[43]	1988	Yes	No	No	No	No	104
[44]	1991	Yes	No	No	No	No	24
[45]	1994	Yes	No	No	No	No	54
[46]	1998	Yes	Yes	No	No	No	54
[47]	2001	Yes	No	No	No	No	N/A
[48]	2003	Yes	Yes	No	No	No	23
[49]	2006	Yes	Yes	No	No	No	7
[50]	2006	Yes	Yes	No	No	No	21
[51]	2008	Yes	Yes	No	No	No	320
[52]	2009	No	Yes	Yes	No	No	100
[22]	2010	No	Yes	Yes	No	No	432
[53]	2011	No	Yes	Yes	No	No	108
[54]	2011	No	Yes	Yes	No	No	3300
[20]	2012	No	Yes	Yes	Yes	No	9600
[55]	2012	No	Yes	Yes	Yes	No	432
[56]	2013	No	Yes	Yes	No	No	432
[57]	2013	No	Yes	Yes	No	No	22834
[58]	2014	No	Yes	Yes	Yes	Yes	9801
[59]	2015	No	Yes	Yes	Yes	No	22834
[60]	2015	Yes	Yes	N/A	No	No	N/A
[61]	2016	No	Yes	Yes	Yes	Yes	3300
[62]	2016	No	Yes	Yes	Yes	No	22834
[63]	2016	No	Yes	Yes	Yes	No	100
[64]	2018	No	Yes	Yes	Yes	Yes	9801

Table 1.1: A simple survey of existing puzzle solving techniques.

For example, in figure 1.9a, if a piece has only two curved edges, then it is a corner piece. In Recent works, puzzle pieces are in squares with the same size. Still, pieces are coloured so that they can form a pattern. Same size square pieces have equivalent straight edges and there is no other information that can be used in similarity computation except the colour. In the correct pattern, each piece has only one correct placement. It means square pieces are more challenging than traditional pieces in puzzle solving. It is why the recent works are using colour as the constraint to solve puzzles. Today, only a few techniques can handle rectangular pieces and most recent works are only available for square pieces with a fixed edge length.

The second problem we have observed is that recent works can only handle square pieces



(a) A corner piece (b) A piece at border (c) A inner piece with blanks (convex regions) (d) A inner piece with tabs (convex regions regions)

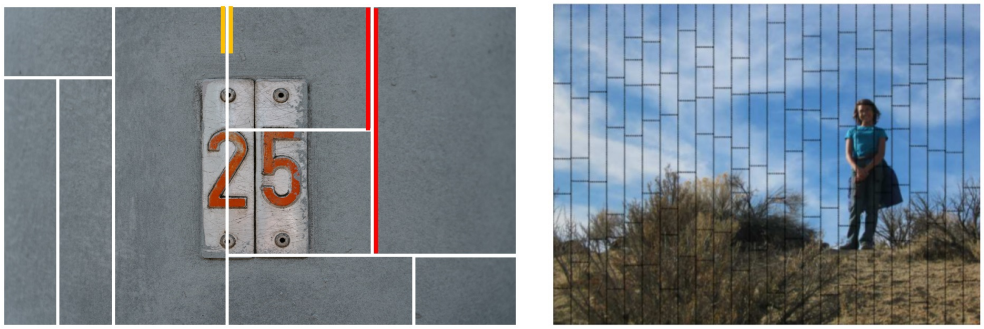
Figure 1.9: Using geometry information in puzzle solving.

with the same size. As shown in Figure 2.1 and discussed in section 2.2.3, current works use the whole edge of each puzzle piece to compute similarity scores. Thus, the colour matrices should have the same dimensions. It leads to a fixed number of pixels on piece edges, which means the same size of puzzle pieces.

We make our second hypothesis:

- Can we develop a more flexible technique that is based on partial information from a whole edge for puzzle solving of rectangle pieces of arbitrary sizes?

As shown in Figure 1.10a, our input puzzle has rectangular pieces with different sizes. If we break/subdivide the whole edge into sub-edges, we can ensure the length of sub-edges (the number of pixels in each sub-edge) is constant. Based on the sub-edges, it is possible to solve puzzles with differently shaped pieces. To the best of our knowledge, there is no existing work that can solve puzzles with rectangular pieces with different edge length. [65] is the only work that closes to our goal. It demonstrated a method that can handle pieces like Brick Walls, where each rectangular piece has a different height and a fixed width, shown in Figure 1.10b. The Brick Walls pieces are still less challenge than our case since in Figure 1.10a we have different length of both height and width in each piece.



(a) Rectangular pieces with different edge-length of both height and width

(b) Brick Wall pieces have a fixed length of width[65]

Figure 1.10: In (a) we use sub-edge (orange lines) instead of the whole edge (red lines) to ensure the same edge-length for similarity measurement between puzzle pieces. We can handle edge-length variation in both height and width, which is more challenging than (b).

1.3.4 Problem 3: Solving Jigsaw Puzzles with Shape Matching Techniques

The third problem we have observed is that 3D shape matching techniques can also be applied in solving jigsaw puzzles. We have noticed that the second step (assembling) can be considered as a matching problem in terms of high-quality correspondences. Still, matching techniques can be adapted in both 3D and 2D domain. For example, functional maps [66] produces good results in 3D shape analysis [67, 68, 69] and [70] shows functional maps can also be adapted in 2D image processing domain with good performance. To the best of our knowledge, only a few works are using cycle-consistency as a constraint in puzzle solving, and diffusion framework has not been considered. Thus, by using an appropriate 3D shape matching technique, we can assemble puzzle pieces in a novel way.

As mentioned in Chapter 1.1.1, cycle-consistency is a useful constraint for the problem of matching multiple objects. For example, in Figure 1.3, given three objects A, B and C, cycle consistency enforces matchings from A to B, and from B to C such that C to A is also meaningful. In puzzle solving the piece-wise similarity computation with backwards-difference estimation is also related to two-way cycle-consistency [54, 59]. The loops constraint in [58, 64] can be considered as a designated four-way cycle-consistency for assembling puzzle pieces.

We make our third hypothesis.

- Can we use cycle-consistency as the constraint to compute piece-wise correspondences for puzzle solving?

From our experiments, under some extreme cases, the existing works only have 30% precision rate in similarity scores. Cycle-consistency ensures that we can locate and match globally confident edges of puzzle pieces, to serve/benefit assembling step and avoid incorrect results from local noise. Local consistency can also perform good edge-wise matching results by ensures the accuracy of each correspondence. For global case, we hope by using MatchLift [16] we can find globally cycle-consistent pieces from noisy input. The reason for using MatchLift is that it has up to 50% error tolerance of input data. It is a robust and open source technique for cycle-consistency computation. For local case, in [3] diffusion framework shows good local-consistency, and we hope it can find good matching between puzzle pieces.

1.4 Contributions

To our first hypothesis our contributions are:

- We propose a multi-layer graph (MLG) representation to capture detailed geometric, topological and hierarchical information from the input and merged segments of shapes.
- We propose a matching technique to obtain geometrically, topologically and hierarchically consistent matching results with over/imperfectly-segmented inputs. From our experiments, it outperforms [36] quantitatively and qualitatively in our user study.
- To the best of our knowledge, this is the first technique which can obtain meaningful merged-to-merged segment-wise correspondences. This has not been considered before in the literature.

To our second hypothesis our contributions are:

- We innovate to use corner-wise correspondences for the puzzle solving task — we demonstrate its usefulness for square puzzle solving, and illustrate one example of how it can be adopted for rectangular pieces of arbitrary sizes.
- We propose a loop discovery technique for puzzle solving by modelling it as a cycle consistent correspondence problem, which allows to use the MatchLift framework [16] for puzzle solving.

To our third hypothesis our contributions are:

- We innovate to use sub-edge-wise correspondences for the puzzle solving task — we demonstrate its usefulness for puzzle solving of square pieces, rectangular pieces with different size, and pieces of squares and triangles.
- We propose a two-way cycle-consistency discovery technique for puzzle solving by modelling it as a local-consistent correspondence problem, which allows us to use the diffusion analysis [3] to infer the global-consistent correspondences for puzzle solving.

1.5 Thesis Structure

In Chapter 1, we have introduced the background of consistent matching problem in both 2D and 3D domains, and we have discussed the three problems that we have observed from existing works and we show our hypothesis to handle each problem. Our contributions are shown in section 1.4.

In Chapter 2, we discuss related works in two categories, shape matching and puzzle solving. We introduce shape matching related works in section 2.1. We first introduce the popular constraints for solving shape matching problem, and then we show variant works in 3D shape matching. The puzzle solving related works are introduced in section 2.2. We first introduce similarity measurement techniques of puzzle pieces, and then we show how to place each puzzle piece by using piece-wise similarity information.

In Chapter 3, we explain our MLGs approach to answer the first problem that we have found. We introduce our merging technique and our pruning technique. Then, we evaluate our proposed method in both rigid and non-rigid data sets, and we show that the MLGs approach is better than state-of-art works by comparing our method with them. We also summarise our limitations and show our short term future work.

In Chapter 4, we show how to use cycle-consistency in puzzle solving and answer our second and third problem (we mainly focus on the third problem). We first introduce MatchLift in details, and then we show how to model puzzle pieces into MatchLift framework. We evaluate our proposed method in different experiments, and we compare our proposed method with state-of-art work. Still, we summarise our limitations and show short term future work.

Chapter 5 shows how to use the diffusion framework in puzzle solving and our technique can handle variously shaped puzzle pieces, this chapter also answers the second and third problem. First, we show our EMD based puzzle pieces similarity measurement. Then, we model puzzle pieces into diffusion framework and we prune inconsistent results by using our cycle-consistent pruning procedure. Our experiments show that the combination of cycle-consistency and diffusion analysis can produce good results. In the end, we show our limitation and short term future work.

Chapter 6 shows our long term future work and the conclusion of this thesis. Section 6.1 will discuss our future works in long term only, since we have addressed the limitations and the short term future works in the section 3.7, 4.7, and 5.4. Finally, section 6.2 will summarise the thesis.

Chapter 2

Related Works

Contents

Since 3D shape matching and 2D puzzle solving can be modelled as correspondences problem, our work covers both domains. Thus, we use section 2.1 and section 2.2 to introduce the background of 3D shape matching and puzzle solving, respectively.

2.1 Registration and Shape Matching

Our method involves several categories of global geometry features and distances 2.1.1, registration in 2.1.2, hierarchical analysis of shape topology in 2.1.3, segment-wise shape matching in 2.1.4, and cycle-consistency in 2.1.5. We first introduce notable works in each category, then we summarise and discuss existing works in 2.1.6.

2.1.1 Global Geometry Features and Distances

There are many shape features and distance measurements developed over the past decades. We mention some important features and distances, and those that are particularly relevant in this section. We would like to refer readers to recent surveys [71, 1].

Geometry features have been used as the constraint to evaluate the similarity between 3D shapes. Light Field Descriptor [72] is one of the notable geometry descriptors. It is based on a set of 2D images of the input shape (captured from different angles) and uses image-based features for measuring shape similarity. [73] introduces a 3D shape histogram approach with sampled points on meshes to determine shape similarity. [74] further extends 3D shape histograms into A3/D1/D2/D3/D4 descriptors with different random sampling-based measures. [75] uses eigenvalues from PCA to determine shape distribution features (such as linearity, sphericity, omni-variance, change of curvature). These distribution-based features may be unreliable in some instances (e.g. the left base and right cap have similar scores in Fig. 3.6). Heat Kernel based descriptors such as Heat Kernel Signature (HKS) [76] use heat diffusion on meshes to define point-based features. Persistent-HKS [77] extends HKS and can be used as a descriptor for partial matching of non-rigid shapes.

Distances have also been used as a constraint in shape matching to limit distance/topological variant. Geodesic distance has been widely used in solving the shape matching problem, which is defined by the shortest path between two points on a mesh. Early works have confirmed that geodesic distance has good performance in preserving original distance information under a large surface/shape/articulations deformation or isometric transformation [78, 79, 80]. To

obtain geodesic distance, different approaches have been developed, such as Bellman–Ford algorithm, Floyd-Warshall algorithm, Dijkstra's Algorithm, or others. [76, 77, 54, 59, 65] shows that the norm is a popular way to estimate the distance between two vectors. Different from a traditional point-to-point distance, Mahalanobis distance [81] computes the distance between one point and a distribution. It is not sensitive to scale changing and it can be used to compute unbalanced or multivariate data sets. For computing two distributions, Earth Mover's Distance or EMD [82, 83] is a choice. Based on the given distance between individual objects, EMD computes the distance between two multi-dimension distributions from objects.

2.1.2 Registration

Shape registration and point-based matching is an important research area with a long history [71]. The research challenges are to develop robust and accurate techniques to handle shapes undergoing different transforms (rigid) and deformations (non-rigid), including near-/non-isometric deformations [84]. The rigid registration techniques optimise correspondences alignment to globally place/align two shapes without deformation. For example, ICP (iterative closest point) is a popular algorithm in rigid transformation[4, 5]. It uses the closest point as the constraint to iteratively compute correspondence construction and correspondences alignment. Variant works are followed by this approach with improved performance in speed or accuracy [6, 7]. The non-rigid registration techniques allow deformation between two shapes to be aligned. It optimises local correspondences alignment on a pair of shapes. For example, [85] uses designated energy functions to compute the local placement of correspondences for shape alignment. Find subsets of sampled shape features can help form meaningful or semantic matching [1]. There are further many existing works, e.g. [86, 87, 88, 89] rely on sampled/key points on input shapes to compute correspondences, and then use designated objective functions to analyse alignment/distortion errors and generate matching. One of the notable techniques [90] uses deformation distortions to obtain semantic matching.

Compared to other techniques that require specific constraints (e.g. sphere topology [84]),

one of the notable matching techniques [10] uses spectral analysis and has inspired many subsequent and useful point-based matching and registration techniques, e.g. [2]. The spectral pruning technique [2] assumes near-isometric deformation using global geodesic isometry. However, when the deformation is large (becoming non-isometric deformation), the technique does not perform well. [3] proposes a diffusion pruning (DP) technique to infer global consistency from locally consistent matching. It has been shown to handle moderate non-isometric deformation well [12]. Shape registration and matching have been studied for more than 20 years, a complete literature survey of shape registration and matching techniques is beyond the scope of this thesis. We would like to refer readers to surveys (e.g. [71] and [1]).

2.1.3 Hierarchical Understanding

Some works solve the shape matching/synthesis problem using a hierarchical approach for higher-level understanding. [91, 92, 93] use graphs encoded with probabilistic and topological information to solve region-wise matching or shape synthesis problems. [56] converts input shapes into component relationship graphs and then combines graph subsets with designated symmetric functional arrangement for synthesising new shapes. [38] combines component relationship graphs and deformation energy constraints to establish meaningful segment-wise correspondences of input shapes. Binary decomposition approaches are also used to help with hierarchical understanding. [94] introduces a novel shape representation in a binary hierarchical manner which cuts a shape from-whole-to-segment hierarchically. [37] finds the best binary segmentation in a top-down manner, via matching along the object hierarchy and uses recognition measures to handle structural variations and inconsistent initial segmentation. It has better performance than [38]. The technique, however, may fail in fine-grained matching because such cases lack the support of cross-layer information (see more discussion in Section 3.7.) [95, 96] focus on merging shape parts to form a hierarchical graph representation of part-functionality with geometry and topological information. Inspired by all these works, we propose to build a multi-layer graph by merging adjacent nodes in a bottom-up manner. We do

not define specific constraints (e.g. functional constraint [56] or binary segmentation [37]). The search space we consider, compared to existing work, is arguably larger. To address this, we further develop a robust matching technique to discover meaningful segment correspondences even under inconsistent (over/imperfect) input segmentation.

2.1.4 Segment-wise Matching

A few works in the literature focus on segment-wise matching which we survey here. [35] relies on HKS features for pre-segmentation. It uses spectral matching to find segment-wise correspondences with a focus on symmetric/pairwise issues. However, it outputs pair-to-pair correspondences and may lead to no matching if there are left-right symmetry issues. [38] uses combinatorial tree search and a deformation energy constraint to establish meaningful segment-wise correspondences. One shortcoming of this method is that it may not work on fine-grained segmented shapes. [37] finds the best binary segmentation in a top-down manner, and matches along the object hierarchy. It does not exploit matching from object hierarchies and may result in some incorrect correspondences (see Figure 3.14a). SHED (Shape Editing Distance) [36] takes shape segments and performs matching to define a better shape similarity measure. It innovates to find both one-to-one and one-to-many segment-wise correspondences, using both geometry and topology information. It forces full matching which means each input segment must have at least one correspondence to another shape, which helps resolve some ambiguities with perfect input segmentation. However, when the input segmentation is inconsistent, incorrect matching may result.

2.1.5 Cycle-consistency

In recent years, various works have successfully applied cycle consistency to obtain globally consistent matchings in the image domain. [97] achieve good performance in structure and motion computation by using cycle consistency as the constraint to remove globally incorrect geometric relations. [98] uses globally cycle consistent image data to handle the problem of

duplicate structure instances in SFM (structure from motion). [70] uses cycle consistency to solve co-segmentation problem in images domain. [14] shows a robust joint image alignment technique which is based on the usage of cycle consistency. [99] uses cycle consistency to solve image-to-image translation problem in computer vision. [100] shows a semantic image-wise matching technique with cycle consistent features in the images.

Cycle consistency can also help in solving shape matching problem. [15] shows a semi-definite programming (SDP) approach for solving the cycle consistent matching problem in the 3D shape matching domain. It shows that SDP can provide up to 50% error tolerance of pair-wise matchings between input objects. Building on [15], [16] introduces MatchLift for solving globally consistent matching in a general setting, with a tolerance rate of $1-\Theta(\log^2 n/\sqrt{n})$ to random outliers.

2.1.6 Summary

Inspired by [16], we suggest that the puzzle solving can be cast in the MatchLift framework, which helps discover loop correspondences. To do so, we use *corners* of pieces as the basic unit while most existing techniques use the whole edges (e.g. MGC). It provides a flexible framework for solving both square and rectangular puzzles of arbitrary sizes. The high tolerance to input errors of our method (due to the MatchLift framework) helps improve the precision of MGC matchings, making our method more robust for challenging inputs. We introduce the details in Chapter 4.

To our knowledge, none of the existing techniques considers inconsistent (over-/imperfect) input segmentation. Our technique is the first work to handle this challenge. Our novel idea is to use a multi-layer graph to represent possible merging arrangement and carry out our matching on such graphs. Together with a novel voting step (details will be explained in the section 3.5), our results are geometrically, topologically and hierarchically consistent (measured by different user studies).

2.2 Puzzle Solving

We discuss existing puzzle solving techniques in three sections. There are two steps in solving jigsaw puzzles: first, compute similarity scores between any pair of puzzle pieces; second, assemble puzzle pieces based on similarity scores. Section 2.2.1 discusses similarity measures for piece matching. Section 2.2.2 summarises assembly techniques for puzzle solving. Finally, Section 2.2.3 summarise the existing techniques.

2.2.1 Similarity Measurement of Puzzle Pieces

Pairwise similarity measures of puzzle pieces have been widely used for puzzle solving. [20] introduced MGC, which is a dissimilarity metric. It computes Mahalanobis distance of colour matrices gradients to determine the boundary similarity between puzzle pieces. Another notable work is SSD (Sum of Squared Distance) approach from [22]. It also uses the colour information of pixels at boundaries of each piece, and the similarity scores can be found by computing squared distance between different colour values. SSD approach is available for various colour spaces, such as RGB(Red Green Blue), HSV(Hue Saturation Value), or LAB(Lightness Red/Green Blue/Yellow). [53, 57] uses SSD as the similarity measurement.

As described above, MGC and SSD have been widely used in solving puzzles. Still, some works involved both MGC and SSD. [101] combines (by addition) MGC and SSD together as piece-wise similarity measurement. It shows that the combined measurement has better performance than MGC alone. Other works such as [58, 102] use original MGC/SSD as their similarity measurement.

Various L norms are also popular in puzzle solving. [54, 59, 65] use L_p^q, L_1, and L_2 norm to compute similarity scores between puzzle pieces, respectively. They convert the pixels of edges into colour matrices. Each colour matrix contains colour channels, and each colour channel is a vector and the elements inside of vector are colour values. Therefore, by using L norms the distance between colour channels/vectors can be determined, which can be considered as the similarity score between puzzle pieces. Using L norms or SSD have a similar computation

pipeline. We use Figure 2.1 and section 2.2.3 to discuss the details.

2.2.2 Assemble Puzzle Pieces

Puzzle solving is a challenging problem due to the large search space. Many works use greedy approaches for puzzle assembly from pieces [54, 20, 59, 102, 103]. In general, a greedy approach uses designated constraints (in terms of placement of pieces and similarity measure) to find the correct assembled results. They often begin from a small, confident region and gradually expand it by accepting new pieces. Since adjacent pieces are locally consistent, these techniques often do not refine or rectify incorrect assembly results. The final assembled results may not be globally consistent.

Greedy approaches with loop constraints show good performance. [58] introduces a novel four-piece-loop constraint for finding small cycles. Each cycle can be considered as an assembled region that contains four puzzle pieces. They first compute all pairwise MGC scores as the similarity measure between puzzle pieces. Based on the MGC scores they find small cycles. Next, they merge small cycles to build larger cycles, which form larger assembled regions. [64] builds on the idea and hierarchically merges small cycles. When incorrect pieces are matched, loop constraints provide a mechanism to examine piece neighbours and remove inconsistent ones. It improves puzzle assembling results. [61] models puzzle assembly as a linear programming (LP) problem. They iteratively optimise pieces and increase the size of the assembled results. Each iteration of LP optimisation can be considered as a general loop constraint optimisation. It shows that LP can perform better than [58].

Global approaches [47, 104, 22, 101] assemble puzzles by optimising a global objective function. [55] shows to use quadratic programming (QP) to optimise piece placement globally. [63] shows QP performs good assembled results even when there are missing puzzle pieces from the input.

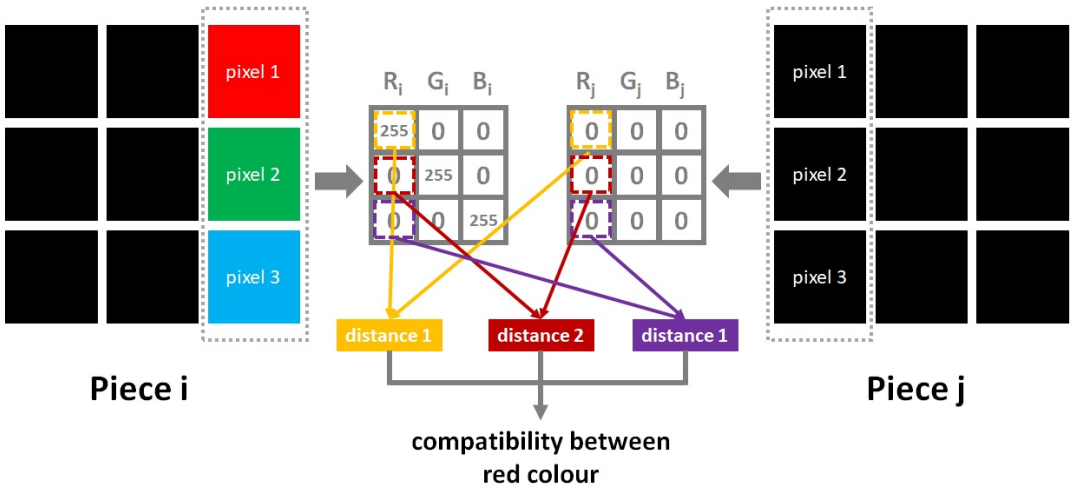
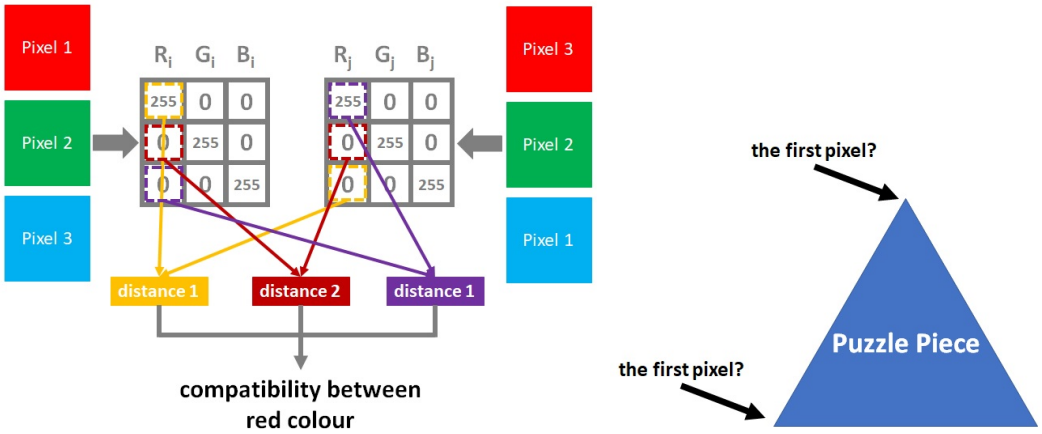


Figure 2.1: Typical similarity measure between a pair of puzzle pieces. Pieces with different edge-length are challenging for this approach.



(a) Flipped pixels ordering generates wrong result

(b) ordering of pixels on an slope edge

Figure 2.2: When ordering of pixels is changed the similarity scores will no longer be stable anymore.

2.2.3 Summary

Though loop constraints have been used in the literature, we observe that these techniques are mostly tailored for solving square puzzles only. They are not flexible to extend and handle rectangular pieces of arbitrary sizes. It inspires us to tackle this challenge, and the use of corners and cycle consistency for puzzle solving. We show our proposed method in Chapter 4.

In the similarity measurement, the ordering of pieces might be a problem. Figure 2.1 shows a left-right similarity measure example of two square pieces with nine pixels on each. For each piece, existing works select all pixels of an edge and convert them into a colour matrix. In this example, we use RGB colour space and there are three colour channels in each colour matrix. Then they compute distance (such as L_1 norm) between each pair of pixels that are from the same colour channel with the same pixel index. Therefore, the top value of colour channel R_i will not be computed with the middle/bottom value in colour channel R_j (if we flip the pixel indexing of $Piece\,j$ then the top value of R_i will only be computed with the bottom value in the R_j). The same procedure will be applied to other RGB values in other colour channels to generate channel-wise distances. The final left-right similarity score between the two pieces will be computed by combining all channel-wise distances. One thing that should be noticed is that the ordering of pixels. As shown in Figure 2.2a when the ordering of pixels are flipped, the similarity measure will return an unstable score to indicate two identical edges are highly dissimilar.

This ordering problem is not challenging square puzzle pieces, but it is tricky for pieces with slope edges (such as the triangular pieces). As shown in Figure 2.2b, the slope edges need a fixed ordering to label which pixel is the first one and where is the last pixel of an edge. In our work, we convert colour values from the same colour channel into histograms, and then we compute histograms distribution to measure the similarity score. This histogram approach is not sensitive to pixels ordering. We show detailed information in Chapter 5.

Chapter 3

Consistent Segment-wise Matching with Multi-layer Graphs

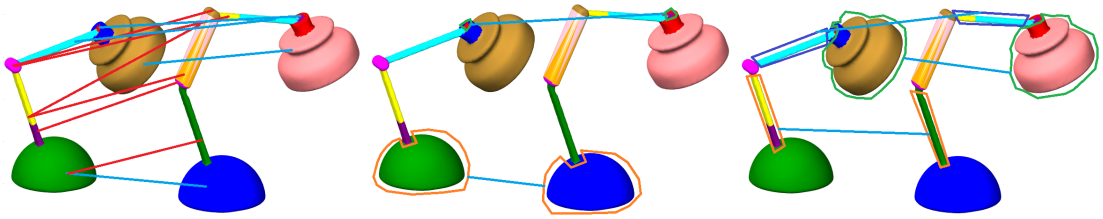
Contents

3.1 Introduction

Given two similar 3D meshes (for instance, two triangle meshes) with pre-defined segments, 3D segment-wise matching aims to establish meaningful correspondences of segments between the two meshes. It is an important problem as it helps with higher-level and hierarchical understanding in geometry analysis [37]. It further impacts many downstream applications, like defining better similarity measures between 3D models [36, 93, 35], functionality analysis [105], surface registration [2] and structure-aware analysis [106].

A few notable techniques propose in recent literature. Many of them combine topological and geometrical information to help solve the segment-wise matching problem. [36, 35] both take input shape segments and build a component graph to capture the topological relationship of segments. Together with the geometric similarity of segments, they adapt the spectral technique [10] for matching. SHED (Shape Editing Distance) [36] innovates to consider one-to-many matching while [35] focuses on the robust matching of non-isometrically deformed segments and disambiguating symmetric segments. [38] also takes pre-defined shape segments as input and builds a component graph to represent their topology. To solve the segment-wise matching problem, they use deformation energy as an effective constraint to produce higher-level semantic matching results. [37] builds a hierarchical component graph using a binary partition technique. Their matching technique adopts a top-down approach and achieves good results.

We observe two problems for the methods in the existing literature. First, most of these techniques rely on input with consistent segmentation [36, 35, 38, 37]. When the input segmentation is inconsistent (over-/imperfectly segmented), they often lead to incorrect correspondences. For example in Figure 3.1, the two lamps are inconsistently segmented (one has more segments than the other on the joint). [36] (Figure 3.1a) investigates one-to-many correspondences and further requires full matching, i.e. every segment from one shape is matched



(a) SHED Matching Result (b) Our Bottom Layer Matching Result (c) Our Higher Layer Matching Result

Figure 3.1: Example matching of inconsistently (over/imperfectly) segmented shapes. In all figures in this chapter, colour of segment indicates segment boundary only (not correct correspondences). Instead, we use blue lines for correct correspondences and red lines for incorrect ones (according to our user study). We further use polygons with the same colour to indicate one-to-merged or merged-to-merged correspondences in our results. In this example, it is difficult to define a correct correspondence for the middle (purple) joint of the left lamp. In our results we do not force full matching but leave it as unmatched to reduce incorrect matching. Full matching techniques such as SHED produce incorrect matching between inconsistently segmented regions.

to at least one segment in another shape. Affected by the different joint composition on the right lamp, the topology (graph distance) of the underlying component graphs differs a lot. As a result, [36] returns incorrect matchings (indicated by red lines). Second, correct segment-wise matching also depends on global shapes and functionality. For example, in Figure 3.1b the upper stick of the right lamp and the lower stick of the left lamp are over-segmented into two segments. Ideally, the left lamp's upper stick should be matched to all segments of the upper stick on the right lamp. It requires merging of segments before a meaningful, consistent segment-wise matching can be established (Figure 3.1c). These observations inspire us to investigate the research questions in section 1.3.2

To address these questions, we propose to construct multi-layer graphs (MLGs) to represent the input shapes with inconsistent segments. Inspired by [95], an MLG is a graph consisting of nodes with input and merged segments which is built in a bottom-up manner by neighbour merging. Different from [95], our merging technique uses many possible combinations based on the connectivity (if two segments share common faces/vertices) of input segments. In this way we achieve better capability with over-/imperfect input segmentation than [95].

Next we find consistent matching between MLGs by adapting the diffusion pruning (DP) technique [3] and using both geometric and topological constraints. Inspired by spectral techniques, DP computes matching results by inferring global consistency from the local matching. It has been shown to be robust against moderate non-isometric deformation [3]. It would allow us to handle moderate changes in graph distance due to over/imperfect input segmentation.

Further, different from existing techniques [36, 35] that apply spectral matching on component graphs built from input segments only, we apply DP on the proposed multi-layer graphs (MLGs) consisting of both input and merged segments. Compared to [36] which innovates in one-to-many matching, our technique can offer both one-to-merged and merged-to-merged correspondences. From our experiments, our technique produces better results than [36]. The obtained matching results are also consistent across layers while existing top-down approach [37] may fail (see Section 3.7).

To be consistent throughout this chapter, we use the term "components" for semantic parts obtained from perfect segmentation that respect human intuition. "Segments" instead refer to regions resulted from perfect or imperfect segmentation. Section 3.2 provides an overview of our technique. Then we discuss the construction of MLGs from input shapes and initial matching computation in Section 3.3. Section 3.4 explains diffusion pruning and how to adapt it on MLGs. After that we vote the pruned results in Section 3.5. We evaluate our method in Section 4.5. Finally, discussions and conclusions are presented in Sections 3.7 and 3.8.

To our knowledge, none of the existing techniques consider inconsistent (over-/imperfect) input segmentation. Our technique is the first work to handle this challenge. Our novel idea is to use a multi-layer graph to represent possible merging arrangement, and carry out our matching on such graphs. Together with a novel voting step, our results are shown to be geometrically, topologically and hierarchically consistent.

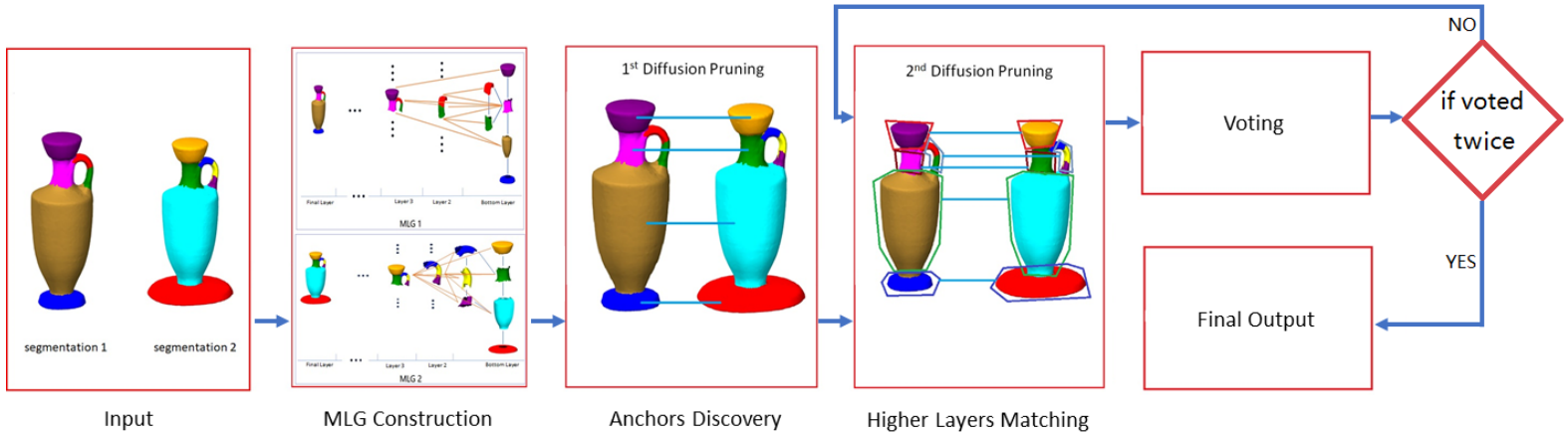


Figure 3.2: Method overview: our technique first builds multi-layer graphs to represent the input meshes from the pre-defined segmentation. Such pre-defined segmentation may be inconsistent between two shapes. Next we adapt diffusion pruning (DP) [3] on the bottom layer to find anchors. With the support of anchor correspondences, we apply DP again on the multi-layer graphs to obtain initial matching. A voting technique is further applied to confirm high quality segment-wise correspondences using matching from high layers.

3.2 Method Overview

Figure 4.3 shows an overview of our proposed method for segment-wise matching with inconsistent input segmentation. It involves four steps, namely **multi-layer graph construction** (Section 3.3), **discovery of anchor correspondences** (Section 3.4), **higher layer matching** (Section 3.4.2) and **voting** (Section 3.5).

Given two shapes with inconsistent segments, we build two hierarchical segment graphs (referred to as multi-layer graphs, MLGs) to represent the original shapes. Each input segment in a shape is assigned a graph node. All input segment nodes are grouped into one layer, denoted as the **bottom layer**. A merging stage is then applied to the nodes in the bottom layer to construct the MLG. It generates new nodes and new layers and is applied recursively until all nodes are merged into one — the original shape. After we have built two MLGs, we compute geometry similarities between nodes in the two MLGs for initial matching. Next, we adapt the diffusion pruning technique to compute good matching. There are two stages: the first pruning stage involves only the bottom layer in both MLGs. This is inspired by [36] as SHED provides reasonable results with perfect segmentation. Only strong results are used as anchors for the second pruning stage. For inconsistent input with large topological/geometrical variation however, using only nodes in the bottom layer alone often does not provide acceptable results. The second pruning further uses these anchors and involves more layers than previous pruning computation. Finally, we apply our voting technique to extract and confirm highly confident segment matching, using correspondences in higher layers.

3.3 Multi-Layer Graph and Initial Matching

Given a shape with predefined segments, we define the multi-layer graph (MLG) as a hierarchical representation. It covers possible merging arrangements of segments that are adjacent in a shape. An MLG consists of nodes and edges. Nodes are further grouped into layers. Bottom layer (layer 1) consists of input segment nodes whilst higher layers consist of nodes due to

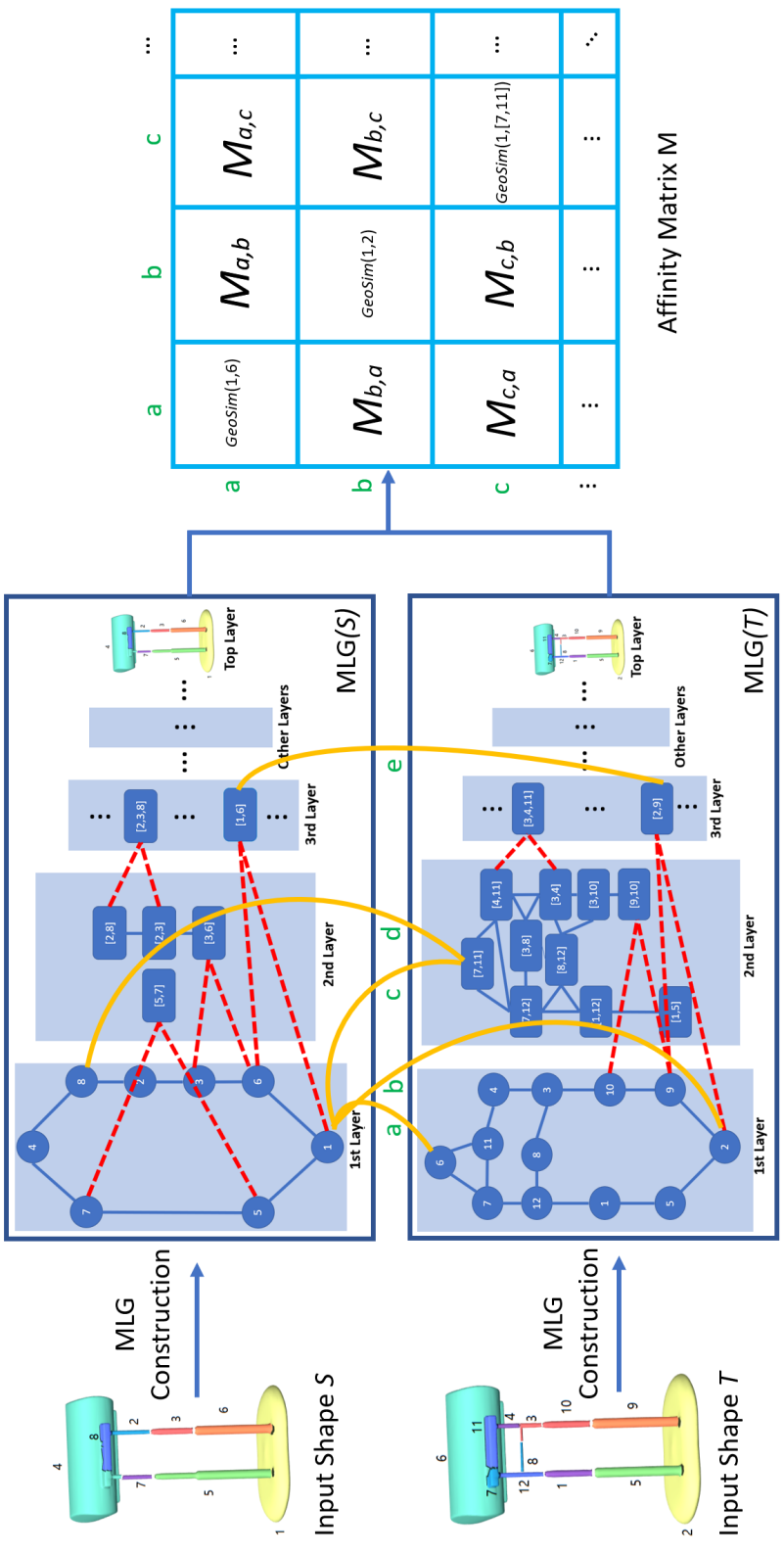


Figure 3.3: Example of MLG construction. We use blue edges to indicate adjacent nodes in the same layer. The red dotted lines indicate cross-layer edges. Note that merged nodes may not reside in the next layer, but a layer that satisfies the volume constraint. We use curved yellow lines to indicate the initial matching. The affinity matrix M is computed by Equation 3.3.

merging of two adjacent nodes in a lower layer. Nodes in internal layers are further connected by edges indicating their adjacent connections (within layer) and where the nodes are merged from (across this and lower layer). The highest layer consists of only one node. It represents the entire shape where all segments are merged. We first define the construction of multi-layer graph equipped with a specific volume constraint, and then discuss the initial correspondences.

3.3.1 Multi Layer Graph

Node Construction with Volume Constraint Precisely, let $\mathcal{S} = (V, E)$ be a 3D shape with sets of vertices V, edges E and pre-defined input segments $\{S_1, S_2, S_3, ...\}$ where $\mathcal{S} = \bigcup S_i$ is the union of vertices $\subset V$ and edges $\subset E$ in S_i. Denote by $\bar{N}_k^{[l]}$ the k^{th} node in the l^{th} layer of a source shape MLG($\mathcal{S}$). We construct the nodes of MLG($\mathcal{S}$) recursively in a bottom-up manner:

$$\bar{N}_k^{[l]} = \begin{cases} S_k & \text{if } l = 1 \\ \bar{N}_i^{[m]} \cup \bar{N}_j^{[m]} & \text{if } \bar{N}_i^{[m]} \cap \bar{N}_j^{[m]} \neq \emptyset,\ i \neq j,\ \mathcal{C}_{vol}^{l-1} \leq VOL(\bar{N}_k^{[l]}) < \mathcal{C}_{vol}^{l},\ m < l \end{cases} \tag{3.1}$$

In this way every input segment S_i is assigned a node $\bar{N}_i^{[1]} = S_i$ and are grouped to form the bottom layer ($l = 1$). Higher-layer nodes are created by merging all vertices and edges in lower-layer nodes (in the same layer) only if they are adjacent. Two nodes are adjacent if they share some vertices $\subset V$, edges $\subset E$ in $\mathcal{S}$ such that $\bar{N}_i^{[l]} \cap \bar{N}_j^{[l]} \neq \emptyset$. Simply merging adjacent nodes would lead to exponential growth in number of merged nodes. We thus define a volume constraint $\mathcal{C}_{vol}^{l-1} < VOL(\bar{N}_k^{[l]}) < \mathcal{C}_{vol}^{l}$ to restrict the volume of a node in each layer. We define the upper bound $C_{vol}^{l} = \frac{l}{L} VOL(S)$ for each layer l, where L is the maximum number (a user defined parameter) of layers in MLG($\mathcal{S}$) and VOL($\mathcal{S}$) is the total volume of shape $\mathcal{S}$.

Edge Construction Next, we define the edges of MLG($\mathcal{S}$). For every pair of nodes $\bar{N}_i^{[l]}, \bar{N}_j^{[l]}$ in the same layer l with shared vertices/edges (i.e $\bar{N}_i^{[l]} \cap \bar{N}_j^{[l]} \neq \emptyset$), a within-layer or "adjacency" edge $(\bar{N}_i^{[l]}, \bar{N}_j^{[l]})$ is established between them. Let $\bar{N}_k^{[l]} = \bar{N}_i^{[m]} \cup \bar{N}_j^{[m]}$ be an internal node which is merged from two nodes $\bar{N}_i^{[m]}$ and $\bar{N}_j^{[m]}$, where $m < l$. We establish two cross-layer or "part-of"

edges $(\bar{N}_i^{[m]}, \bar{N}_k^{[l]})$ and $(\bar{N}_j^{[m]}, \bar{N}_k^{[l]})$ between them. That is, the edge $e \in E_{MLG(\mathcal{S})}$, the edge set of MLG($\mathcal{S}$), is defined as:

$$e = \begin{cases} (\bar{N}_i^{[m]}, \bar{N}_j^{[l]}) & \text{if } m = l,\ i \neq j,\ \bar{N}_i^{[m]} \cap \bar{N}_j^{[l]} \neq \emptyset \\ (\bar{N}_f^{[m]}, \bar{N}_k^{[l]}) & \text{if } m < l,\ i \neq j,\ f \in \{i, j\},\ \text{s.t. } \bar{N}_k^{[l]} = \bar{N}_i^{[m]} \cup \bar{N}_j^{[m]} \end{cases} \tag{3.2}$$

We have tried different weights for within-layer and cross-layer edges, and found empirically that setting all edge weights to 1 can produce good results. We therefore use this for all subsequent experiments due to simplicity. An example of the construction of nodes and edges in MLG is shown in Figure 3.3.

3.3.2 Initial Matching

Next, we compute the geometric similarity score and generate initial correspondences. Our proposed technique mainly uses LFD as it is more robust for small segments. In general, local features can be used to obtain initial matching, but the results are likely to be globally inconsistent. Our technique aims to produce consistent segment-wise matching results. We have also tried several techniques and found that LFD [72] similarity scores perform well (even for our non-rigid experiments as individual segments are relatively small and close to rigid). We will use LFD similarity throughout this chapter. We pre-compute MLG($\mathcal{S}$) and MLG($\mathcal{T}$) for two input shapes $\mathcal{S}$ and $\mathcal{T}$. For each node $\bar{N}_i^{[u]}$ in MLG($\mathcal{S}$) we pre-compute the K best matching (in terms of LFD similarity scores) of node $\tilde{N}_j^{[v]}$ in MLG($\mathcal{T}$), as its initial matching (shown as the yellow lines in Figure 3.3).

3.4 Diffusion Pruning with Anchors

Once the initial matching has been pre-computed, we adapt and apply diffusion pruning to obtain consistent matching results. We equip our technique with two pruning stages (Figure 4.3). The first stage considers input matching between nodes in the bottom layers of the two MLGs only (i.e. correspondences between nodes $\bar{N}_i^{[1]}$ and $\tilde{N}_j^{[1]}$). We treat these first-stage matching

results, with high confident scores as anchors. In the second stage, we consider higher layers matching (i.e. correspondences between nodes $\bar{N}_i^{[u]}$ and $\tilde{N}_j^{[v]}$) in the MLG hierarchy. There are often a large number of nodes in the MLG. The first-stage anchors offer good constraints to the second-stage matching results.

One of the matching problems with inconsistent input segmentation is that the underlying connectivity graph often shows non-isometric inconsistency in term of topological distances. The diffusion pruning technique [3] has been shown useful to obtain good point-wise correspondences under moderate non-isometric shape deformation. We thus adapt it to our use for segment-graph hierarchical matching. Given some initial correspondences, we construct an affinity matrix to encode both geometry similarity and topological consistency of initial matching. We then adapt the diffusion framework to generate confidence scores. Based on the scores, inconsistent correspondences are pruned in a greedy manner. We would refer readers to [3] for the mathematical and implementation details. Here, we focus on the adaptation for our segment-wise matching task.

3.4.1 Affinity Matrix Computation

Given some segment-wise correspondences C, we build an affinity matrix M of size $|C| \times |C|$. M encodes both topological (MLG distance) and geometry (LFD) information. As shown in Figure 3.3 each element in $M(a,b)$ indicates the compatibility of two segment-wise correspondences $a = (\bar{N}_i^{[u]}, \tilde{N}_j^{[v]})$ and $b = (\bar{N}_x^{[n]}, \tilde{N}_y^{[m]})$ $(a,b \in C)$.

Using local isometry to infer global consistency is a key concept in diffusion pruning [3]. For a pair of nodes $\bar{N}_i^{[u]}$ and $\bar{N}_x^{[n]}$ in the same MLG, we define the MLG distance $d(\bar{N}_i^{[u]}, \bar{N}_x^{[n]})$ as the number of edges in the shortest path (in the MLG) between them. We use Dijkstra algorithm to compute the shortest path and the time complexity will be $O(log(N) \times e)$, where N is the number of nodes in the MLG and E is the number of edges in the MLG. The distance models the topological (both adjacent and part-of) relationship between segments within the MLG hierarchy. A local topological MLG region can be further defined around a node $\bar{N}_i^{[u]} \in$

$MLG(\mathcal{S})$ (similarly for nodes $\tilde{N}_j^{[v]} \in MLG(\mathcal{T})$) in the MLG hierarchy as $R^{\delta}_{\bar{N}_i^{[u]}} = \{x | d(\bar{N}_i^{[u]}, x) \leq \delta D\}$ where $\delta \in [0,1]$ is a user defined threshold and D is the largest MLG distance in an MLG. Given this, we can compute the element of matrix M. Let $m_{a,b}$ be the distance compatibility for two segment-wise correspondences $a, b \in C$. We follow the normalisation procedure in [2, 3] to obtain $M_{a,b}$ as follows:

$$M_{a,b} = \begin{cases} \frac{m_{a,b} - c_0}{1 - c_0}, & a \neq b,\ m_{a,b} \geq c_0,\ 0 \leq c_0 \leq 1 \\ GeoSim(\bar{N}_i^{[u]}, \tilde{N}_j^{[v]}), & \text{otherwise,} \end{cases}$$

$$m_{a,b} = \min\left(\frac{d(\bar{N}_i^{[u]}, \bar{N}_x^{[n]})}{d(\tilde{N}_j^{[v]}, \tilde{N}_y^{[m]})}, \frac{d(\tilde{N}_j^{[v]}, \tilde{N}_y^{[m]})}{d(\bar{N}_i^{[u]}, \bar{N}_x^{[n]})} \right),$$

$$\bar{N}_x^{[n]} \in R^{\delta}_{\bar{N}_i^{[u]}} \ \ and \ \ \tilde{N}_y^{[m]} \in R^{\delta}_{\tilde{N}_j^{[v]}}$$

$$GeoSim(\bar{N}_i^{[u]}, \tilde{N}_j^{[v]}) = |\mathrm{LFD}(\bar{N}_i^{[u]}) - \mathrm{LFD}(\tilde{N}_j^{[v]})| \tag{3.3}$$

$M_{a,b}$ will take into account only segment-wise correspondences a and b with end-point nodes fall into respective local topological MLG regions $R^{\delta}_{\bar{N}_i^{[u]}}$ and $R^{\delta}_{\tilde{N}_j^{[v]}}$ [3]. It further ensures $c_0 \leq m_{a,b} \leq 1$, i.e., $m_{a,b}$ be at least c_0 isometrically consistent [2], and sparsifies M if it does not. Different from [3], we further encode geometric similarity in the diagonal entries $M_{a,a}$ where $GeoSim(\bar{N}_i^{[u]}, \tilde{N}_j^{[v]})$ is the dissimilarity score of their LFD features.

3.4.2 Diffusion Framework and Pruning

Matrix M encodes both local geometric similarity and local topological isometric consistency information. The matrix is then normalised to a Markov probability matrix P to model the Markov random walk for diffusion analysis.

$$P(a,b) = \frac{K(a,b)}{d(a)},$$

$$d(a) = \sum_b K(a,b),$$

$$\sum_b P(a,b) = 1$$

$$\pi(a) = \frac{d(a)}{\sum_{b \in C} d(b)} \tag{3.4}$$

The normalised $P(a,b)$ can be considered as a confident/probability score of the jumping from correspondence a to b. After this, the stationary distribution π is computed as the confidence score $\pi(a)$ for a correspondence a. The function of the confidence score $\pi(a)$ is similar to the usage of eigenvector in [10, 2, 36], which shows the reliability/consistency of a correspondence. However, in diffusion computing we do not compute eigendecomposition to obtain $\pi(a)$. The normalisation step is essential to infer the global consistency from local topological isometric compatibility in MLGs. This framework is supported by the spectral graph theory [3]. We sort all initial matchings with descending confidence scores and examine each of them in a greedy manner [10, 2, 3].

In our algorithm, we apply diffusion pruning twice. In the first run, we only use bottom layers to obtain good correspondences (anchors). In the second run, we involve more layers in the two MLG($\mathcal{S}$) and MLG($\mathcal{T}$). During the second pruning stage, we first accept anchors into result correspondences, and then greedily add new consistent correspondences from higher layers. The idea is supported by two observations. First, SHED [36] produces reasonable results if the input contains perfect segments or there are some segments with high distinctive geometric scores. Our technique is similar to SHED that uses spectral analysis and shows similar behaviour, which means anchors are necessary for producing good results (explained in the Figure 3.4). Second, given imperfect input segmentation, our technique can better handle moderate non-isometric differences because of diffusion pruning. It can often find good and consistent matching based on local regions using just bottom layer. Given these good anchors, we can further constrain consistent outputs in the higher layers.

3.5 Voting and Final Output

Most of the results obtained in the previous step are useful. Still, some incorrect matching may still be present due to the greedy pruning procedure. There are two further reasons. First,

our simple topological distance incorporates both adjacency and part-of relationships as one measure and does not differentiate the two relationships. Second, nodes in higher layers often have similar shorter MLG distances, which easily lead to ambiguous matching. In our final step, we would like to further confirm that the pruned segment-wise correspondences are consistent throughout the MLG hierarchy. For example, a consistent segment-wise correspondence should appear as "part of" some merged-to-merged segment-wise correspondences in a higher layer. To confirm lower-layer correspondences using higher-layer ones, we develop a voting-prune procedure which is discussed below.

Let C_{dp} be a set of segment-wise correspondences (e.g. Figure 3.4) obtained from our adapted diffusion pruning step (Section 3.4.2). We first go through each correspondence $a = (\bar{N}_i, \tilde{N}_j) \in C_{dp}$ and check against another correspondence $b = (\bar{N}_x, \tilde{N}_y) \in C_{dp}$ where $a \neq b$. If both $\bar{N}_i \subset \bar{N}_x$ and $\tilde{N}_j \subset \tilde{N}_y$, we increment a vote $\text{Vote}(a)$ for a. A correspondence a from lower layers which are consistent with higher layer correspondences will accumulate more votes. Next, we sort all $a \in C_{dp}$ in descending order of $\text{Vote}(a)$ and use higher confidence score $\pi(a)$ from DP to break the tie if possible. Figure 3.4 shows example values of $\text{Vote}(a)$ and $\pi(a)$ of each correspondence at the top left and right corners of each subfigure respectively.

Our greedy hierarchical pruning step is then carried out using the sorted list. We first accept the first $a \in C_{dp}$ with the highest $\text{Vote}(a)$ into the C_{vote}, and remove a from C_{dp}. For each subsequent $b = (\bar{N}_x, \tilde{N}_y) \in C_{dp}$, we check $\forall a = (\bar{N}_i, \tilde{N}_j) \in C_{vote}$ if b satisfies either:

$$\bar{N}_i \subset \bar{N}_x \text{ and } \tilde{N}_j \subset \tilde{N}_y \qquad \text{or} \qquad \bar{N}_i \not\subset \bar{N}_x \text{ and } \tilde{N}_j \not\subset \tilde{N}_y$$

This step requires that the new segment-wise correspondence b is consistent with all accepted $a \in C_{vote}$ or b is not seen before. We then move b from C_{dp} to C_{vote}. If b violates both constraints, it means that b is an inconsistent correspondence. We simply prune it from C_{dp}. Matchings highlighted in blue round boxes in Figure 3.4 are all accepted correspondences C_{vote}. Matchings highlighted in red are inconsistent correspondences that are pruned. In our implementation, we further use C_{vote} as anchors for DP (which sometimes improves the greedy

Algorithm 1: Voting Algorithm

Input: C_{dp}
Output: C_{vote}

procedure VOTING(C_{dp})
  for each $a = (\tilde{N}_i, \tilde{N}_j) \in C_{dp}$
    Vote(a) $\leftarrow 0$
    for each $b = (\tilde{N}_x, \tilde{N}_y) \in C_{dp} \setminus a$
      if $\tilde{N}_i \subset \tilde{N}_x$ and $\tilde{N}_i \subset \tilde{N}_y$ **then**
        Vote(a) $\leftarrow$ Vote(a) + 1
      end if
    end for
  end for
  $C_{vote} \leftarrow \emptyset$
  while $C_{dp} \neq \emptyset$ **do**
    $b = (\tilde{N}_x, \tilde{N}_y) \leftarrow \text{argmax}_{a \in C_{dp}}$ Vote(a)
    if $\forall a = (\tilde{N}_i, \tilde{N}_j) \in C_{vote}$
      $\tilde{N}_i \subset \tilde{N}_x$ and $\tilde{N}_j \subset \tilde{N}_y \quad \vee$
      $\tilde{N}_i \not\subset \tilde{N}_x$ and $\tilde{N}_i \not\subset \tilde{N}_y$ **then**
      $C_{vote} \leftarrow C_{vote} \cup b$
    end if
    $C_{dp} \leftarrow C_{dp} \setminus b$
  end while
  return C_{vote}
end procedure

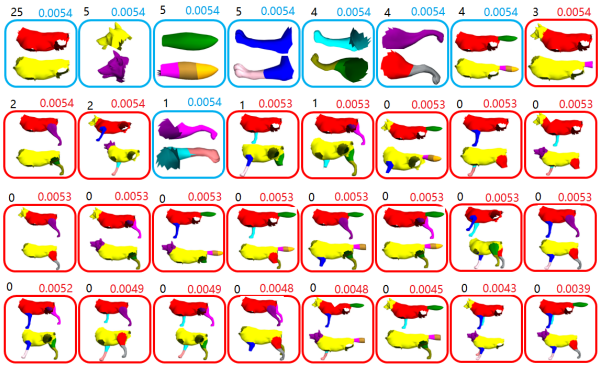
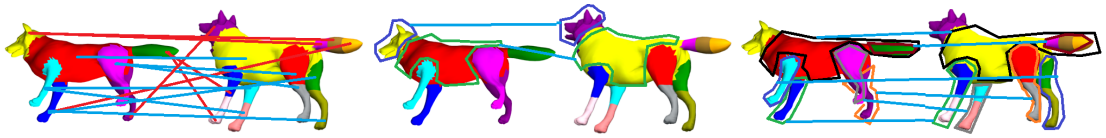


Figure 3.4: Voting of pruned results. The red ones are removed by voting. Top left numbers are votes and top right numbers are diffusion pruning confident scores.



(a) SHED Matching Result (b) Our Bottom Layer Matching Result (c) Our Higher Layers Matching Result

Figure 3.5: Refined matching by voting mechanism. All diffusion pruning results have been visualized in Figure 3.4. There are incorrect matchings, for example, at the top right corner *head-body* is matched to *body-tail*. After voting, these incorrect matchings are pruned (b-c). As a comparison, the SHED result is shown in (a).

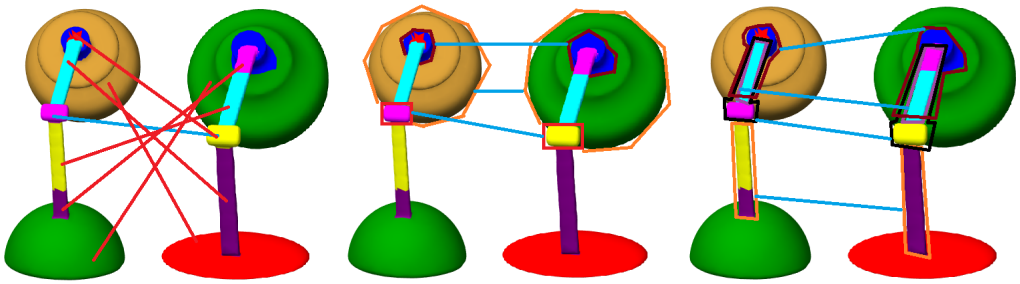
results), and run the voting-prune step again to obtain C'_{vote} as the final output (Figure 3.5(b)-(c)).

This voting step, together with diffusion pruning (Section 3.4.2), ensures that the accepted correspondences are topologically and hierarchically consistent within MLGs, and their end-point nodes are geometrically similar. The algorithm is detailed in Algorithm 1. Our time complexity is $O(|C|^2 + |C_{vote}|nlogn)$.

3.6 Evaluation

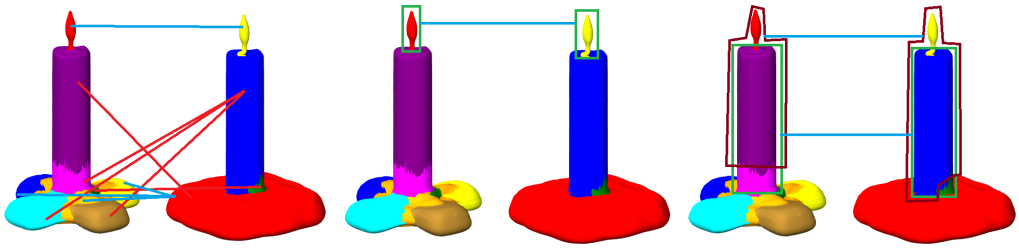
We evaluate our method on both rigid (man-made) and non-rigid shapes. The rigid data set is downloaded from the SHED's project page, which consists of four subsets, namely vases, airplanes, lamps and candles. All rigid shapes are segmented by a weakly-convex segmentation technique as mentioned in [36]. Our non-rigid set consists of wolf, human, horse, and centaur. We use the consistent segmentation results from [40] and further manually over-segment those shapes to provide initial inconsistent segmentations for our evaluation. With these inputs, we use SHED [36] and our proposed technique to compute segment-wise correspondences, and evaluate both techniques qualitatively (visual examples) and quantitatively (precision). The reason for using SHED in our experiments is that SHED finds one-to-many correspondences (which is similar to our merged-segment-wise correspondences). Still, [36] also released their source code which benefits our experiments.

To our knowledge, there is no existing ground-truth dataset for segment-wise matching. For high-level matching, there is a certain degree of human subjectivity involved. For example in Figure 3.1b, the purple joint on the left lamp has only one segment, but there would be many possible correct matching segments (e.g. all or one of the unmatched segments) on the right lamp. Even a no matching as shown in Figure 3.1b-3.1c can be a correct choice. To provide a fair evaluation, we recruited three volunteers (one sculptor, two musicians) from non-computer science background to carry out the annotations. We informed all volunteers that their annotations should be based on their own intuition of meaningful/reasonable correspondences with respect to the shape and segments. In this way each correspondence produced by [36] and our technique is given a correct or wrong label. We use a majority vote in cases where there is a discrepancy. These are used to compute the precision and to indicate correct or incorrect matchings in all figures. For all visualized figures, segment colours are only used to show distinct boundaries of segments, rather than matching correctness. Blue (red) lines indicate correct (incorrect) segment-wise correspondences. We further use colored polygonal lines to indicate our one-to-merged / merged-to-merged segment-wise correspondence results.



(a) SHED Matching Result (b) Our Bottom Layer Matching Result (c) Our Higher Layer Matching Result

Figure 3.6: Near-consistent segmentation matching result. Our method outputs meaningful matching at upper and lower sticks. There is a large variation between the two bases and our method does not match them.



(a) SHED Matching Result (b) Our Bottom Layer Matching Result (c) Our Higher Layer Matching Result

Figure 3.7: Comparison results of candles with inconsistent segmentation. As shown in green polygons our method can match body segments in a meaningful way.

3.6.1 Qualitative Evaluation

We have tried HKS [76] and persistent HKS [77] but they cannot produce distinct similarity scores for MLG nodes. Similarly, PCA [75] and D1/D2 [74] distributions occasionally produce incorrect scores. In this chapter we use LFD [72] to generate geometry similarity scores for segments, since it performs well in our experiments.

3.6.1.1 Rigid Shapes

In this section we evaluate our method on rigid shapes. We first test our method on shapes with near consistent input segmentation, and then with inconsistent input segmentation. Our evaluation mainly focuses on inconsistent segmentation which is the focus of this chapter.

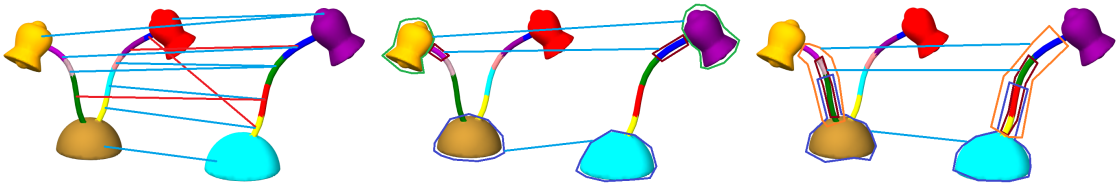
Near-consistent Input Segmentation

In Figure 3.6 two lamps have similar input segmentation except some small over-segmented pieces in the stand and cap joint. Figure 3.6a shows that SHED mismatches the base of the left lamp to the right lamp's cap. The mismatch is caused by a good geometry similarity score due to D1/D2/volume computation between the base (left lamp) and cap (right lamp). Further, both segments are located at the endpoints of their respective component graphs with similar topological distances to the rest of the nodes. As both geometric and topological information is very similar, SHED outputs an upside down matching. The volunteers consider the result as a mismatch. Our technique shows reasonable correspondences, with many one-to-merged segment-wise correspondences (Figures 3.6b-3.6c). For example in the left lamp, the small red piece above the cap joint, and the small purple piece in the lower stand are merged with respective larger piece in the matching results. Further, our technique is able to solve the upside down ambiguity because the one-to-merged segment-wise correspondences offer better geometric, topological and hierarchical consistency. The base is not matched because their geometry (LFD features) differs a lot.

In Figure 3.7, we show another example where the upper candlestick is near-consistently segmented, but the lower base is highly over-segmented. It is a challenging case because the base contains six inter-connected segments making the component graph very complex. Though SHED is able to obtain a one-to-many base-to-base matching, it also badly mismatches both candlesticks to the bases. Our technique is able to discover candlestick matching in a reasonable manner without any incorrect matching. It does not discover the one-to-merged base matching because it requires merging of all six segments to form the base which is beyond the number of layers we consider for the example (see Section 3.6.2).

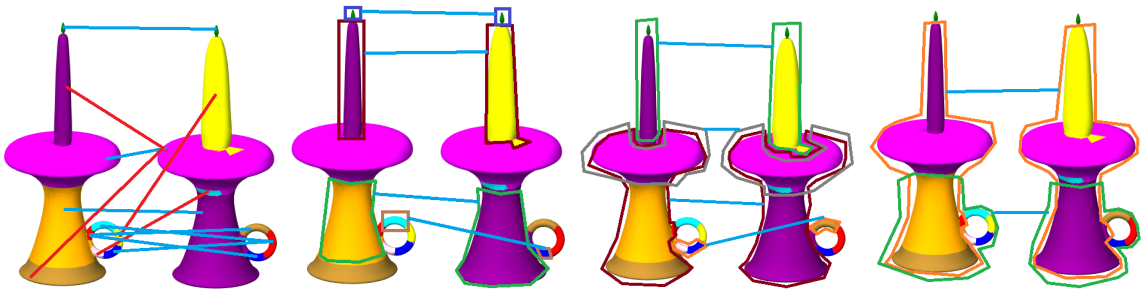
Inconsistent Input Segmentation (Matching with large difference in number of nodes)

Figure 3.8 shows an example with large topological variation. The number of segments in the left lamp is almost two times more than that of right lamp. SHED's one-to-many results are mostly good, but mismatches still appear. For example, the lower stick in the left lamp is



(a) SHED Matching Results (b) Our Bottom Layer Matching Results (c) Our Higher Layers Matching Results

Figure 3.8: Lamp matching results with large topological variation. Our method can find consistent matching with no mismatched correspondences.

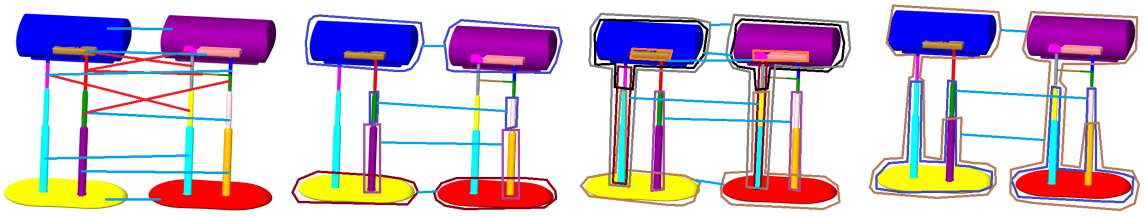


(a) SHED Matching Result (b) Our Bottom Layer Result (c) Our Higher Layer Result (d) Our Higher Layers Result

Figure 3.9: Matching results of candles with inconsistent segmentation. For challenging segments our method matches them in higher layers to avoid incorrect correspondences. We show all higher layers results in two sub-figures (c) and (d) for clarity purpose.

adjacent to the base, but it is mismatched to a node in the right lamp which is not adjacent to the base. Our volunteers consider the matching incorrect.

Our technique considers merged nodes in higher layers. It finds consistent matching on the left branch of the left lamp. Caps and bases are matched with two one-to-one correspondences, whilst the main stick is matched with a merged-to-merged correspondence. In this way we match all stick segments consistently, and avoid incorrect matching. Our technique does not offer one-to-many matching and thus no matching is obtained for the right stick (which is plausible). Our method may be extended to produce matching to the right lamp by first removing matched nodes and re-applying our technique (as demonstrated in [12] for discovering point-wise correspondences of multiple parts).



(a) SHED Matching Result (b) Our Bottom Layer Result (c) Our Middle Layers Result (d) Our Higher Layers Result

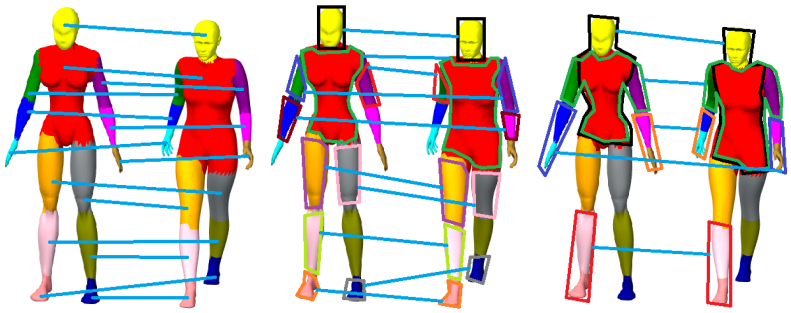
Figure 3.10: Matching result comparison for shapes with large topological variation and loops. We show all higher layers results in two sub-figures (c) and (d) for clarity purpose.

Inconsistent Input Segmentation (Matching with inconsistent input segments and loops)

Figure 3.9 shows another challenging candle example with inconsistent over-segments and loops. SHED matches many segments incorrectly. These incorrect matchings are largely influenced by the topologically-adjacent correct matchings. However, by using geometric and topological information alone, it is not sufficient to find good matching. Our technique discovers many reasonable matchings with merged nodes in higher layers which are consistent with human intuition. The loop handle is very challenging as it consists of many small pieces. Note that both SHED and our technique cannot resolve symmetry issue. Therefore, both SHED and our technique have some matchings that are controversial. For example, SHED returns many one-to-many matchings in the loop handle (Figure 3.9a). Our technique obtains a matching from the lower piece of the loop to the upper piece of the loop handle (in Figure 3.9b, and similarly upper piece to lower piece matching in the loop in Figure 3.9c). Our volunteers independently consider them (both SHED's and ours results) correct because they are part of the handle (due to functionality). Having said that, our technique discovers the loop pieces in a upside down, but consistent manner.

Inconsistent Input Segmentation (Matching with multiple loop structures)

Next, we focus on a more challenging example. Figure 3.10 shows the matching between two lamps with highly inconsistent input segmentation. In particular, the crossbeam and T-shaped segment (adjacent to the crossbeam) exist only in the right lamp. SHED tries to find one-to-many matchings for all segments. Though it can find some good matchings, it also returns



(a) SHED Matching Result (b) Our Bottom Layer Result (c) Our Higher Layers Result

Figure 3.12: Non-rigid matching comparison with consistent segmentation.

many incorrect ones (Figure 3.10a). Note that in the left stand (left lamp), the upper segment is inconsistently matched to the left and right stand (right lamp). The results can be explained by the segment graphs in Figure 3.11 as both segment graphs contain cycles. The crossbeam acts as a shortcut edge and creates another shorter cycle. This shorter path significantly distorts the topological distance on the segment graphs, leading to the inconsistent matchings in SHED.

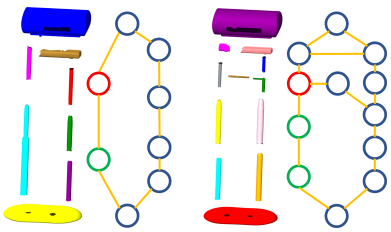
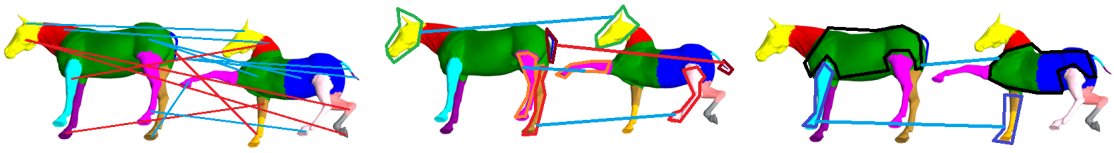


Figure 3.11: Segment graphs of two lamps.

Figure 3.10b shows that our technique obtains more reliable one-to-one matchings in the right stand. For the left stand, nodes are merged in the higher layers in the MLG graph (green circles in Figure 3.11). One-to-merged and merged-to-merged segment-wise matchings are resulted (see also the brown, blue and purple polygons in Figure 3.10c-3.10d). Since our technique looks for geometrically, topologically and hierarchically consistent matching, the crossbeam is not matched. The volunteers find our result reasonable.

3.6.1.2 Non-Rigid Shapes

In the literature, some segment-wise matching techniques do not support non-rigid shapes (e.g. [38, 37]). We further evaluate if our technique can support them. We have tried some geomet-



(a) SHED Results (b) Our Bottom Layer (c) Our Higher Layers

Figure 3.13: Non-rigid matching comparison with inconsistent segmentation.

ric features designed for non-rigid shapes (e.g. HKS [76], Persistent HKS [77]) but they do not provide distinctive geometric measures. Therefore, we use LFD in these experiments. Both SHED and our technique are built on top of component/segment graphs, and are not designed to handle symmetry issue — both cannot differentiate left or right. In non-rigid shapes symmetry is common. We thus consider matching say, left arm to right arm (or vice versa) as correct, as long as the whole arm (every segments in the arm) is consistently matched. Such symmetry issues could be addressed by incorporating a symmetry detection technique to resolve ambiguities.

Figure 3.12 demonstrates one human example with consistent input segmentation. In our technique, the non-rigidly deformed hands are not matched due to no initial correspondences (low LFD scores). LFD is defined mostly for rigid shapes only. In our result, hand and arm merged-to-merged matching can be obtained in higher layers because initial correspondences are available (merging hand and arm offer good LFD scores). We do not obtain matching for lower legs because of the volume constraint defined in the MLG (see section 3.3) where the leg (for the left human) is moved into higher layer for one of the shapes. It can be easily solved by relaxing the topological consistency thresholds c_0. We argue that our technique still performs reasonably well in this example despite of the LFD issue.

Figure 3.13 shows a horse example with inconsistent input segmentation. Our technique is able to obtain accurate matching under inconsistent input segmentation in legs and body. Note that under symmetry, front legs to back legs matching in both SHED and our techniques are considered correct. In Figure 3.13b our method outputs 1 incorrect result between tails. This

is caused by highly similar LFD scores. However, SHED often mismatches leg to tail or head. The volunteers consider them incorrect.

3.6.2 Quantitative Evaluation

We further evaluate our technique on large rigid and non-rigid data sets. Our method outputs matching of higher layers. There is no ground truth dataset, so volunteers have to manually examine each output matching to compute precision rate — it is a time consuming process. It is also not possible for us to enumerate all higher layer matching. For example, a shape with 14 segments can lead to 500+ internal nodes in the MLG depending on their topology. It is simply too laborious and time-consuming to annotate all of them. Therefore we do not evaluate on recall rate. Following [84] we randomly select pairs of shapes from each set and annotate the output. The whole annotation process takes several weeks to finish among all three unpaid volunteers. In our experiments, we use fixed parameters for all pairs in a set (similar to [36]).

For the rigid set, we use the following parameters for the adapted diffusion pruning to compute anchors: local distance $\delta_1 = 0.2$ and LFD threshold is 0.8. The second run of diffusion pruning uses $\delta_2 = 0.8$ and LFD threshold = 0.8. For both runs, the number of initial matching for each node K is set to 7; the threshold in diffusion pruning is set as default $c_0 = 0.7$ ([3]). For the non-rigid set, the values of $\delta_1 = 0.8$ and $\delta_2 = 0.2$ and other parameters stay the same as the rigid set.

The only parameter we adjust is the number of layers in MLG construction. We use eight layers in lamp and plane sets, and four layers for vase and candle sets. The reason is that there are too many internal nodes in the constructed MLGs with eight layers. Reducing the number of layers to four still provides reasonable results. All shapes in non-rigid sets have eight layers.

All quantitative results are shown in Tables 3.1 and 3.2, and are based on 72 pairs of rigid shapes and 20 pairs of non-rigid shapes. Our method outperforms SHED in all cases. For both rigid (man-made) and non-rigid sets, our technique outperforms SHED in precision with lower standard deviation. The lower standard deviation further shows the stability and robustness of

Rigid	MLG	SHED	pairs	layers
lamps	85.2%	76.7%	30	8
vases	86.0%	63.0%	20	4
candles	86.2%	71.3%	11	4
planes	83.5%	60.8%	11	8
average	85.3%(1.2)	69.6%(7.4)		

Table 3.1: Precision (std. dev.) on rigid (man-made) set.

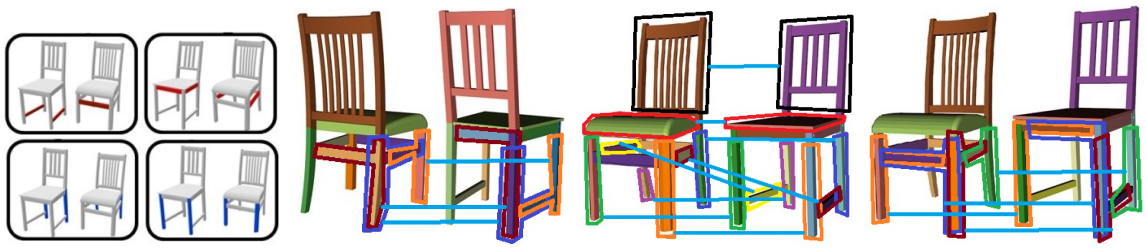
Non-Rigid	MLG	SHED	pairs	layers
wolf	97.2%	59.0%	3	8
human	83.3%	62.7%	7	8
horse	85.3%	81.6%	6	8
centaur	90.9%	67.5%	4	8
average	87.5%(6.2)	68.8%(9.9)		

Table 3.2: Precision (std. dev.) on non-rigid set.

our technique.

Our annotation focuses on the outputs of the two techniques. We plan to release the annotation results and codes to the research community, for inspection, comparison and downstream applications.

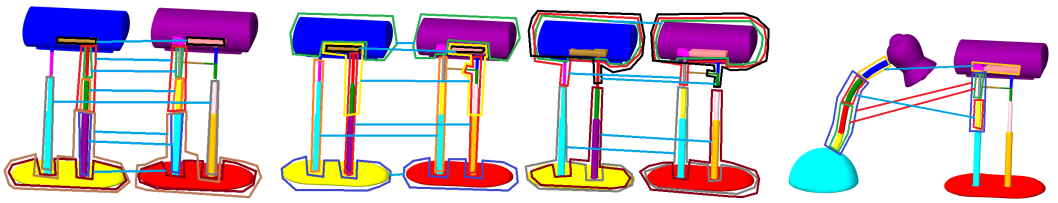
3.7 Discussion



(a) (b) (c) (d)

Figure 3.14: (a) image courtesy of [37]. (b)(c)(d) are our method matching results.

Here, we further provide a brief comparison of our technique with the state-of-the-art [37]. Figure 3.14a shows the matching result of two chairs (image courtesy of [37]). In the figure, the red side panels are mismatched to the front panels between chairs. The technique proposed



(a) Results of changed δ (b) Results of changed c_2 (c) Large difference in shapes

Figure 3.15: Results from adjusting different parameters, compared to Figure 3.10. The $\delta 1$ used in (a) is 0.5 (anchor stage) and 0.7 (final stage) - note the symmetric issue, (b) uses the same δ as (a) and further reduce threshold c_2 to 0.2. (c_2 is a threshold used in diffusion pruning for greedy pruning.) Here, relaxing c_2 leads to only higher layer matchings. (c) shows our method performs poorly when the inputs have large topological and geometrical difference.

in [37] is a top-down matching technique assuming perfect input segmentations. The technique seeks the best split along the component tree. To our knowledge, it does not use higher layer matching to support lower layer matching which would have solved the mismatch.

We tried our technique on the same set of chairs by manually labelling the left (right) chair into 10 (12) segments, according to the initial segmentation as shown in Figure 3.14a. We then apply our technique using our two-stage diffusion pruning (DP) with a 2-layer MLG for each chair. We only use 2 layers because the chair is highly complex with high connectivity for each segment. If we use 3 or more layers, the number of internal nodes grows to 1000+ which is too slow to compute. Due to the lack of high-layer nodes, we cannot apply our voting step. However, simply using the proposed two-stage DP step yields perfect matching result (Figures 3.14b-3.14d). This answers our research question that considering merged nodes in the MLG hierarchy can improve matching results. As the source code and data for [37] are not available, further comparison is not possible. Having said that [37] cannot support non-rigid shapes, and assume consistent input segmentations. Our technique is comparatively more flexible. It can handle non-rigid shapes and inconsistent input segmentations.

There are limitations in our technique however. One issue is the sensitivity to the chosen parameters. Figure 3.15 compares the results in Figure 3.10 with different parameters. In Figure 3.15a, we tighten the δ_1 threshold (i.e., use smaller local isometric disk). Though the volunteers consider the results correct, it leads to more local matching and cannot avoid the

symmetry issue. In Figure 3.15b we further reduce c_2 (a threshold used in [3] for the last greedy pruning step), the matching results all shift to higher layers, with no bottom-layer one-to-one correspondences found. Figure 3.15c further shows that our technique does not perform well when the input shapes have large difference in topology and/or geometry.

Our current un-optimised code is too slow to handle shapes with a large number of input segments. There is an exponential growth in the number of possible internal nodes in MLG, with respect to the number of input segments. We constrain the MLG using volume, but it can sometimes miss some matchings (e.g. the leg in Figure 3.12). In the future, we hope to develop a more robust hierarchical representation than MLG to reduce the search space. Another direction is to incorporate our bottom-up idea into a top-down approach [37]. Further, our technique consists of quite a few parameters. Although most of them are fixed to default settings, we plan to develop a more robust technique and make it more generic to a large variety of input shapes and inconsistent segmentations.

In the future, we are going to use better geometrical features to enhance our matching techniques. We also need to condense the size of MLG, which means a better merging technique is necessary. Based on the simplified MLG we can further investigate the convergence property of our technique.

3.8 Conclusion

In this chapter, we propose a novel segment-wise matching technique that can handle shapes with inconsistent (over-/imperfect) input segmentation. Our idea is to greedily optimise matchings that are geometrically, topologically and hierarchically consistent. To do so, we develop a multi-layer graph (MLG) representation to store the possible merging arrangement of segments. Apart from geometric and topological consistency, we explicitly seek consistency in the hierarchical segment merging space. Experimental results demonstrate the effectiveness of our technique when compared to two state-of-the-art methods.

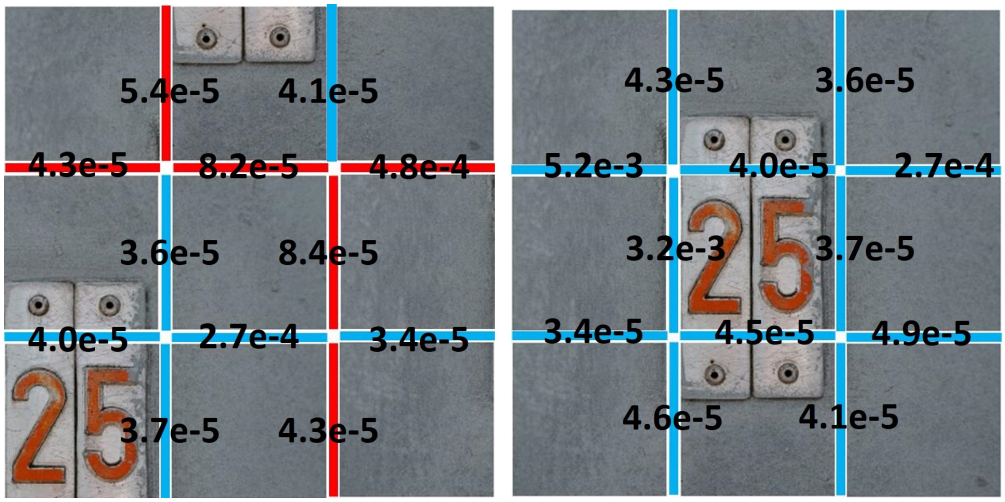
Chapter 4

Robust and Flexible Puzzle Solving with Corner-based Cycle Consistent Correspondences

Contents

4.1 Introduction

Solving jigsaw puzzles is a classic problem in computer vision. In 1964, [21] introduced the first algorithm for matching puzzle pieces. Since then, approaches have focused on using shape and colour information [44, 45] for puzzle solving. Puzzle solving has great applications in many research areas, like forensics [107, 108] and archaeology [109, 110], to recover documents or art works from small fragments.



(a) Assembled result by [20]. Red edges show incorrect matching pairs in greedy assembly.

(b) Correct assembled result obtained by our proposed technique.

Figure 4.1: Comparison between [20] and our proposed technique on square puzzle solving. The number on each edge shows the MGC similarity score between a pair of pieces.

Techniques to solve a jigsaw puzzle consist of two steps: i) computing constraints (e.g. colour-based similarity between puzzle pieces) and ii) assembling puzzle pieces via some optimisation technique. Notable examples include [20] which introduces the novel Mahalanobis Gradient Compatibility (MGC) measure to compute the similarity between puzzle pieces, and a minimal spanning tree (MST) [111] approach to assemble similar pieces in a greedy manner. Based on colour space normalisation, [22] proposes a global approach to assembling similar

puzzle pieces. Their compatibility measure is based on a thin region (often 1 column of pixels of the edge) of each piece. These two measures are frequently used in subsequent works for puzzle solving [58, 59, 103, 102]. More recently, the loop constraint [58, 61, 64] was proposed to enforce cycle consistency when pieces are matched, and good performance was demonstrated.

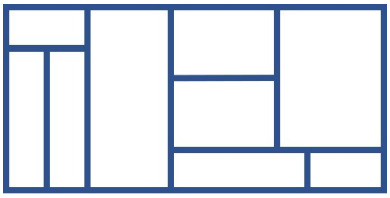


Figure 4.2: Rectangular pieces with arbitrary sizes are challenging for edge-wise similarity measures and assembly techniques.

From the literature, we made two observations. First, much of the previous work focuses on puzzles with square pieces of the same size but they may not apply to puzzle solving with rectangle pieces of arbitrary sizes (Figure 4.2). The problem of solving such puzzles is arguably harder with a larger search space because of the arbitrary edge lengths. It challenges most of the existing edge-wise similarity measures. Second, even though the loop constraint is powerful, we observe that many of the existing works assume some form of input regularity, and either build loops explicitly from square pieces which would be slow, or use the loop constraint by casting puzzle assembly in a sophisticated optimisation. These techniques however are not easy to extend to arbitrarily shaped puzzle pieces. These observations motivate the research question in section 1.3.3 and 1.3.4.

Instead of using the whole edges of pieces for puzzle assembly like in existing work, we investigate if corners of puzzle pieces can be used. Next we cast the problem of discovering loops in possible puzzle pieces as a cycle consistent correspondence problem [16]. Once we identify good pairwise corner-wise correspondences, we adapt minimum spanning tree [20] for puzzle solving. Our results show that the approach can improve the performance of [20] which uses MGC alone.

We provide an overview of our method in Section 4.2, we show how we model corner-wise matching in the MatchLift framework [16] for square puzzle solving in Sections 4.3 and 4.4.

We evaluate our method in Section 4.5. Section 4.6 illustrates one example how puzzle with rectangular pieces of arbitrary sizes can be solved. We discuss limitations and future work in Section 4.7, and conclude in Section 4.8.

4.2 Method Overview

Figure 4.3 shows the pipeline of our technique for solving puzzles. The input image to our method is first sliced and shuffled into (e.g. square) puzzle pieces. Our method further breaks each puzzle piece into four (2-by-2) corners, by subdividing each edge of a piece into 2 sub-edges (Figure 4.5, Section 4.3.2). Then we use MGC to compute the similarities between all possible pairs of sub-edges. We treat these pairs as correspondences. Section 4.3.3 presents how we use MatchLift to identify cycle consistent correspondences. Section 4.4 discusses how we refine the respective MGC scores of correspondences identified by MatchLift, and finally solve the puzzle using minimum spanning tree. Further, in Section 4.6, we discuss how we extend our technique to solve puzzles consisting of rectangular pieces of arbitrary sizes. In this chapter, we assume all pieces have known orientation with unknown position (so called Type I puzzle problem [20]).

4.3 MatchLift and Puzzle Solving

MatchLift [16] is a convex optimisation technique to find cycle-consistent correspondences from a set of noisy input. For example, for 3D reconstruction of a chair, it is critical to estimate depth by computing reliable point-to-point correspondences across a collection of images of the same chair from different views. Key point descriptors (such as SIFT) can generate correspondences, but inconsistent correspondences cannot be avoided. [16] can identify cycle-consistent correspondences across multiple images. The idea is to encode all pairwise correspondences between images in a permutation matrix. Then it applies SDP (semi-definite programming) with relaxed binary constraint and sparsity to enforce cycle consistency.

Figure 4.3: Pipeline of our method. We slice the input image into pieces. For each piece, we break each edge into two sub-edges. We use sub-edges to represent four corners on each square piece. We use MGC to determine the similarities between all possible pairs of sub-edge based corners. We treat these pairs as correspondences, use MatchLift to identify cycle-consistent correspondences and update their MGC scores. Finally we apply minimum spanning tree [20] on the updated MGC scores for puzzle assembly.

For example, as shown in Figure 4.4, we have n input objects and there are two correspondences between object 1 and object 2. MatchLift has introduced a virtual augmented universe m. The matching between two objects can be decomposed into two binary matching matrix $\mathcal{X}_1$ and $\mathcal{X}_2$, where the rows are indicating the object and the columns are indicating the augmented universe m. For n input objects and we form all object-wise matching into matrix $\mathcal{M}$ (the diagonal elements are self-matching matrix and non-diagonal elements are object-wise matching), this matching matrix $\mathcal{M}$ can be decomposed into

$$\mathcal{M} = \mathcal{X}\mathcal{X}^T, where$$

$$\mathcal{X} = (\mathcal{X}_1, \mathcal{X}_2, ..., \mathcal{X}_n)$$

$$rank(\mathcal{M} = m) \tag{4.1}$$

The rank of $\mathcal{M}$ is the size of augmented universe m, which means m cliques in the graph partitioning. Furthermore, a confident correspondence can be discovered by using a low-rank, PSD (positive semi-definite) programming to infer/approximate the input. The m can be estimated by using spectral technique in finding the largest drop between eigen-values of input matching matrix $\mathcal{M}^{in}$. Therefore, the formulation can be further lifted one more dimension as,

$$\begin{bmatrix} m & 1^T \\ 1 & \mathcal{M} \end{bmatrix} = \begin{bmatrix} 1^T \\ X \end{bmatrix} \begin{bmatrix} 1 & X^T \end{bmatrix} \succeq 0 \tag{4.2}$$

Then, they use the input matching $\mathcal{M}^{in}$ to discover globally consistent matching $\mathcal{M}$ as the output. The discovered matrix $\mathcal{M}$ should be close to the input matrix $\mathcal{M}^{in}$, and an $\mathcal{L}_1$ regularisation term is needed to reinforce the sparsity.

$$maximize_{\mathcal{M} \in \mathbb{R}^{n \times n}} \sum_{(i,j) \in \mathcal{G}} \langle \mathcal{M}^{in}_{i,j}, \mathcal{M}_{ij} \rangle - \lambda \langle 1 \times 1^T, \mathcal{M} \rangle, subject\ to,$$

$$\mathcal{M}_{ii} = \mathcal{I}_m, \ \ 1 \leq i \leq n,$$

$$\mathcal{M} \geq 0,$$

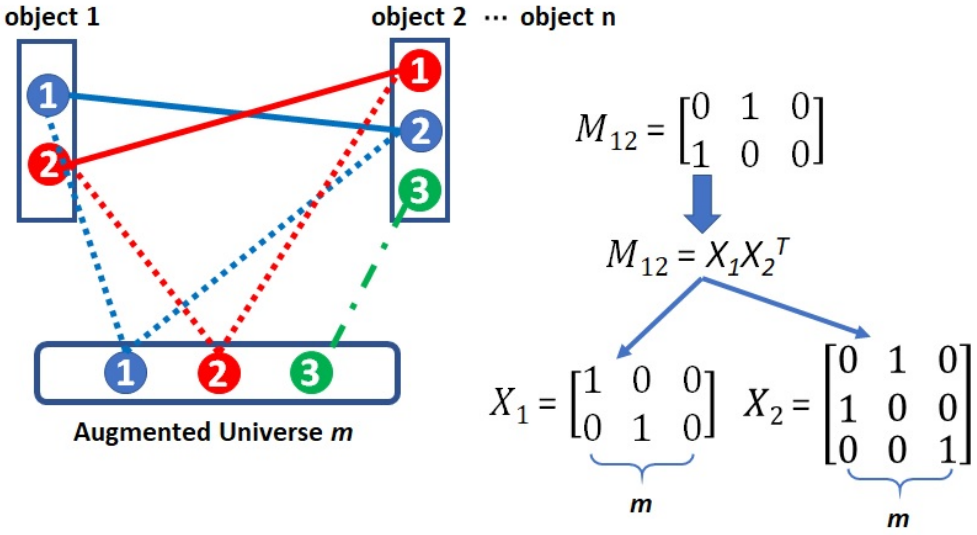


Figure 4.4: An example of MatchLift. M_{12} is the matching matrix of object 1 and object 2.

$$\begin{bmatrix} m & 1^T \\ 1 & \mathcal{M} \end{bmatrix} \succeq 0 \tag{4.3}$$

$\mathcal{I}_m$ is a self-matching matrix in the $\mathcal{M}$. The control parameter λ can balance the compatibility between the input matching and e sparsity structure. The experiments in the [16] shows that MatchLift is a parameter free formulation and it is not sensitive to the parameter λ. The multiple iterations of eigendecomposition are needed for solving SDP in the MatchLift, which means the time complexity for each iteration is $O(m^3)$.

In this chapter, we use MatchLift to find reliable cycle-consistent correspondences for puzzle solving. The cycle-consistent correspondences indicate a set of reliable similarity scores, which will benefit the assembly stage in puzzle solving. In our modelling, we treat each corner as an object (similar to one of the images in the chair reconstruction example), and pairwise matching of sub-edges as correspondence between corners (similar to point-to-point correspondences between images). Our contribution is to model piece matching in the puzzle problem as a corner-wise cycle-consistent correspondence problem in the MatchLift framework. We show that it can handle square and rectangular puzzles of arbitrary sizes.

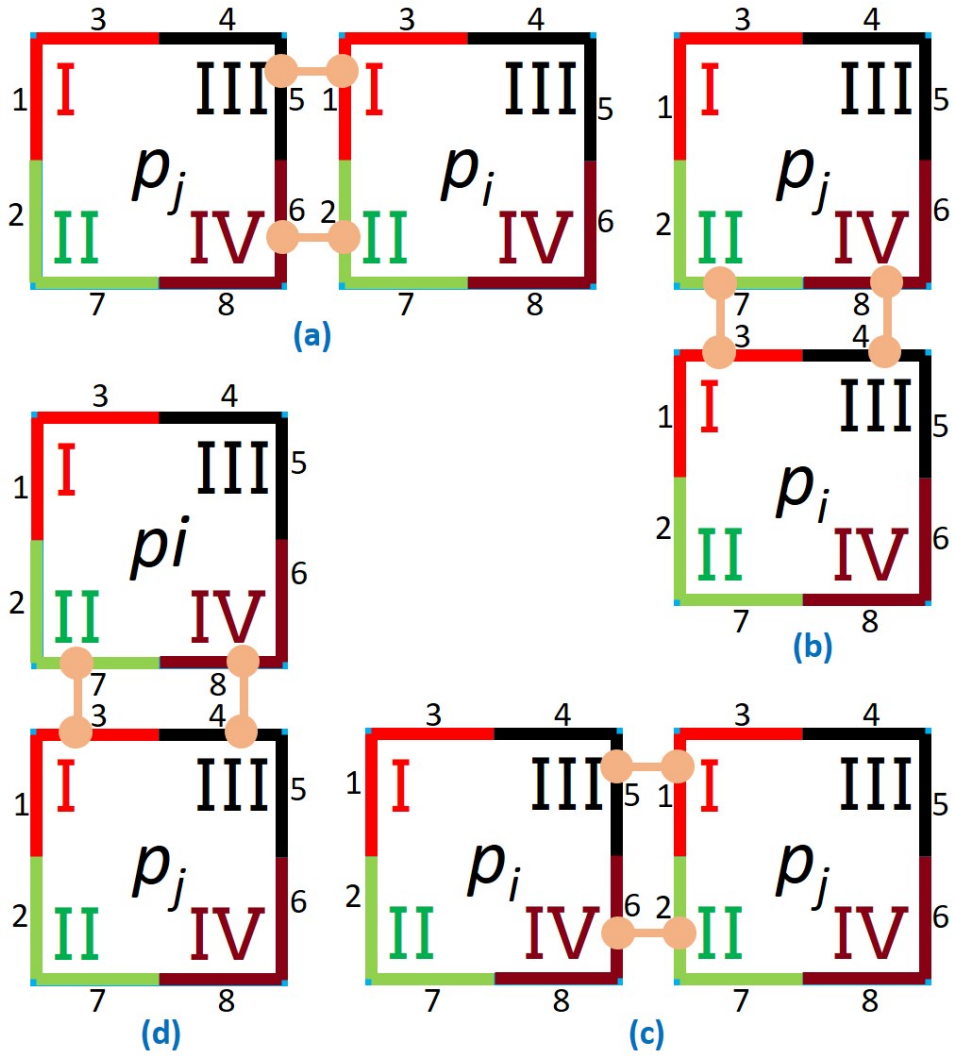


Figure 4.5: An ordering scheme to generate correspondences of sub-edges between puzzle pieces. We fix the position of p_i and move p_j around of p_i.

4.3.1 Computing MGC Scores

Our technique builds on MGC scores [20] which we briefly discuss here. MGC is a gradient-based compatibility measurement between puzzle piece edges (all pieces must have the same size). For an edge, it first defines a matrix of colour distribution with dimensions $px \times 3$, where px is the number of pixels of a piece edge with 3 colour channels (red, green, blue). For a pair of edges on two square pieces, MGC determines a compatibility score by computing

Mahalanobis distance between their colour distribution matrices.

4.3.2 Modelling Puzzle Pieces by Corners

Next we introduce the sub-edge. Let $P = \{p_1, ..., p_n\}$ be the set of all input puzzle pieces. For each puzzle piece $p_i \in P$, we break each edge into two sub-edges (for generating corners later). Our method is also available for breaking each edge into other numbers of sub-edges (three, four, five, or others). However, more sub-edges per edge will increase the timing in computing. Based on our experiments, two sub-edges per edge produce good results. We then only break each edge into two sub-edges. There are four edges of a square piece and in total eight sub-edges per piece. We label each sub-edge in a fixed order as shown in Figure 4.5. We further define $e_a(p_i)$ as an operator to return the sub-edge from p_i where $1 \leq a \leq 8$. For each pair of pieces p_i, p_j, we consider eight possible correspondences associated to the sub-edges of p_i, p_j based on an ordering scheme as shown in Figures 4.5 (a)-(d). Beginning from the left two sub-edges of p_i and the right two sub-edges of p_j, we define correspondences $c_k = (e_1(p_i), e_5(p_j))$ and $c_l = (e_2(p_i), e_6(p_j))$ (shown as tan coloured correspondences in Figure 4.5 (a)). Following the ordering scheme, we can define eight correspondences for p_i and p_j, and we repeat the procedure for all pairs of pieces to compute the set of input correspondences C. For each correspondence $c_k \in C$, where $1 \leq k \leq 8n(n-1)$, we define the similarity between the two sub-edges using MGC score. MGC scores have a large range (the maximum value might be ten thousand times larger than minimum). We normalise them into [0,1]. After normalisation, scores close to 1 mean two sub-edges are highly similar. Take c_k for example, our measure is thus $sim(c_k) = MGC_{normalised}(e_1(p_i), e_5(p_j))$. Other cases can be similarly defined.

Next, we define the corners of pieces as units for puzzle solving. We use $p_i^\alpha \in P$ to indicate a corner on a piece, where $\alpha \in \{I, II, III, IV\}$, as shown in Figure 4.5. For example, assuming there are ten pieces in a puzzle, the corner II on the tenth piece is labelled as p_{10}^{II}, and it contains two sub-edges $e_2(p_{10})$, $e_7(p_{10})$. We define $v(p_i^\alpha, p_j^\beta)$, where $\alpha, \beta \in \{I, II, III, IV\}$, as the corner-wise similarity score of two corners p_i^α and p_j^β. Since the orientation of input pieces

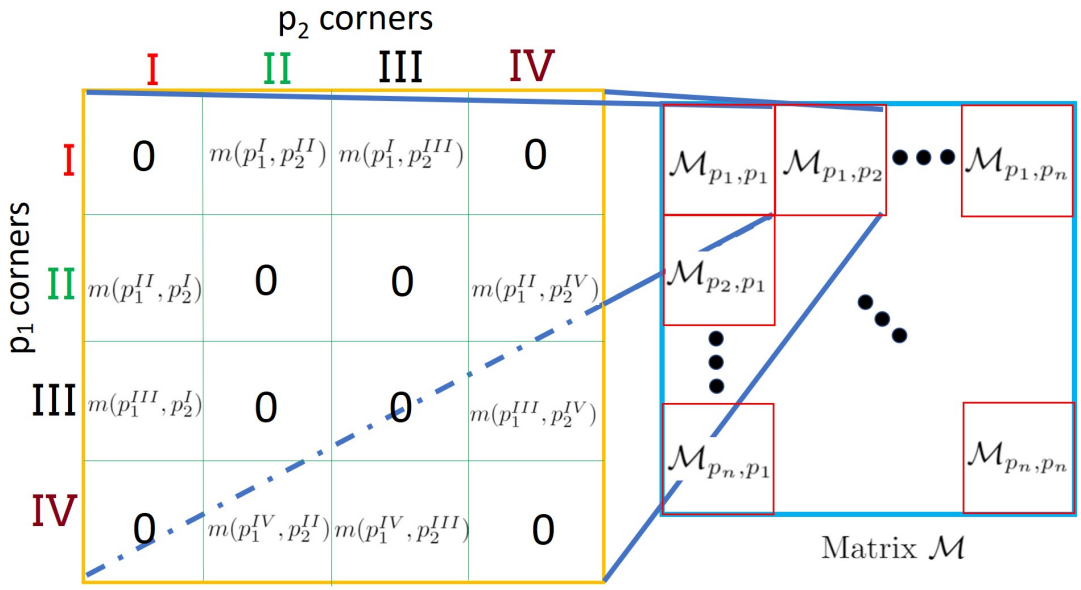


Figure 4.6: Example of matrix $\mathcal{M}$ with n puzzle pieces. All diagonal red blocks are self matching between a pair of the same piece. Non-diagonal red blocks contain corner-wise similarity measures of pieces.

is known (Type I puzzle), some corners are incompatible with each other such as p_i^I and p_j^I. For incompatible corners, we set $v(p_i^\alpha, p_j^\beta) = 0$. $v(p_i^\alpha, p_j^\beta)$ can be summarised as

$$v(p_i^\alpha, p_j^\beta) = \begin{cases} sim(c_k), & \text{if } c_k \in C \\ 0, & \text{otherwise,} \end{cases} \tag{4.4}$$

We encode the corner-wise similarity in a block matrix as the input of MatchLift. Let $\mathcal{M}_{p_i,p_j}$ be a 4×4 matrix, which is shown in Figure 4.6 (left). Given a puzzle with n pieces, we can encode all $\mathcal{M}_{p_i,p_j}$ blocks into a piece-wise similarity matrix $\mathcal{M}$ of dimension $4n \times 4n$ (i.e. $\mathcal{M}_{p_i,p_j} \subset \mathcal{M}$ in Figure 4.6 (right)). It is arranged such that the non-diagonal block $\mathcal{M}_{p_i,p_j} \subset \mathcal{M}$, where $i \neq j$ contains all corner-wise MGC scores $m(p_i^\alpha, p_j^\beta) \in \mathcal{M}_{p_i,p_j}$ between pieces p_i and p_j. The diagonal elements $m(p_i^\alpha, p_i^\alpha) \in \mathcal{M}$ represent self-matching between a pair of the same corner p_i^α and p_i^α. We set those elements as 1. In summary, we define the element $m(p_i^\alpha, p_i^\alpha) \in \mathcal{M}$ as

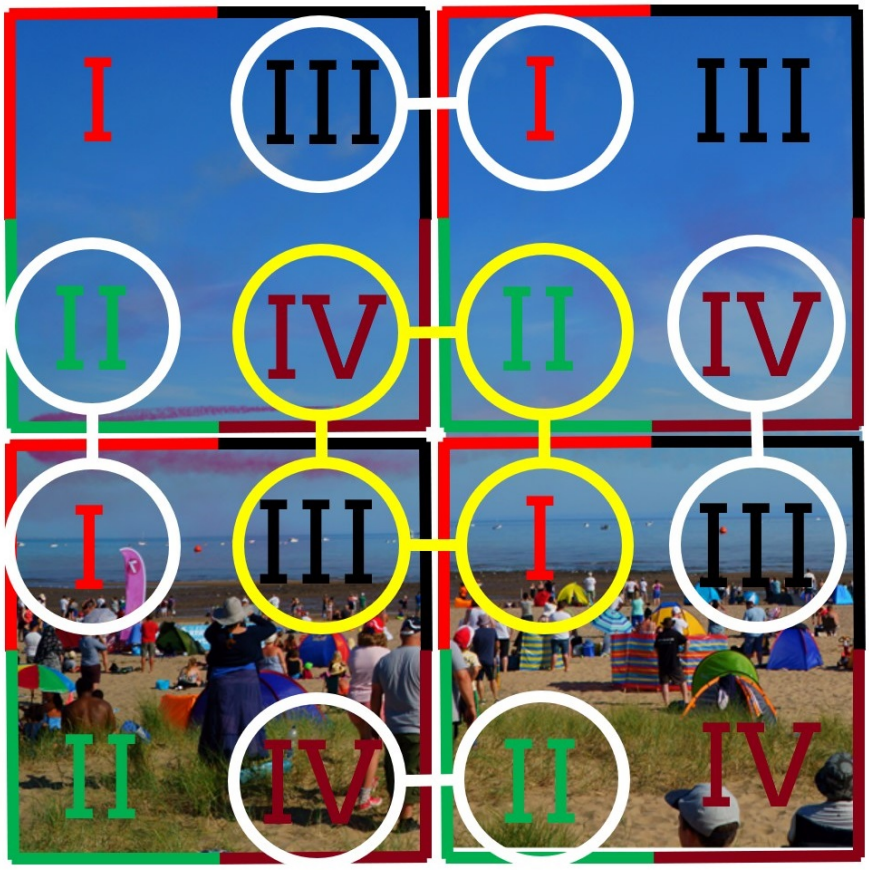


Figure 4.7: Yellow lines and circles indicate a four-cycle correspondences between four corners. White lines and circles are two-cycle correspondences between two corners.

$$m(p_i^\alpha, p_j^\beta) = \begin{cases} v(p_i^\alpha, p_j^\beta), & i \neq j,\ v(p_i^\alpha, p_j^\beta) \geq t_1 \\ 1, & i = j,\ \ \alpha = \beta \\ 0, & \text{otherwise} \end{cases} \tag{4.5}$$

and t_1 is a user defined threshold to accept correspondences with good MGC scores. $\mathcal{M}_{p_i,p_j}$ represents partial matching whilst $\mathcal{M}$ represents the full matching of input pieces. This matrix can then be optimised using MatchLift framework, using SDP [16].

4.3.3 Corners and Cycle Consistency

Our intuition of using corners in the MatchLift framework to handle square puzzle pieces is that it can find two-cycle (direct correspondence, white) and four-cycle consistent correspondences (yellow) as shown in Figure 4.7. We mark the positions of corners to indicate the matching between sub-edges. For example, in Figure 4.7 each bottom-right corner contains sub-edges 6 and 8 and each bottom-left corner contains sub-edges 2 and 7. If there is a matching between sub-edges 6 and 2, then it means corner IV and corner II have been matched. Since we have cycle consistency as a constraint there will not be displaced-matching, such as sub-edge 6 will not match to sub-edge 1.

Compared with sub-edges, using corners can reduce the size of $\mathcal{M}$. For example, for n square pieces and we break each edge into f sub-edges, the dimensions of the resulting matrix $\mathcal{M}$ based sub-edge matching will be $4fn \times 4fn$. By adopting this corner-wise approach our $\mathcal{M}$ is only $4n \times 4n$.

4.4 Assembling Pieces

After running MatchLift, the matrix $\mathcal{M}$ will be updated. The elements $m(p_i^\alpha, p_j^\beta) \in [0,1]$ with 1 indicating a confident correspondence that forms a cycle whilst 0 means the associated corner matching is not cycle consistent. Confident correspondences with $m(p_i^\alpha, p_j^\beta) \geq t_2$ are then returned and t_2 is a user defined threshold.

We follow [20] to assemble puzzle pieces using minimum spanning tree (MST), which is a greedy technique. Based on the extracted corners from MatchLift we can infer the matching between sub-edges. If two sub-edges are matched we set the MGC score of the whole corresponding edge with a small value (by multiplying 0.000001 to the MGC scores) so that MST can prioritise the matching for piece assembling earlier. For example, in Figure 4.7 sub-edges 6 and 2 of the bottom pieces are matched. The MGC score between the entire right edge (containing sub-edges 5 and 6) and the entire left edge (containing sub-edges 1 and 2) is reduced. This allows MST to prioritise such matching to be considered first leading to correct assembly.

Image	100	144	196	assemble	t_1 t_2
1	100 79	94 90	100 41	6.3s 17.9s	0.2 0.2
2	100 81	100 55	100 37	5.0s 18.5s	0.1 0.9
3	91 94	50 51	31 47	15.9s 16.5s	0.4 0.9
4	73 93	57 39	41 84	17.4s 16.4s	0.2 0.7
5	54 32	46 37	51 35	17.2s 17.0s	0.3 0.9
6	96 83	76 74	49 52	12.8s 12.8s	0.5 0.8
7	92 100	64 62	49 58	11.5s 10.7s	0.3 0.9
avg (%)	87 80	70 58	60 51	12.3s 15.7s	

Table 4.1: We compare our method and [20] by showing percentage of correctly assembled pieces with 100, 144 and 196 pieces input. The assemble column shows the time requires to run MST for assembling. t_1, t_2 are parameters we used in our method. Our and [20] results are shown in red and blue respectively.

The detailed information about MST and how to use MST to assemble puzzle pieces can be found in [111, 20].

4.5 Evaluation

We evaluate our method against [20] in this section. First, we perform a quantitative evaluation on a small collection of images in Section 4.5.1 to compare the success assembly rate of our technique against MGC alone. Section 4.5.2 shows some assembled results from both methods as qualitative evaluation. For both methods we use the same number of puzzle pieces and images. We also evaluate with puzzle pieces of different resolutions in our experiments.

4.5.1 Quantitative Evaluation

We evaluate our method against [20] on seven images of varying numbers of pieces and resolutions. We use five of our own (high and low resolution) images and two images from public data set [22] (low resolution). We slice each image into 100, 144, and 196 pieces as the input of both methods. The higher number of pieces leads to lower resolution of each piece. Though MatchLift [16] in theory has good tolerance to random outliers, the stability of MGC is low. When there are too many incorrect correspondences, it would lead to poor results.

We therefore need to adjust two of our parameters t_1 and t_2. t_1 controls the number of correspondences accepted as input to MatchLift (most initial correspondences are incorrect). t_2 controls how confident we accept the matching results from MatchLift. These parameters are somehow dependent on the resolution of images and stability of MGC. For t_1 we try 20 values $0.6 \leq t_1 \leq 1$ and 9 values for t_2 where $0.5 \leq t_2 \leq 0.95$ and report the best assembled results in Table 4.1. The overall process is time consuming. On average, it takes five hours (i7-6700 4.0GHz CPU with 32GB memory) per image in this experiment. The long time computing is mainly caused by the low usage of CPU (only 20%). In the future, it can be addressed by using C++ instead of MATLAB (low efficiency in the loop computing) for higher performance in iteratively computing.

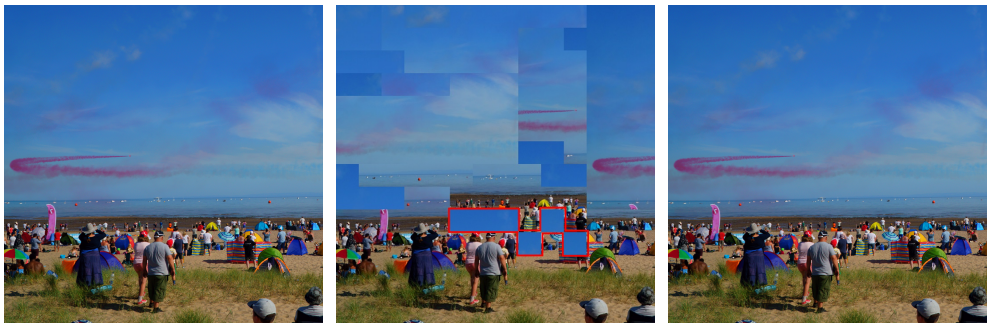
We use the ground truth coordinates of each piece to evaluate the assembled results, so-called the direct comparison [20]. When an assembling technique misaligned a large assembled region, the percentage of correctly assembled pieces will reduce significantly. The evaluation results are shown in Table 4.1. Our initial results show that our method can produce better results than [20] with the proper parameters. Because our technique recovers better piece matching, the MST assembling step is faster than using MGC alone. Nevertheless, we hope to discover the best parameter settings automatically for our technique in the future, for example, to investigate the spectrum of the matrix $\mathcal{M}$ [16].

4.5.2 Image Resolution and Puzzle Solving

Next, we qualitatively evaluate our technique on high resolution images (all input images have a resolution above 2700 by 2700) in Figure 4.8. In Figures 4.8b and 4.8c both images are assembled from 49 pieces. For regions with distinctive texture, such as clouds at the low part of the image, MGC and MST perform well and produce good assembled results. However, MGC produces unreliable scores around the white smoke and cloud at the top. This leads to incorrect assembled results. In our case, after MatchLift refinement, MST can assemble 100% correct results. When we increase the number of puzzle pieces to 100, MGC becomes

(a) Original image (b) Assembled result by directly using MGC (c) Our method result

(d) Input 100 puzzle pieces (e) Assembled result by directly using MGC (f) Our method result



(g) Input image (h) Assembled result by directly using MGC (i) Our method result

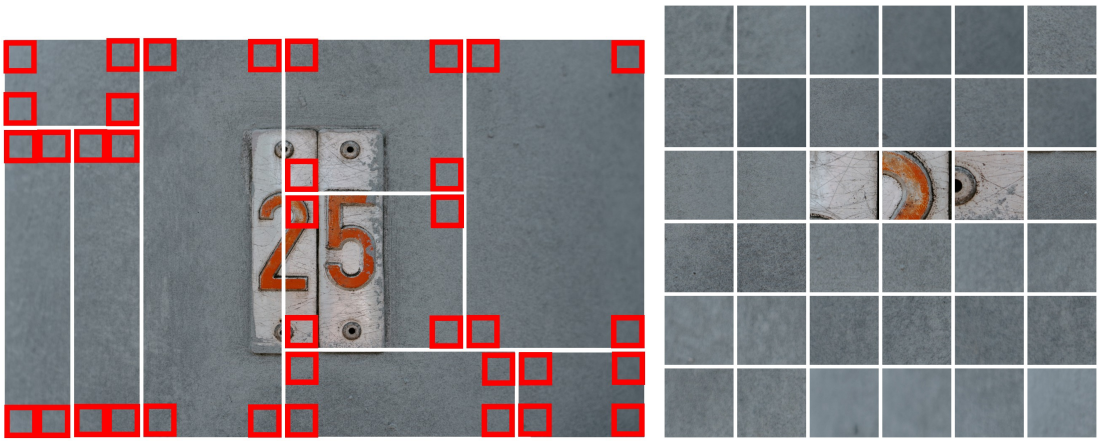
(j) Input image (k) Assembled result by directly using MGC (l) Our method result

Figure 4.8: Experimental results on puzzles built from high resolution images.

(a) Input image (b) Assembled result by directly using MGC (c) Our method assembled result

(d) Input image (e) Assembled result by directly using MGC (f) Our method assembled result

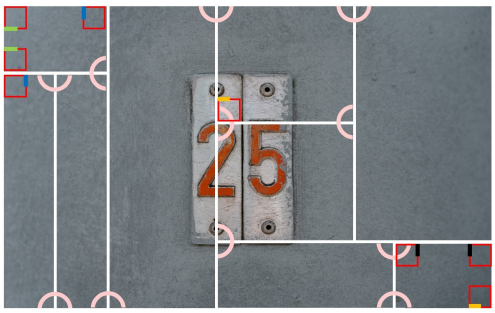
Figure 4.9: Directly using MGC causes more incorrect assembled results in low resolution images or images containing indistinctive pieces. Our method maintains 100% correctness in assembled images.



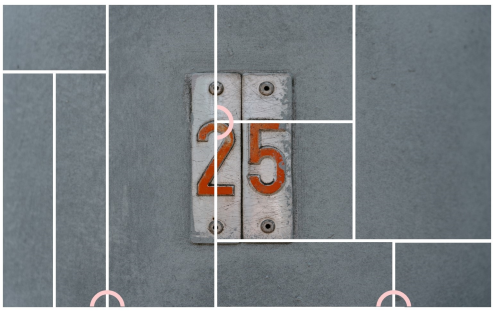
(a) Rectangular pieces of arbitrary sizes. We compute pairwise similarity by using corners as labelled in the red boxes.

(b) We compute all possible corner-wise similarities.

Figure 4.10: We use corners on rectangular pieces with arbitrary sizes.



(a) Result of our method. Correct corner-wise matchings are labelled as pink half-circles. Incorrect matchings are labelled as other colours.



(b) All correct matchings obtained by [20]. These matchings are insufficient to solve the puzzle. To avoid clutter, we do not show incorrect matchings (since they are too many).

Figure 4.11: By using MatchLift on corners we can find reasonable matching between rectangular pieces with arbitrary sizes.

unreliable (Figure 4.8e). Meanwhile, our technique can still discover confident matching. This allows MST to assemble 100% correct results (Figure 4.8f).

Low distinctive regions are challenging for MGC. The resolution of Figure 4.8g is 2700 by 2700, and there are 196 puzzle pieces. Similar to Figure 4.8a the sky is difficult to be assembled by MGC. When puzzle pieces become smaller (since the number of pieces increased), the number of pixels to compute in MGC is fewer. MGC will return more unreliable scores. For example, in the red highlighted region of Figure 4.8h, MGC considers the sky and cloth are highly similar.

Figure 4.8j is another high resolution image of resolution 3840 by 3840. It consists of 64 pieces. Though the resolution is higher with fewer pieces, MGC does not perform well on under-exposed regions and leads to incorrectly assembled top-left region.

Figure 4.9 evaluates the two methods with images of low resolution. Figure 4.9a has a resolution of 1200 by 1200, and size of 289KB. We slice it into 144 pieces. In Figure 4.9b, without MatchLift refinement, the technique struggles to assemble regions around the deck, gun and road pieces.

Our method models puzzle solving based on corners and cycle consistency constraint. We can better handle unreliable MGC scores of such pieces and assemble the correct results in

Figure 4.9c. Similar situation appears in Figures 4.9e and 4.9f. Without MatchLift refinement, MST cannot find a correct assembling of the gun barrel and the camouflage netting behind the vehicles.

4.6 Puzzle Solving for Rectangular Pieces of Arbitrary Sizes

We show one interesting example of applying our technique to solve puzzles of rectangular pieces with arbitrary sizes (to our knowledge no existing techniques can handle such challenging case). Since the pieces have arbitrary sizes, our earlier square puzzle slicer does not apply. To produce the input puzzle pieces, we manually slice the image as shown in Figure 4.10a into 9 pieces. Next, we manually select 36 local regions (Figure 4.10b) to represent the four corners of all 9 pieces (red boxes in Figure 4.10a). Similar to square puzzle examples, for each corner, our technique breaks each edge into two sub-edges and computes similarity to other corners/pieces. We encode all similarity scores and pass them to MatchLift to obtain corner-wise matchings on these rectangular pieces.

Our method outputs 15 corner-wise correspondences. 11 of them are correct and are visualised as pink half circles in Figure 4.11a. The four incorrect corner-wise matchings are visualised as coloured bars with associated local regions (the red boxes) in Figure 4.11a. Among these four mismatched pairs, the green pair and the black pair are respectively from the same rectangular piece and can be removed as it is not possible to assemble corners/sub-edges from the same rectangular piece. The matched sub-edges of the blue pair are located inside the two rectangular pieces. The sub-edges/whole edges that are inside pieces should not be used in the assembling, because an assembling is based on the borders of each piece. Similarly, one of the matched sub-edges in the orange pair is also inside the rectangular piece. These four mismatched pairs can be easily removed in a pruning scheme as post-processing. On close inspection, we argue that such a puzzle with rectangular pieces of arbitrary sizes can be assembled correctly using the returned matched corners (visualised as the pink half circles) as shown in Figure 4.11a.

Figure 4.11b shows all the correct matchings obtained by [20] with the same input of Figure 4.10b. To avoid clutter, we visualise all (only three) correct assembled corners (the pink half-circles). Since most corners are incorrectly assembled, we cannot refine/infer the results as we did for Figure 4.11a.

4.7 Limitation and future work

Long computational time is an issue for our current technique. MatchLift requires multiple eigendecompositions which can be slow for puzzles with a large number of pieces. Another problem is that due to the nature of the images (e.g. distinctiveness, texture, resolution), our technique requires some parameter adjustment to obtain the best results, tailoring to the image properties. We hope to investigate and develop a parameter-free technique.

Currently, we are using square puzzle pieces with known rotations. We can use the same modelling idea to solve puzzle pieces with unknown rotation (Type II puzzle [20]). In that way, the matrix $\mathcal{M}$ will be denser than the current configuration. We also would like to try non-rectangular pieces, or a mixture of square, triangle and polygonal pieces. Since our method models the puzzle problem with corners, it can be extended to such challenging examples, which existing techniques cannot solve. Given our promising cycle consistent corner constraint, we hope to develop a fully automatic technique to solve such problems.

4.8 Conclusion

In this chapter, we try to solve square puzzle problems by considering two novel ideas. First, we use corner-wise correspondences, rather than edge-wise correspondences. Second, we model the subsequent puzzle problem into the MatchLift framework, solved via a semi-definite programming approach to recover cycle-consistent correspondences. We then refine the confident scores of these correspondences to promote their use for piece assembling early via a minimum spanning tree puzzle solver. Experimental results show that our technique can achieve

better results than the non-refined cases. Finally we show that our technique can be extended to puzzles consisting of rectangular pieces of arbitrary sizes. It is an exciting and arguably more challenging problem. Our technique can still show promising initial results.

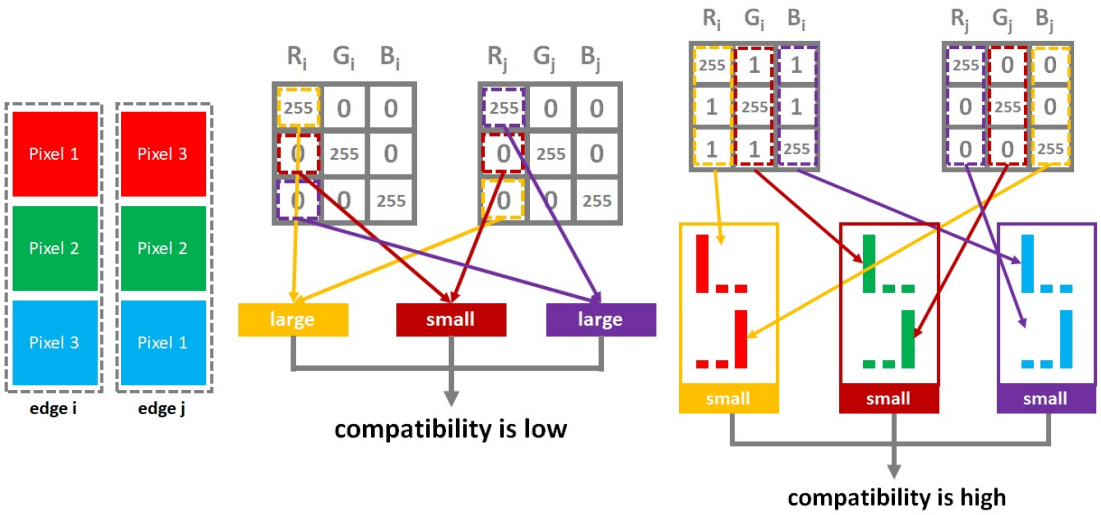
Chapter 5

Solving Variously Shaped Puzzles with Diffusion Pruning

Contents

5.1 Introduction

In this chapter, we introduce a new puzzle solving technique to handle variously shaped pieces. Puzzle solving techniques have two stages: compute similarities between all input puzzle



(a) Input edges (b) Traditional row-approach (c) Our histograms distribution based column-approach

Figure 5.1: We use histograms distribution to estimate similarity. This column-approach avoids the index ordering issue of puzzle pieces.

pieces, and assemble puzzle pieces based on computed similarities. As mentioned in section 2.2.3, the traditional puzzle solving techniques use colour compatibility between a pair of pixels to estimate the similarity. For example, shown in Figure 5.1b, for two given edges, all pixels on both edge will be converted into two colour matrices. The two matrices must have the same dimensions. The number of rows is decided by the number of pixels of edge, and the number of columns is decided by the colour space. In this example we use RGB space and there are three columns for each matrix. Next, they compute colour compatibility for each pair of channels with the same colour. Inside of each pair of channels, values from pixels that have the same index will be computed, to generate a distance. When all pairs of values have been computed, the channel-wise compatibility is measured. The final edge-wise similarity score is defined by grouping all channel-wise compatibility. We name this approach as the row-approach.

We have stated the limitations of row-approach in section 2.2.3 and section 1.3.3. To over-

come the limitations, we introduce our column-approach which uses histograms distribution to measure the similarity between input puzzle pieces. Shown in Figure 5.1c, for all pixels of an edge we convert all colour values into three histograms for RGB colour space red green and blue, respectively. The histograms distribution will not be sensitive to the index ordering of pixels. EMD is a popular method for evaluate histograms distribution. Therefore, we use EMD To compute the distribution of our colour histograms. The computed EMD scores will be considered as similarity scores between sub-edges.

Once we have computed EMD scores, we use diffusion framework to find consistent scores and output confident correspondences between puzzle pieces. In Chapter 3 we have demonstrated the robustness of the diffusion framework. And Chapter 4 shows cycle-consistency helps finding consistent correspondences. We use a designated pruning procedure with two-way cycle-consistency as a constraint to prune EMD scores and output confident correspondences for solving puzzles. So that, we can address the research questions in the sections 1.3.3 and 1.3.4. To the best of our knowledge, this is the first work to use EMD as the similarity measurement in puzzle solving.

Due to the time of our implementation, in this chapter we do not have a proper assembly stage. Our diffusion pruning approach is designed as a pre-assembly technique which provides the same function as MatchLift in Chapter 4. However, in section 5.3 our experiments show that we can handle small puzzles without using assembly stage.

Section 5.2.1 shows how to use EMD to compute similarity scores between sub-edges. Section 5.2.2 shows how we compute initial correspondences, build affinity matrix, and use diffusion framework to analyse initial correspondences. Section 5.2.3 shows our pruning procedure to remove inconsistent correspondences. Section 5.3 shows our experiment results in both quantitative and qualitative evaluation. We also show our experiment results from inputs of variously shaped pieces. Section 5.4 and 5.5 will discuss our limitation with future work and summarise this chapter, respectively.

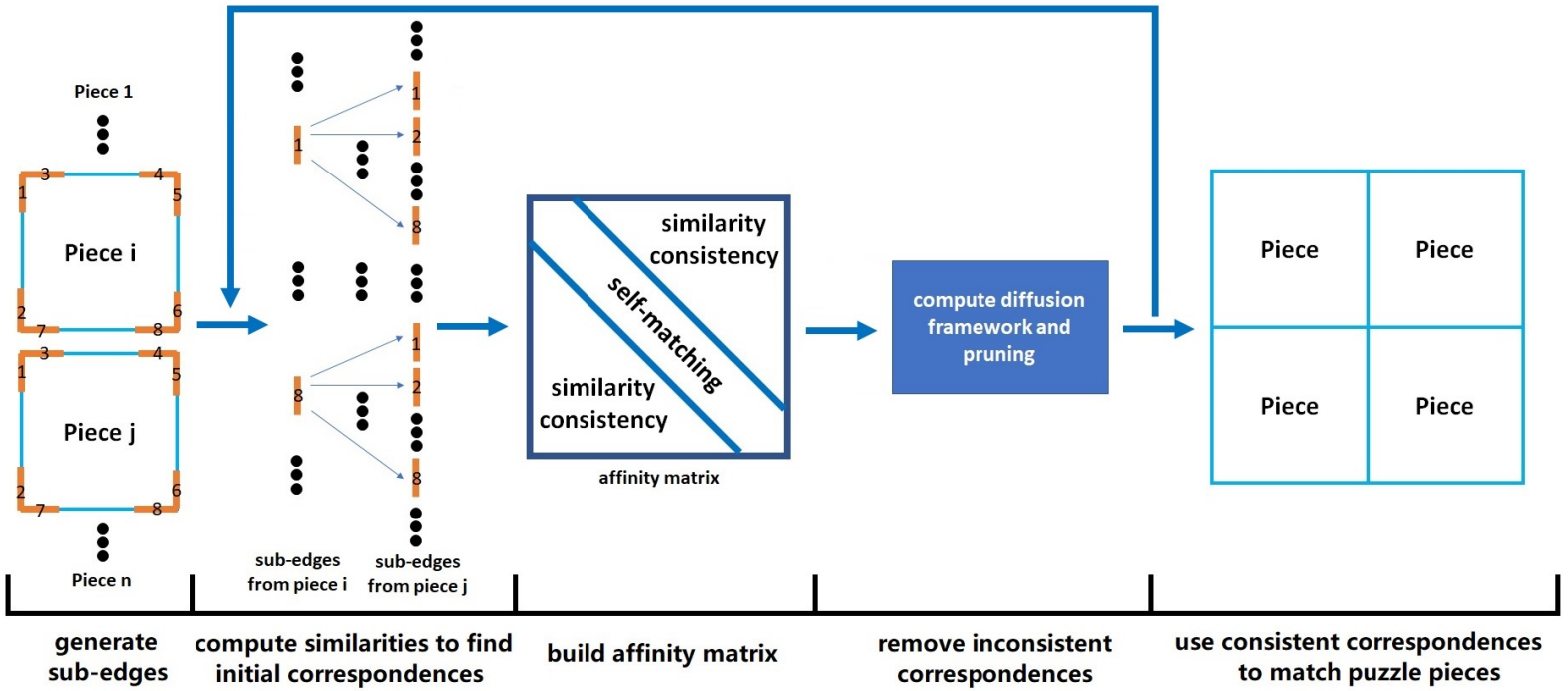


Figure 5.2: Method overview: for n in put pieces we first subdivide each whole edge into sub-edges. We compute all pair-wise similarity scores between all input pieces and build affinity matrix. We use diffusion pruning to extract consistent scores and pre-assemble puzzle pieces.

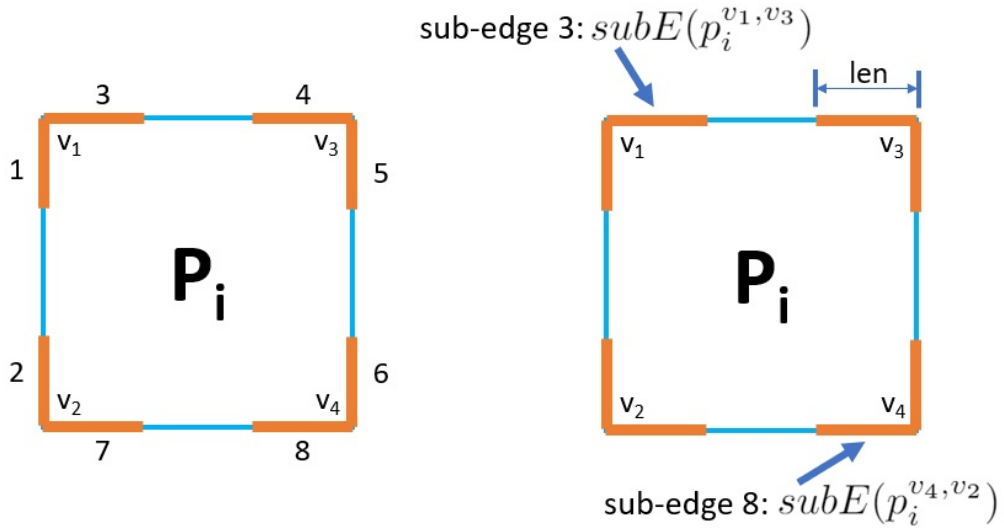
5.2 Proposed Method

Figure 5.2 shows the pipeline of our proposed method. We subdivide each edge into two-sub edges. Then we compute EMD between all possible pairs of sub-edges to generate our similarity scores. Based on scores, for each sub-edge we find the *knn*-nearest-neighbours as the initial correspondences. We use pair-wise isometrically information of initial correspondences to build affinity matrix. Then we apply diffusion framework on affinity matrix and sort all input correspondences by using diffusion confident scores. Finally, we select good correspondences by using our designated pruning procedure. We use multiple iterations of our method to generate more correct sub-edge-wise correspondences.

5.2.1 Sub-edges and EMD

For each input puzzle piece we need to subdivide each edge into two sub-edges. Let $\mathcal{P} = \{p_1, p_2, ..., p_n\}$ be the set of all input puzzle pieces. Since we are going to compute pieces with slope edges/rectangular pieces with different sizes, the edge of each piece can not be simply labelled into left, right, top, and bottom as the existing works. First, we introduce a user defined value len to indicate the number of pixels. We use len to set the length of all sub-edges (same length for all sub-edges). Then, we define vertices v_{corner} on each piece, where *corners* means the *corner*-th corner of a puzzle piece. We use the same ordering scheme to name each corner/vertex and sub-edge as stated in Chapter 4. For example, in Figure 5.3a there are four corners in the piece i, then v_2 indicates the second (left-bottom) corner of piece i.

Next, we based on index of vertices to subdivide each edge into sub-edges. We define a source vertex v_s and a target vertex v_t to form a a straight line of pixels, which is a sub-edge. A sub-edge begins at v_s and towards, however, not end at v_t (otherwise it will be a full edge between two vertices). The length of a sub-edge is len. Thus, a sub-edge is a line of pixels that locates at the boundary of a puzzle piece. For a square piece, four corners produce eight sub-edges as shown in Figure 5.3a. The first sub-edge is from v_1 to v_2, the second sub-edge is from v_2 to v_1, the third sub-edge is from v_1 to v_3, the fourth sub-edge is from v_3 to v_1, and so on. The



(a) sub-edge notations

(b) access to sub-edge

Figure 5.3: The notation of sub-edges and how to access them.

sub-edges are only about pixels and they are not directed edge. Same procedure for rectangular and triangular pieces, only there are three corners/tips and six sub-edges in triangular pieces. We define an operator $subE(p_i^{v_s,v_t})$ to return a sub-edge that between vertex v_s to vertex v_t. For example, in Figure 5.3b, in a set of square pieces $subE(p_i^{v_4,v_2})$ will return the eighth sub-edge of the i-th input puzzle piece, since the source vertex $s = 4$ and the target vertex $t = 2$. It begins at the fourth vertex and towards the second vertex, which match the eighth sub-edge.

We consider each pair of sub-edges as a correspondence $c_k \in C$. The range of k is changing, which depends on the shape of input puzzle pieces and a user defined value *knn*. The value *knn* indicates the *knn*-nearest-neighbour and we will use *knn* in initial correspondences computation (in the next section). When input puzzle has n square/rectangular pieces, then $1 \leq k \leq knn \times 8 \times n^2$. If the input puzzle contains both triangular and square pieces, the range of k will depend on the number of each type of pieces. We use EMD in [83] to compute similarity score $sim(c_k)$ as the weight of this correspondence. In [83] EMD is defined as

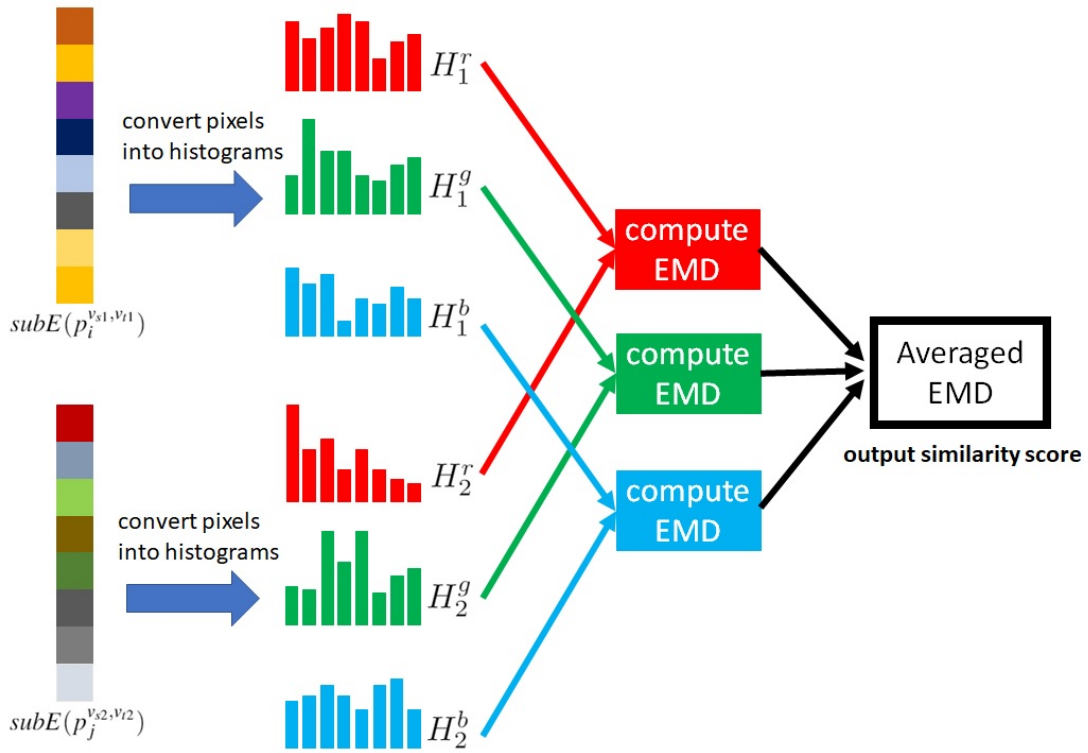


Figure 5.4: An example of EMD similarity score: for a pair of input sub-edges we convert all pixels into colour histograms. In this example user defined value $len = 8$, which means eight pixels in each sub-edge. Therefore, each histogram has eight bins.

$$
\begin{aligned}
EMD_1(H1,H2) =& (\min_{\{f_{ij}\}} \sum_{i,j} f_{ij} d_{ij}) + |\sum_i H1_i - \sum_j H2_j| \, t \max_{i,j} d_{ij} \\
& s.t \\
& f_{ij} \geq 0, \; \sum_j f_{ij} \leq H1_i, \; \sum_i f_{ij} \leq H2_j, \\
& \sum_{i,j} f_{ij} = \min(\sum_i H1_i, \sum_j H2_j)
\end{aligned}
\tag{5.1}
$$

Where $H1$ and $H2$ are input histograms. $\{f_{ij}\}$ is the flows and f_{ij} means the amount that will be transferred from i to j. d_{ij} is the ground distance between the i-th bin in $H1$ and the j-th bin in $H2$.

In our case, histograms contain colour distribution of each sub-edge and the ground distances are the absolute value of bin values' subtraction. As shown in Figure 5.4 for a pair of

sub-edges $subE(p_i^{v_{s_1},v_{t_1}})$ and $subE(p_j^{v_{s_2},v_{t_2}})$, we extract all red values in $subE(p_i^{v_{s_1},v_{t_1}})$ and form them into a histogram H_i^r. And H_i^g, H_i^b are histograms for colour green and blue, respectively. We do the same procedures on sub-edge $subE(p_j^{v_{s_2},v_{t_2}})$ to generate H_j^r, H_j^g, and H_j^b. Then, we compute three EMD values $EMD_1(H_i^r,H_j^r)$, $EMD_1(H_i^g,H_j^g)$, and $EMD_1(H_i^b,H_j^b)$ and average them as the raw similarity score $sim_{raw}(c_k)$ between sub-edge $subE(p_i^{v_{s_1},v_{t_1}})$ and $subE(p_j^{v_{s_2},v_{t_2}})$. We define $sim_{raw}(c_k) = avgEMD(subE(p_i^{v_{s_1},v_{t_1}}), subE(p_j^{v_{s_2},v_{t_2}}))$ as

$$avgEMD(subE(p_i^{v_{s_1},v_{t_1}}), subE(p_j^{v_{s_2},v_{t_2}})) = \begin{cases} \frac{EMD_1(H_i^r,H_j^r)+EMD_1(H_i^g,H_j^g)+EMD_1(H_i^b,H_j^b)}{3}, & i \neq j \\ 0, & otherwise \end{cases} \tag{5.2}$$

We further normalise $sim_{raw}(c_k)$ by divide them with the smallest similarity score $min(sim_{raw})$. We also set a user defined threshold t_1 to remove large values (larger means more dissimilar). The final EMD similarity score $sim(c_k)$ is defined as

$$sim(c_k) = \begin{cases} \frac{sim_{raw}(c_k)}{min(avgEMD)}, & \frac{sim_{raw}(c_k)}{min(avgEMD)} \geq t_1 \\ 0, & otherwise \end{cases} \tag{5.3}$$

5.2.2 Initial Correspondences and Diffusion Analysis

We base on $sim(c_k)$ of each correspondence to find initial correspondences C_{init} by using *knn*-nearest-neighbours. For each sub-edge we compute similarity to all other sub-edges. The small value of similarity score means two sub-edges are similar to each other, vice versa. We sort these scores in ascent order. We select the top *knn* scores and pass their corresponding sub-edges to C_{init} as initial correspondences. We will introduce the selection of value *knn* in section 5.2.3.

We use C_{init} to build affinity matrix $\mathcal{M}$ and apply diffusion pruning to find consistent matching between sub-edges. The size of $\mathcal{M}$ is $\mid C_{init} \mid \times \mid C_{init} \mid$. Each element in $\mathcal{M}$ indicates the pair-wise isometric information of two initial correspondences. For a pair of initial

correspondences a and b,

$$a = (subE(p_i^{v_{s_1}, v_{t_1}}), subE(p_j^{v_{s_2}, v_{t_2}})),\ a \in C_{init},\ p_i p_j \in \mathcal{P}$$

$$b = (subE(p_\alpha^{v_{s_3}, v_{t_3}}), subE(p_\beta^{v_{s_4}, v_{t_4}})),\ b \in C_{init}\ p_\alpha p_\beta \in \mathcal{P}$$

we define their pair-wise isometrically information as

$$M_{a,b} = \begin{cases} m_{a,b}, & a \neq b,\ m_{a,b} \geq t_2,\ 0 \leq t_2 \leq 1 \\ 1, & a = b \\ 0, & otherwise, \end{cases}$$

$$m_{a,b} = \begin{cases} \frac{\min(sim(a)/sim(b), sim(b)/sim(a))}{\min(sim(a), sim(b))}, & (i = \alpha\ \&\ j = \beta) \parallel (i = \beta\ \&\ j = \alpha) \\ 0, & otherwise \end{cases} \tag{5.4}$$

Where t_2 is a user defined threshold, for the most experiments in this chapter we set $t_2 = 0.7$ (see section 5.3 for details). The non-zero $m_{a,b}$ indicates the similarity consistency between two sub-edges. Finally, the diagonal elements in $\mathcal{M}$ are representing self-matching, the consistency of sub-edges are encoded in non-diagonal elements.

Once we have built affinity matrix $\mathcal{M}$, we apply the diffusion framework on it to compute diffusion confident score for all input initial correspondences C_{init}. We follow the same normalisation as stated in [3] to convert $\mathcal{M}$ into Markov probability matrix. The diffusion analysis will be based on Markov random walk to generate diffusion confident score for each correspondence in $\mathcal{M}$.

5.2.3 Pruning and Iteration

We prune inconsistent correspondences to find good matchings. First, we base on diffusion confident scores to sort all C_{init} into C_{dp} as in descent order. Since lower diffusion confident scores indicate more inconsistent correspondences, we only consider the top t_3% correspondences in C_{dp} and we remove the rest of them, where $t_3$is a user defined threshold. Our pruning

procedure ensures the two-way cycle-consistency between a pair of the whole edge, which means all sub-edges of a pair of the whole edge must form forward-backwards matching. Otherwise, we consider it is a incorrect pair of the whole edge and we do not accept them in the pruned results C_{pruned}. The detailed algorithm of multiple diffusion analysis is shown in Algorithm 2.

Algorithm 2 Pruning Algorithm

Input: C_{dp}
Output: C_{pruned}

procedure PRUNE(C_{dp})
  for each $a = (subE(P_i^{v_{s_1}, v_{t_1}}), subE(P_j^{v_{s_2}, v_{t_2}})) \in C_{dp}$
    $counter \leftarrow 0$
    $matchedVs_1 \leftarrow \emptyset$
    $matchedVs_2 \leftarrow \emptyset$
    for each $b = (subE(P_\alpha^{v_{s_3}, v_{t_3}}), subE(P_\beta^{v_{s_4}, v_{t_4}})) \in C_{dp}$
      if $(i = \alpha \; and \; j = \beta \; and \; subE(P_i^{v_{s_1}, v_{t_1}}) != subE(P_\alpha^{v_{s_3}, v_{t_3}}))$ **then**
        $counter = counter + 1$
        $matchedVs_1 \leftarrow v_{s_3}$
        $matchedVs_2 \leftarrow v_{s_4}$
      else if $(i = \beta) \; and \; j = \alpha$ **then**
        $counter1 = counter1 + 1$
      end if
    end for
    if $(counter1 = 3 \; and \; | \; matchedVs_1 | = 1 \; and \; | \; matchedVs_2 | = 1)$ **then**
      $val = max(sim(a)/sim(b) \; , \; sim(b)/sim(a))$
      if $(abs(v_{s_1} - matchedVs_1) = 1 \; and \; abs(v_{s_2} - matchedVs_2) = 1 \; and \; val \leq 10)$ **then**
        $C_{pruned} \leftarrow a$
      end if
    end if
  end for
  return C_{pruned}
end procedure

We run diffusion framework and our pruning procedure in multiple iterations to obtain more correct correspondences. We base on the pruned results C_{pruned} to recompute C_{init} for the following iteration. For all sub-edges that are existing in C_{pruned} we do not compute their initial correspondences, and we focus on unmatched sub-edges (this is similar to adaptive spectral matching in [36]). Then, we build $\mathcal{M}$, run diffusion framework, prune correspondences to finish a iteration. The number of iterations is also defined by the user, and we show the detailed iteration setting in section 5.3.

5.3 Evaluation

We test our technique in both quantitative and qualitative evaluations. In the quantitative part, we use fixed parameters setting in all experiments. We set $knn = 2, t_1 = 0.1, t_2 = 0.7, t_3 = 0.3$. The number of iterations is automatically generated. When there are no new pruned results we stop our diffusion pruning computation. We first show the reliability of our EMD-based similarity scores by comparing the precision rate from the popular similarity measurements MGC [20]. Then, we use neighbour comparison (the ratio between the number of correctly placed pieces and the number of input pieces) to test our method, [54] and MGC. For comparisons with other works we use square pieces. In the qualitative part, we use tuned parameters to obtain the best precision and recall from different input puzzles. We also use differently shaped pieces to test out method.

We use images from SSD data sets [22] and ten images from our data set in evaluation. In the quantitative part, for SSD images we slice them into 192 square pieces with the same size. For our images we slice them into 100 square pieces with the same size. In the qualitative part, we slice our images into a different number of pieces in terms of piece shapes. For all experiments in this section we use known orientation (the Type I Puzzle in [20]).

Images	MGC	EMD
1	72	76
2	19	80
3	42	79
4	55	84
5	61	81
6	68	85
7	77	92
8	74	89
9	57	94
10	63	86
avg	59	85

Table 5.1: Similarity score stability comparison.

5.3.1 Quantitative

We show reliability of EMD in Table 5.1. We use five SSD images and five of our images to compute similarity scores by using EMD and MGC. Each score means the similarity between two edges of two pieces. Thus, for each edge we search the smallest score (smaller

means more similar) and locate the two piece and their placement. We compare the placement with ground-truth to check the correctness of this placement. We use this way to evaluate the precision rate of EMD and MGC. Our experiments show that EMD has higher precision rate than MGC and it is more reliable than MGC. Another thing to be noticed is that EMD is more stable than MGC. In our pruning implementation we consider two similarity scores are not consistent if they are ten times different. For MGC scores it is normal to have two consistent scores with a huge difference (see Figure 4.1b).

Table 5.2 shows the neighbour comparison between MGC [20], our method by using MGC scores as similarity measurement, and our method by using EMD as similarity measurement. Overall, our method with EMD performs the best result. As mentioned in the section4.5, MGC is sensitive to low-distinctive regions. It leads to incorrect pieces placement. In our case, the local-consistency from diffusion framework and two-way cycle-consistency from our pruning procedure ensured our method is not sensitive to low-distinctive regions. Although our placement correctness slightly dropped when we use MGC instead of EMD in our method, we still have better results than the other two methods.

5.3.2 Qualitative

In this section, we use variously shaped pieces to test our method. We use customised parameters in the three different experiments. The first experiment uses 25 square pieces with the same size as the input, parameters setting is $knn = 3, t1 = 0.1, t2 = 0.7, t3 = 0.3$. The Second experiment uses rectangular and square pieces as input. The parameters setting is the same as the first experiment. Experiment three uses triangular and square pieces as input and we manually customise all parameters.

Figure 5.5 shows the first experiment. Our method can solve square puzzles without using assembly techniques. The correspondences from diffusion pruning cover all input pieces with 100% correctness. Even though some edges are not matched by our technique, it does not affect the placement of pieces. However, for larger puzzles (hundreds of pieces or even more), our

Images	MGC	Ours with MGC	Ours with EMD
1	23	82	82
2	27	88	87
3	94	90	94
4	56	85	88
5	82	87	85
6	70	91	90
7	93	87	87
8	39	89	90
9	60	91	90
10	69	93	92
11	87	95	89
12	98	80	82
13	44	59	85
14	88	85	84
15	83	90	88
16	49	82	87
17	77	86	88
18	32	98	100
19	85	92	98
20	89	96	94
21	20	100	90
22	22	99	100
23	54	51	100
24	69	93	100
25	87	95	100
26	77	60	87
27	98	92	85
28	83	90	100
29	95	94	71
30	91	88	86
avg	68	87	90

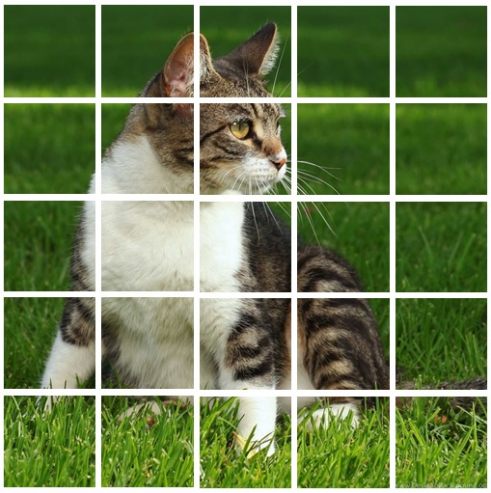
Table 5.2: Best Neighbour comparison between MGC [20], our method with MGC as similarity scores, and our method with EMD as similarity scores. Image 1 to 20 are SSD images [22] and image 21 to 30 are our images.

technique will generate incorrect results and we need a proper assembly technique for solving.

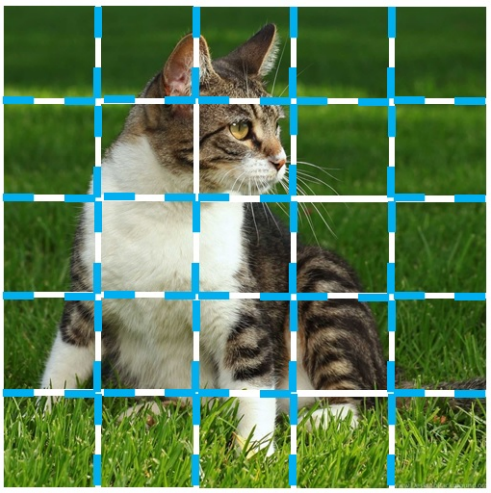
Figures 5.6 and 5.9 show the second experiment. We first generate square pieces in each image, then randomly merge the adjacent squares to form a rectangular piece. Our pruning procedure ensures the fully sub-edge-wise matching between a pair of the whole edge, which means the longer edge of a rectangular piece will not be matched with a square piece. Thus, a rectangular piece can only be matched with a square piece in horizontal. When two rectangular pieces are perfectly aligned in vertical, the two longer edges can be matched. Our matching results ensure the good placement of input pieces without using assembly techniques.

The third experiment is shown in Figure 5.8a. We use another slicing procedure to generate triangular and square pieces from one image. We set 300 pixels as the length of each sub-edge and there are 350 pixels on each whole edge. We set $knn = 3, t1 = 0.05, t2 = 0.8, t3 = 0.5$ as the parameters setting. From the visualised correspondences we can claim that it is possible to place input pieces with 100% correctness.

We show a failure case by using 56 triangular pieces and 20 square pieces. The sliced image is shown in Figure 5.9a. The length of each whole edge is 120 pixels. We set 70 pixels as the length of the sub-edges. The first iteration outputs acceptable correspondences. However, incorrect matchings are generated from the second iteration. By inspecting EMD scores we found that only 52% scores are reliable. It is caused by our slicer can not generate a consistent number of pixels on a sloping edge, which means incorrect EMD scores have been generated from un-consecutive/un-adjacent pixels. When the sub-edges do not have enough pixels EMD scores will become unreliable. Since we do not have a proper assembly stage, we can not handle these incorrectly computed correspondences. Based on our observation, after the third iteration the correct correspondences have covered most pieces. in the future we can use a designated assembly technique with global constraints to handle incorrect similarity scores and correspondences.



(a) Input Pieces

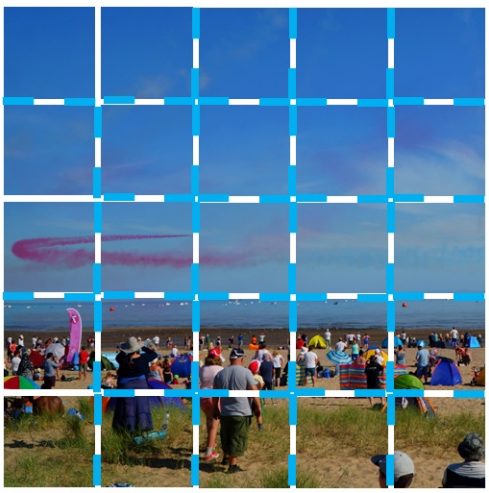


(b) Matching results after five iterations

(c) Input Pieces

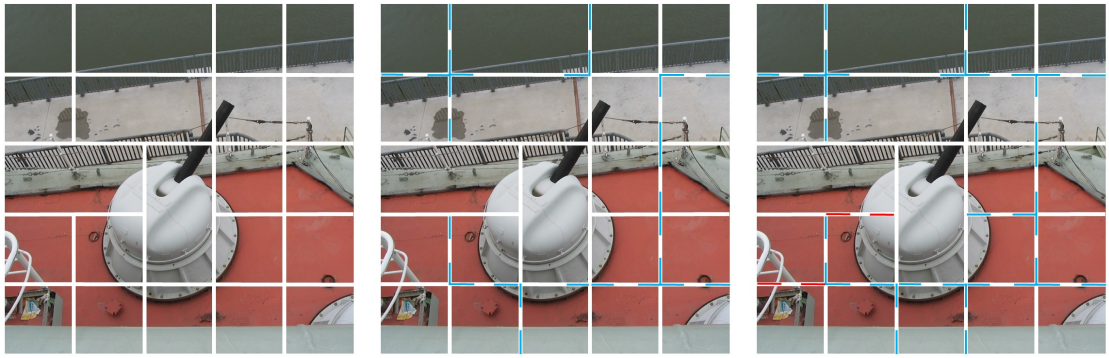
(d) Matching Results after four iterations

(e) Input Pieces



(f) Matching Results after five iterations

Figure 5.5: For a small puzzle, without a assembly stage our method can find 100% correct placement of all input pieces. Although some edges have not been matched, it is possible to correctly place those pieces by using matchings from their neighbours.

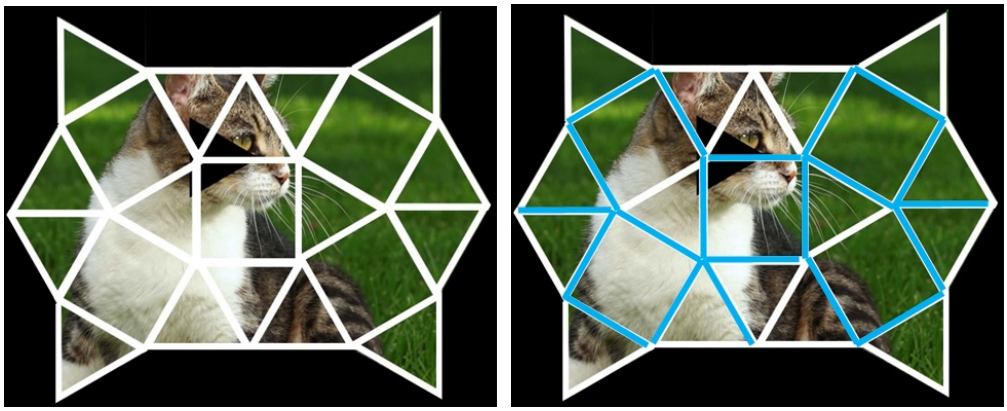


(a) Input Pieces (b) Matchings from the first iteration (c) Matchings from the second iteration

Figure 5.6: Matching results of rectangular pieces with different sizes. After two iterations 19 pieces have been correctly placed out of 21 input pieces.

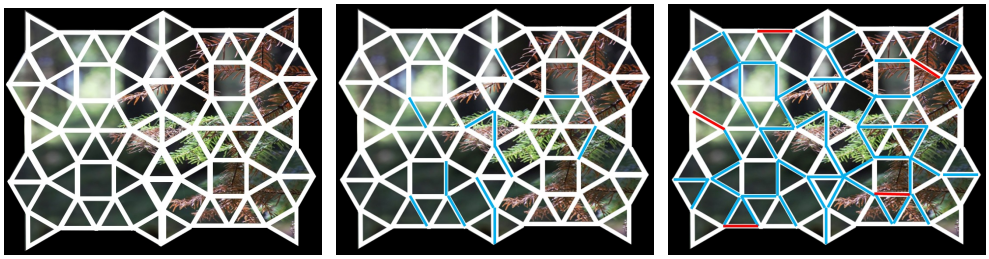
(a) Input Pieces (b) Matchings from the first iteration (c) Matchings from the fourth iteration

Figure 5.7: After four iterations all input pieces have been correctly placed.



(a) Input Pieces (b) Matchings after the third iteration

Figure 5.8: After the third iteration all input pieces can be correctly placed by using pruning correspondences.



(a) Input Pieces (b) Matchings from the first iteration (c) Matchings from the second iteration

Figure 5.9: After the first iteration incorrect correspondences are found, shown in the red.

5.4 Limitations and future work

Computation time is the first limitation of our method. For each puzzle we have loops to compute and there are multiple diffusion analysis. The repeating usage of diffusion framework caused a long computation time. Currently, our implementation is using Matlab. There are built-in functions and toolbox functions in Matlab that can be used in CUDA programming, and these functions have covered our implementation. To address this limitation, in the future we can use parallel computing in GPU to reduce the computation time.

Assembly is the second limitation of our method. Currently, we are using MST to assemble un-matched puzzle pieces. As stated in Chapter 4 MST may generate incorrect results when the input similarity scores are unreliable. in the future, we hope to develop a new assembling technique that is based on the matching results in each loop to adjust assembled pieces automatically.

5.5 Conclusion

In this chapter we have introduced a puzzle solving technique based on EMD and diffusion analysis. We have demonstrated that the usage of EMD performs good results in similarity measurement. Our diffusion pruning technique shows good two-way cycle-consistency between pieces. We use multiple diffusion computation to obtain good results. Our experiments show we can handle differently shaped puzzle pieces with good precision rate. Our next step is

to develop a proper assembly technique to achieve better assembled results. Moreover, we can use parallel computing in GPU to reduce the time consumption, so that we can handle larger puzzles.

Chapter 6

future work and Conclusion

Contents

6.1 future work

In this chapter we discuss the long term future work and we have demonstrated the the short term future works in section 3.7, 4.7, and 5.4. In the 3D domain we want to use cycle-consistency as the constraint to solve the matching problem with time steps, such as video frame processing. In the 2D domain we want to use pixel-level matching to solve puzzle solving.

6.1.1 Cycle-consistency and Video Frames

Video frames processing is a classic problem in computer science/vision with vast applications. Track a constant region/object during the framing is an essential function, and this can be modelled as finding cycle-consistent correspondences between different frames. Video frame

interpolation is also an essential function for high frame rate per second. Gamer, game studios, smartphone/tablet market, and hardware manufacturers are focused on increase frame rate per second for the better user experience. The interpolation can also be considered as a consistent cycle-wise correspondences problem.

Although cycle-consistency has been studied in 3D matching or 2D image processing, it has been rarely applied in video frame processing. [112] introduces a novel way to solving the tracking problem by using multi-way cycle-consistency as a constraint in consecutive frames (time steps). The multi-way cycle-consistency supports the tracking of globally consistent objects. [113] demonstrates the first work by using cycle-consistency as the constraint in unsupervised video interpolation. The involved two-way cycle-consistency ensures the system to interpolate the correct pixels that match the adjacent frames (the consecutive input frames). In the future, we hope to use designated cycle-consistency constraints to obtain longer cycles than existing works in video frame processing problem.

6.1.2 Pixel-level Matching Between Puzzle Pieces

Using pixel-level correspondences computing in puzzle solving may produce better results than existing works. Existing puzzle solving techniques are using pair-wise pixels to generate similarity scores. It requires a post-processing stage to combine/merge all pixel-level similarities into a single value. There is a considerable variation of similarity scores, and some correct pairs of pieces have been assigned with large scores (which means more dissimilar, see Figure 4.1b). By computing matching on pixel-level of each puzzle piece, we may generate more accurate correspondence and improve assembly results to solve puzzles. However, the amount of computing will be significantly increased. In the future, we want to find a high-efficiency correspondences computation to solve puzzles.

6.2 Conclusion

In this thesis, we have demonstrated using consistent correspondences to solve matching problems in both 3D and 2D domain. We first introduced the existing works of computing correspondences in both 3D and 2D domain. Then we demonstrated our works in both 3D/2D.

In the 3D, we have introduced a novel shape matching method to handle over/imperfect-segmentation. We use segments merging approach to generate our novel multi-layer graphs with hierarchical understanding. Then we use our diffusion pruning based matching technique to find consistent correspondences between shape segments. Our multi-layer graphs preserve geometric, topological, and hierarchical information of input shape segments, and we can find meaningful merged-to-merged segment-wise correspondences to achieve a higher level understanding of input shape segments.

In the 2D, we demonstrated our sub-edges based puzzle solving techniques to handle low-distinctive pieces and variously shaped pieces. Based on the sub-edges we can formulate the traditional edge-wise puzzle solving into our corner-wise approach, which performs better results than the traditional techniques. Our technique also shows that by combining the diffusion framework and properly designed pruning procedure, we can find high-quality matching results of puzzle solving problem. Still, we show MatchLift framework produces good results in puzzle solving with low-distinctive pieces. In the future, we will refine and optimise our techniques and keep researching the matching problem in 3D/2D.

List of Figures

List of Tables

Bibliography

[1] O. van Kaick, H. Zhang, G. Hamarneh, and D. Cohen-Or, "A survey on shape correspondence," vol. 30, pp. 1681–1707, 09 2011.

[2] Q.-X. Huang, B. Adams, M. Wicke, and L. J. Guibas, "Non-rigid registration under isometric deformations," in *Proceedings of the Symposium on Geometry Processing*, SGP '08, pp. 1449–1457, Eurographics Association, 2008.

[3] G. K. L. Tam, R. R. Martin, P. L. Rosin, and Y.-K. Lai, "Diffusion pruning for rapidly and robustly selecting global correspondences using local isometry," *ACM Trans. Graph.*, vol. 33, pp. 4:1–4:17, Feb. 2014.

[4] P. J. Besl and N. D. McKay, "A method for registration of 3-d shapes," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 14, pp. 239–256, Feb 1992.

[5] Y. Chen and G. Medioni, "Object modelling by registration of multiple range images," *Image and Vision Computing*, vol. 10, no. 3, pp. 145 – 155, 1992. Range Image Understanding.

[6] S. Rusinkiewicz and M. Levoy, "Efficient variants of the icp algorithm," in *Proceedings Third International Conference on 3-D Digital Imaging and Modeling*, pp. 145–152, May 2001.

[7] S. Bouaziz, A. Tagliasacchi, and M. Pauly, "Sparse iterative closest point," *Computer Graphics Forum (Symposium on Geometry Processing)*, vol. 32, no. 5, pp. 1–11, 2013.

[8] S. Bouaziz and M. Pauly, "Dynamic 2d/3d registration for the kinect," 07 2013.

[9] R. Dyke, Y.-K. Lai, P. Rosin, and G. Tam, "Non-rigid registration under anisotropic deformations," *Computer Aided Geometric Design*, vol. 71, 04 2019.

[10] M. Leordeanu and M. Hebert, "A spectral technique for correspondence problems using pairwise constraints," in *Proceedings of the Tenth IEEE International Conference on Computer Vision - Volume 2*, ICCV '05, (Washington, DC, USA), pp. 1482–1489, IEEE Computer Society, 2005.

[11] Varunjain, Haozhang, and O. van Kaick, "Non-rigid spectral correspondence of triangle meshes," *International Journal of Shape Modeling*, vol. 13, 11 2011.

[12] G. K. Tam, R. R. Martin, P. L. Rosin, and Y.-K. Lai, "An efficient approach to correspondences between multiple non-rigid parts," *Computer Graphics Forum*, vol. 33, no. 5, pp. 137–146, 2014.

[13] Q.-X. Huang, G.-X. Zhang, L. Gao, S.-M. Hu, A. Butscher, and L. Guibas, "An optimization approach for extracting and encoding consistent maps in a shape collection," *ACM Trans. Graph.*, vol. 31, pp. 167:1–167:11, Nov. 2012.

[14] T. Zhou, Y. J. Lee, S. X. Yu, and A. Efros, "Flowweb: Joint image set alignment by weaving consistent, pixel-wise correspondences," pp. 1191–1200, 06 2015.

[15] Q.-X. Huang and L. Guibas, "Consistent shape maps via semidefinite programming," *Computer Graphics Forum*, vol. 32, no. 5, pp. 177–186, 2013.

[16] Y. Chen, L. Guibas, and Q. Huang, "Near-optimal joint object matching via convex relaxation," in *Proceedings of the 31st International Conference on Machine Learning* (E. P. Xing and T. Jebara, eds.), vol. 32 of *Proceedings of Machine Learning Research*, (Bejing, China), pp. 100–108, PMLR, 22–24 Jun 2014.

[17] J. Maintz and M. A. Viergever, "A survey of medical image registration," *Medical Image Analysis*, vol. 2, no. 1, pp. 1 – 36, 1998.

[18] M. A. Viergever, J. A. Maintz, S. Klein, K. Murphy, M. Staring, and J. P. Pluim, "A survey of medical image registration – under review," *Medical Image Analysis*, vol. 33, pp. 140 – 144, 2016. 20th anniversary of the Medical Image Analysis journal (MedIA).

[19] C. F. Olson, A. I. Ansar, and C. W. Padgett, "Robust registration of aerial image sequences," in *Advances in Visual Computing* (G. Bebis, R. Boyle, B. Parvin, D. Koracin, Y. Kuno, J. Wang, R. Pajarola, P. Lindstrom, A. Hinkenjann, M. L. Encarnação, C. T. Silva, and D. Coming, eds.), (Berlin, Heidelberg), pp. 325–334, Springer Berlin Heidelberg, 2009.

[20] A. Gallagher, "Jigsaw puzzles with pieces of unknown orientation," in *Proc. CVPR*, 2012.

[21] H. Freeman and L. Garder, "Apictorial jigsaw puzzles: The computer solution of a problem in pattern recognition," *IEEE Transactions on Electronic Computers*, vol. EC-13, pp. 118–127, April 1964.

[22] T. S. Cho, S. Avidan, and W. T. Freeman, "A probabilistic image jigsaw puzzle solver," in *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 183–190, June 2010.

[23] D. Toth, M. Panayiotou, A. Brost, J. M. Behar, C. A. Rinaldi, K. S. Rhode, and P. Mountney, "3d/2d registration with superabundant vessel reconstruction for cardiac resynchronization therapy," *Medical image analysis*, vol. 42, p. 160—172, December 2017.

[24] J. Krüger, S. Schultz, H. Handels, and J. Ehrhardt, "Registration with probabilistic correspondences — accurate and robust registration for pathological and inhomogeneous medical data," *Computer Vision and Image Understanding*, vol. 190, p. 102839, 2020.

[25] V. Corona, A. I. Aviles-Rivero, N. Debroux, M. Graves, C. Le Guyader, C.-B. Schönlieb, and G. Williams, "Multi-tasking to correct: Motion-compensated mri via joint reconstruction and registration," in *Scale Space and Variational Methods in Computer Vision* (J. Lellmann, M. Burger, and J. Modersitzki, eds.), (Cham), pp. 263–274, Springer International Publishing, 2019.

[26] B. Amos, B. Ludwiczuk, and M. Satyanarayanan, "Openface: A general-purpose face recognition library with mobile applications," tech. rep., CMU-CS-16-118, CMU School of Computer Science, 2016.

[27] P. Barra, C. Bisogni, M. Nappi, and S. Ricciardi, "Fast quadtree-based pose estimation for security applications using face biometrics," in *Network and System Security* (M. H. Au, S. M. Yiu, J. Li, X. Luo, C. Wang, A. Castiglione, and K. Kluczniak, eds.), (Cham), pp. 160–173, Springer International Publishing, 2018.

[28] R. Hu, H. Xu, M. Rohrbach, J. Feng, K. Saenko, and T. Darrell, "Natural language object retrieval," *CoRR*, vol. abs/1511.04164, 2015.

[29] A. Grabner, P. M. Roth, and V. Lepetit, "3d pose estimation and 3d model retrieval for objects in the wild," *CoRR*, vol. abs/1803.11493, 2018.

[30] M. Ioannides and E. Quak, *3D Research Challenges in Cultural Heritage: A Roadmap in Digital Heritage Preservation*. 01 2014.

[31] J. Wang, D. Gu, Z. Yu, C. Tan, and L. Zhou, "A framework for 3d model reconstruction in reverse engineering," *Computers Industrial Engineering*, vol. 63, no. 4, pp. 1189 – 1200, 2012.

[32] A. V. Dongaonkar and R. M. Metkar, *Reconstruction of Damaged Parts by Integration Reverse Engineering (RE) and Rapid Prototyping (RP)*, pp. 159–171. Singapore: Springer Singapore, 2019.

[33] N. Derech, A. Tal, and I. Shimshoni, "Solving archaeological puzzles," *CoRR*, vol. abs/1812.10553, 2018.

[34] D. Hoffman and W. Richards, "Parts of recognition," *Cognition*, vol. 18, no. 1, pp. 65 – 96, 1984.

[35] Y. Kleiman and M. Ovsjanikov, "Robust structure-based shape correspondence," *CoRR*, vol. abs/1710.05592, 2017.

[36] Y. Kleiman, O. van Kaick, O. Sorkine-Hornung, and D. Cohen-Or, "Shed: Shape edit distance for fine-grained shape similarity," *ACM Trans. Graph.*, vol. 34, Oct. 2015.

[37] C. Zhu, R. Yi, W. Lira, I. Alhashim, K. Xu, and H. Zhang, "Deformation-driven shape correspondence via shape recognition," *ACM Trans. Graph.*, vol. 36, pp. 51:1–51:12, July 2017.

[38] I. Alhashim, K. Xu, Y. Zhuang, J. Cao, P. Simari, and H. Zhang, "Deformation-driven topology-varying 3d shape correspondence," *ACM Trans. Graph.*, vol. 34, pp. 236:1–236:13, Oct. 2015.

[39] A. Itskovich and A. Tal, "Surface partial matching and application to archaeology," *Computers Graphics*, vol. 35, no. 2, pp. 334 – 341, 2011. Virtual Reality in Brazil Visual Computing in Biology and Medicine Semantic 3D media and content Cultural Heritage.

[40] Q. Huang, V. Koltun, and L. Guibas, "Joint shape segmentation with linear programming," *ACM Trans. Graph.*, vol. 30, pp. 125:1–125:12, Dec. 2011.

[41] N. Fish, O. van Kaick, A. Bermano, and D. Cohen-Or, "Structure-oriented networks of shape collections," *ACM Trans. Graph.*, vol. 35, Nov. 2016.

[42] H. Freeman and L. Garder, "Apictorial jigsaw puzzles: The computer solution of a problem in pattern recognition," *IEEE Transactions on Electronic Computers*, vol. EC-13, pp. 118–127, April 1964.

[43] H. Wolfson, E. Schonberg, A. Kalvin, and Y. Lamdan, "Solving jigsaw puzzles by computer," *Annals of Operations Research*, vol. 12, pp. 51–64, Dec 1988.

[44] R. Webster, P. Lafollette, and R. Stafford, "Isthmus critical points for solving jigsaw puzzles in computer vision," *Systems, Man and Cybernetics, IEEE Transactions on*, vol. 21, pp. 1271 – 1278, 10 1991.

[45] D. Kosiba, P. Devaux, S. Balasubramanian, T. Gandhi, and K. Kasturi, "An automatic jigsaw puzzle solver," vol. 1, pp. 616 – 618 vol.1, 11 1994.

[46] Min Gyo Chung, M. M. Fleck, and D. A. Forsyth, "Jigsaw puzzle solver using shape and color," in *ICSP '98. 1998 Fourth International Conference on Signal Processing (Cat. No.98TH8344)*, vol. 2, pp. 877–880 vol.2, Oct 1998.

[47] Weixin Kong and B. B. Kimia, "On solving 2d and 3d puzzles using curve matching," in *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001*, vol. 2, pp. II–II, Dec 2001.

[48] F.-H. Yao and G.-F. Shao, "A shape and image merging technique to solve jigsaw puzzles," *Pattern Recognition Letters*, vol. 24, no. 12, pp. 1819 – 1835, 2003.

[49] M. Makridis and N. Papamarkos, "A new technique for solving a jigsaw puzzle," pp. 2001–2004, 01 2006.

[50] M. Sagiroglu and A. Ercil, "A texture based matching approach for automated assembly of puzzles," in *2006 18th International Conference on Pattern Recognition*, vol. 3, (Los Alamitos, CA, USA), pp. 1036–1041, IEEE Computer Society, aug 2006.

[51] T. R. Nielsen, P. Drewsen, and K. Hansen, "Solving jigsaw puzzles using image features," *Pattern Recognition Letters*, vol. 29, no. 14, pp. 1924 – 1933, 2008.

[52] A. Naif, "Solving square jigsaw puzzles using dynamic programming and the hungarian procedure," *American Journal of Applied Sciences*, vol. 6, 11 2009.

[53] X. Yang, N. Adluru, and L. J. Latecki, "Particle filter with state permutations for solving image jigsaw puzzles," in *CVPR 2011*, pp. 2873–2880, June 2011.

[54] D. Pomeranz, M. Shemesh, and O. Ben-Shahar, "A fully automated greedy square jigsaw puzzle solver," in *CVPR 2011*, pp. 9–16, June 2011.

[55] F. Andaló, G. Taubin, and S. Goldenstein, "Solving image puzzles with a simple quadratic programming formulation," pp. 63–70, 08 2012.

[56] Y. Zheng, D. Cohen-Or, and N. J. Mitra, "Smart variations: Functional substructures for part compatibility," *Computer Graphics Forum (Eurographics)*, vol. 32, pp. 195–204, 2013.

[57] D. Sholomon, O. David, and N. S. Netanyahu, "A genetic algorithm-based solver for very large jigsaw puzzles," in *2013 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1767–1774, June 2013.

[58] K. Son, J. Hays, and D. B. Cooper, "Solving square jigsaw puzzles with loop constraints," in *Computer Vision – ECCV 2014* (D. Fleet, T. Pajdla, B. Schiele, and T. Tuytelaars, eds.), (Cham), pp. 32–46, Springer International Publishing, 2014.

[59] G. Paikin and A. Tal, "Solving multiple square jigsaw puzzles with missing pieces," in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 4832–4839, June 2015.

[60] F. Richter, C. Eggert, and R. Lienhart, "Fisher vector encoding of micro color features for (real world) jigsaw puzzles," in *2015 13th International Conference on Document Analysis and Recognition (ICDAR)*, pp. 521–525, Aug 2015.

[61] R. Yu, C. Russell, and L. Agapito, "Solving jigsaw puzzles with linear programming," *CoRR*, vol. abs/1511.04472, 2016.

[62] D. , O. E. David, and N. S. Netanyahu, "Dnn-buddies: A deep neural network-based estimation metric for the jigsaw puzzle problem," in *Artificial Neural Networks and Machine Learning – ICANN 2016* (A. E. Villa, P. Masulli, and A. J. Pons Rivero, eds.), (Cham), pp. 170–178, Springer International Publishing, 2016.

[63] F. A. Andaló, G. Carneiro, G. Taubin, S. Goldenstein, and L. Velho, "Automatic reconstruction of ancient portuguese tile panels," *IEEE Computer Graphics and Applications*, 07 2016.

[64] K. Son, J. Hays, and D. B. Cooper, "Solving square jigsaw puzzle by hierarchical loop constraints," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pp. 1–1, 2018.

[65] S. Gur and O. Ben-Shahar, "From square pieces to brick walls: The next challenge in solving jigsaw puzzles," in *2017 IEEE International Conference on Computer Vision (ICCV)*, pp. 4049–4057, Oct 2017.

[66] M. Ovsjanikov, M. Ben-Chen, J. Solomon, A. Butscher, and L. Guibas, "Functional maps: a flexible representation of maps between shapes," vol. 31, pp. 1–11, 07 2012.

[67] Q. Huang, F. Wang, and L. Guibas, "Functional map networks for analyzing and exploring large shape collections," *ACM Trans. Graph.*, vol. 33, pp. 36:1–36:11, July 2014.

[68] M. Ovsjanikov, M. Ben-Chen, F. Chazal, and L. Guibas, "Analysis and visualization of maps between shapes," *Comput. Graph. Forum*, vol. 32, pp. 135–145, Sept. 2013.

[69] M. Ovsjanikov, Q. Mérigot, V. Pătrăucean, and L. Guibas, "Shape matching via quotient spaces," in *Proceedings of the Eleventh Eurographics/ACMSIGGRAPH Symposium on Geometry Processing*, SGP '13, (Aire-la-Ville, Switzerland, Switzerland), pp. 1–11, Eurographics Association, 2013.

[70] F. Wang, Q. Huang, and L. J. Guibas, "Image co-segmentation via consistent functional maps," in *Proceedings of the 2013 IEEE International Conference on Computer Vision*, ICCV '13, (Washington, DC, USA), pp. 849–856, IEEE Computer Society, 2013.

[71] G. K. L. Tam, Z. Cheng, Y. Lai, F. C. Langbein, Y. Liu, D. Marshall, R. R. Martin, X. Sun, and P. L. Rosin, "Registration of 3D point clouds and meshes: A survey from rigid to nonrigid," *IEEE Transactions on Visualization and Computer Graphics*, vol. 19, no. 7, pp. 1199–1217, 2013.

[72] D.-Y. Chen, X.-P. Tian, Y.-T. Shen, and M. Ouhyoung, "On Visual Similarity Based 3D Model Retrieval," *Computer Graphics Forum*, 2003.

[73] M. Ankerst, G. Kastenmüller, H.-P. Kriegel, and T. Seidl, "3D shape histograms for similarity search and classification in spatial databases," in *Proceedings of the 6th International Symposium on Advances in Spatial Databases*, SSD '99, pp. 207–226, Springer-Verlag, 1999.

[74] R. Osada, T. Funkhouser, B. Chazelle, and D. Dobkin, "Shape distributions," *ACM Trans. Graph.*, vol. 21, pp. 807–832, Oct. 2002.

[75] R. Blomley, M. Weinmann, J. Leitloff, and B. Jutzi, "Shape distribution features for point cloud analysis – a geometric histogram approach on multiple scales," *ISPRS annals*, vol. II-3, p. 9–16, 2014. ISPRS Technical Commission III Symposium, Zürich, CH, September 5-7, 2014.

[76] J. Sun, M. Ovsjanikov, and L. Guibas, "A concise and provably informative multi-scale signature based on heat diffusion," in *Proceedings of the Symposium on Geometry Processing*, SGP '09, pp. 1383–1392, Eurographics Association, 2009.

[77] T. Dey, K. Li, C. Luo, P. Ranjan, I. Safa, and Y. Wang, "Persistent heat signature for pose-oblivious matching of incomplete models," in *Computer Graphics Forum*, vol. 29, pp. 1545 – 1554, 07 2010.

[78] A. B. Hamza and H. Krim, "Geodesic object representation and recognition," in *Discrete Geometry for Computer Imagery* (I. Nyström, G. Sanniti di Baja, and S. Svensson, eds.), (Berlin, Heidelberg), pp. 378–387, Springer Berlin Heidelberg, 2003.

[79] H. Ling and D. W. Jacobs, "Using the inner-distance for classification of articulated shapes," in *In Proc. CVPR*, pp. 719–726, 2005.

[80] A. Bronstein, M. Bronstein, A. Bruckstein, and R. Kimmel, "Matching two-dimensional articulated shapes using generalized multidimensional scaling," vol. 4069, pp. 48–57, 07 2006.

[81] "Reprint of: Mahalanobis, p.c. (1936) "on the generalised distance in statistics."," *Sankhya A*, vol. 80, pp. 1–7, Dec 2018.

[82] O. Pele and M. Werman, "A linear time histogram metric for improved sift matching," in *Computer Vision–ECCV 2008*, pp. 495–508, Springer, October 2008.

[83] O. Pele and M. Werman, "Fast and robust earth mover's distances," in *2009 IEEE 12th International Conference on Computer Vision*, pp. 460–467, IEEE, September 2009.

[84] V. G. Kim, Y. Lipman, and T. Funkhouser, "Blended intrinsic maps," *ACM Trans. Graph.*, vol. 30, pp. 79:1–79:12, July 2011.

[85] R. W. Sumner, J. Schmid, and M. Pauly, "Embedded deformation for shape manipulation," *ACM Trans. Graph.*, vol. 26, p. 80, 2007.

[86] J. a. Maciel and J. a. P. Costeira, "A global solution to sparse correspondence problems," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 25, pp. 187–199, Feb. 2003.

[87] A. C. Berg, T. L. Berg, and J. Malik, "Shape matching and object recognition using low distortion correspondences," in *Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05) - Volume 1 - Volume 01*, CVPR '05, pp. 26–33, IEEE Computer Society, 2005.

[88] N. Gelfand, N. J. Mitra, L. J. Guibas, and H. Pottmann, "Robust global registration," in *Proceedings of the Third Eurographics Symposium on Geometry Processing*, SGP '05, Eurographics Association, 2005.

[89] H. Zhang, A. Sheffer, D. Cohen-or, Q. Zhou, O. V. Kaick, and A. Tagliasacchi, "Deformation-driven shape correspondence," 2008.

[90] O. van Kaick, H. Zhang, and G. Hamarneh, "Bilateral maps for partial matching," *Comput. Graph. Forum*, vol. 32, pp. 189–200, Sept. 2013.

[91] S. Chaudhuri, E. Kalogerakis, L. Guibas, and V. Koltun, "Probabilistic reasoning for assembly-based 3D modeling," *ACM Trans. Graph.*, vol. 30, pp. 35:1–35:10, July 2011.

[92] E. Kalogerakis, S. Chaudhuri, D. Koller, and V. Koltun, "A probabilistic model for component-based shape synthesis," *ACM Trans. Graph.*, vol. 31, pp. 55:1–55:11, July 2012.

[93] L. Shapira, S. Shalom, A. Shamir, D. Cohen-Or, and H. Zhang, "Contextual part analogies in 3D objects," *International Journal of Computer Vision*, vol. 89, pp. 309–326, Sep 2010.

[94] Y. Wang, K. Xu, J. Li, H. Zhang, A. Shamir, L. Liu, Z. Cheng, and Y. Xiong, "Symmetry hierarchy of man-made objects," *Computer Graphics Forum Eurographics 2011*, vol. 30, no. 2, pp. 287–296, 2011.

[95] H. Laga, M. Mortara, and M. Spagnuolo, "Geometry and context for semantic correspondences and functionality recognition in man-made 3D shapes," *ACM Trans. Graph.*, vol. 32, pp. 150:1–150:16, Oct. 2013.

[96] M. Pechuk, O. Soldea, and E. Rivlin, "Learning function-based object classification from 3D imagery," *Comput. Vis. Image Underst.*, vol. 110, pp. 173–191, May 2008.

[97] C. Zach, M. Klopschitz, and M. Pollefeys, "Disambiguating visual relations using loop constraints," pp. 1426–1433, 06 2010.

[98] R. Roberts, S. N. Sinha, R. Szeliski, and D. Steedly, "Structure from motion for scenes with large duplicate structures," in *CVPR 2011*, pp. 3137–3144, June 2011.

[99] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros, "Unpaired image-to-image translation using cycle-consistent adversarial networks," *2017 IEEE International Conference on Computer Vision (ICCV)*, pp. 2242–2251, 2017.

[100] Q. Wang, X. Zhou, and K. Daniilidis, "Multi-image semantic matching by mining consistent features," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 685–694, 2018.

[101] D. Mondal, Y. Wang, and S. Durocher, "Robust solvers for square jigsaw puzzles," in *2013 International Conference on Computer and Robot Vision*, pp. 249–256, May 2013.

[102] C. Zanoci, "Making puzzles less puzzling : An automatic jigsaw puzzle solver," 2016.

[103] K. Son, D. Moreno, J. Hays, and D. B. Cooper, "Solving small-piece jigsaw puzzles by growing consensus," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1193–1201, June 2016.

[104] D. Goldberg, C. Malon, and M. Bern, "A global approach to automatic solution of jigsaw puzzles," *Computational Geometry*, vol. 28, no. 2, pp. 165 – 174, 2004. Special Issue on the 18th Annual Symposium on Computational Geometry - SoCG2002.

[105] O. van Kaick, K. Xu, H. Zhang, Y. Wang, S. Sun, A. Shamir, and D. Cohen-Or, "Co-hierarchical analysis of shape structures," *ACM Trans. Graph.*, vol. 32, pp. 69:1–69:10, July 2013.

[106] N. Mitra, M. Wand, H. R. Zhang, D. Cohen-Or, V. Kim, and Q.-X. Huang, "Structure-aware shape processing," in *SIGGRAPH Asia 2013 Courses*, SA '13, (New York, NY, USA), pp. 1:1–1:20, ACM, 2013.

[107] F. Richter, C. X. Ries, N. Cebron, and R. Lienhart, "Learning to reassemble shredded documents," *IEEE Transactions on Multimedia*, vol. 15, no. 3, pp. 582–593, 2013.

[108] H. Li, Y. Zheng, S. Zhang, and J. Cheng, "Solving a special type of jigsaw puzzles: Banknote reconstruction from a large number of fragments," *IEEE Transactions on Multimedia*, vol. 16, no. 2, pp. 571–578, 2014.

[109] C. Papaodysseus, T. Panagopoulos, M. Exarhos, C. Triantafillou, D. Fragoulis, and C. Doumas, "Contour-shape based reconstruction of fragmented, 1600 bc wall paintings," *IEEE Transactions on Signal Processing*, vol. 50, pp. 1277–1288, June 2002.

[110] Q.-X. Huang, S. Flöry, N. Gelfand, M. Hofer, and H. Pottmann, "Reassembling fractured objects by geometric matching," *ACM Trans. Graph.*, vol. 25, pp. 569–578, July 2006.

[111] J. B Kruskal Jr, "On the shortest spanning subtree of a graph and the traveling salesman problem," *Proceedings of the American Mathematical Society*, vol. 7, pp. 48–50, 02 1956.

[112] X. Wang, A. Jabri, and A. A. Efros, "Learning correspondence from the cycle-consistency of time," *CoRR*, vol. abs/1903.07593, 2019.

[113] F. A. Reda, D. Sun, A. Dundar, M. Shoeybi, G. Liu, K. J. Shih, A. Tao, J. Kautz, and B. Catanzaro, "Unsupervised video interpolation using cycle consistency," *CoRR*, vol. abs/1906.05928, 2019.

Glossary

3D meshes 3D models with structural polygons, such as triangles.. 30

diffusion analysis analysis technique which is evolved from spectral analysis.. 3

eigendecomposition A linear algebra operation of a matrix.. 3

eigenvalues A set of values which is computed from eigendecomposition, corresponding to the eigenvector.. 20

eigenvectors A vector which is computed from eigendecomposition, corresponding to the eigenvalue.. 3

geodesic distance the shortest path between a pair of vertices in a graph.. 20

MatchLift A cycle-consistent matching technique for shape matching.. 16

puzzle solving solve jigsaw puzzles. In recent works, the input jigsaw puzzle pieces are in square only.. 7

SHED Shape edit distance, a segment-wise matching technique.. 8

spectral analysis mathematical analysis technique with variously application in a range of domains.. 3