Segmentation and Recognition of 3D Point Clouds within Graph-theoretic and Thermodynamic Frameworks

A thesis presented

by

Anupama Jagannathan

to

the Department of Electrical and Computer Engineering

in partial fulfillment of the requirements

for the degree of

Doctor of Philosophy

in

Electrical Engineering

in the field of

Signal Processing

Northeastern University

Boston, Massachusetts

August 2005

NORTHEASTERN UNIVERSITY

Graduate School of Engineering

Thesis Title: Segmentation and Recognition of 3D Point Clouds within Graph-theoretic and Thermodynamic Frameworks.

Author: Anupama Jagannathan.

Department: Electrical and Computer Engineering.

Approved for Thesis Requirements of the Doctor of Philosophy Degree:

Thesis Advisor: Prof. Eric Miller	Date
Thesis committee: Prof. Jennifer Dy	Date
Thesis Committee: Prof. Dana Brooks	Date
Thesis Committee: Michael McCormack	Date
Department Chair: Prof. Steve McKnight	Date
Director of the Graduate School: Prof. Yaman Yener	Date

NORTHEASTERN UNIVERSITY

Graduate School of Engineering

Thesis Title: Segmentation and Recognition of 3D Point Clouds within Graph-theoretic and Thermodynamic Frameworks.

Author: Anupama Jagannathan.

Department: Electrical and Computer Engineering.

Approved for Thesis Requirements of the Doctor of Philosophy Degree:

Thesis Advisor: Prof. Eric Miller	Date
Thesis committee: Prof. Jennifer Dy	Date
Thesis Committee: Prof. Dana Brooks	Date
Thesis Committee: Michael McCormack	Date
Department Chair: Prof. Steve McKnight	Date
Director of the Graduate School: Prof. Yaman Yener	Date

Copy Deposited in Library:

Reference Librarian	Date

Abstract

In the context of recognition of free-form objects, which are characterized by range sensor-generated 3D point clouds, this dissertation addresses two fundamental issues (1) segmentation of surface meshes constructed over the input point clouds, (2) determination of correspondence between the Scene and Model point clouds. Many existing recognition systems require uniform sampling of the Model and the Scene or they assume that these point clouds overlap. This dissertation describes the solutions to the segmentation and correspondence problems without resorting to any of these restrictive assumptions.

Mesh segmentation is an important step toward deriving an efficient representation of the underlying object and is challenging due to noisy input data. In the proposed approach, curvedness, which is a rotation and translation invariant shape descriptor, is computed at every vertex in the input mesh. Iterative graph dilation and morphological filtering of the outlier curvedness values result in multiple, disjoint sub-meshes corresponding to the physical parts of the underlying object. Results indicate that the algorithm compares well with the existing state-of-the-art approaches and it provides robust segmentations in the presence of noise.

The second contribution of this thesis is toward the determination of a one-to-one correspondence between the Scene and the Model point clouds during recognition, when the cardinalities of the two point sets are orders of magnitude different. Formulations for graph enthalpy and the Gibbs free energy capture the structural nuances between a pair of graphs and the spatial differences between the underlying point sets. The desired correspondence is obtained by tackling a sequence of inexact graph matching problems that optimizes the Gibbs free energy. Results indicate that the proposed approach outperforms many existing state-of-the-art graph matching algorithms in dealing with clutter and noise.

The third contribution of this thesis aims at reducing the computational and storage burden to enable real-time recognition. A graph-based mesh decimation algorithm is proposed to obtain shape-preserving coarser approximations of a highly detailed 3D surface mesh. A degradation metric is then derived to link hierarchical decimation with the multi-scale correspondence.

Acknowledgments

The best part of pursuing a PhD at Northeastern University was having the opportunity to work with Professor Eric Miller, my thesis advisor. Eric provided the ultimate freedom to explore my developing interests. His question: 'What's new this week?', his patience and critical thinking during our undeniably long weekly meetings and his words of appreciation greatly boosted my performance levels. His advice on the need to think outside the box holds the key to the unification of classical thermodynamics, graph theory and computer vision. I would like to express my sincere gratitude to Eric for being such an awesome advisor.

The financial support provided by Textron Systems is gratefully acknowledged. For expressing their genuine appreciation for my work, special thanks are due to Dr. Robert Kessler, Dr. Charles Davis and Dr. Michael McCormack of Textron Systems. Many thanks are due to Professors Dana Brooks and Jennifer Dy for serving on my committee. It was indeed a pleasure to learn Digital Signal Processing under the guidance of Professor Brooks during Fall 2003 and his support and understanding during that semester is graciously acknowledged.

Sincere thanks are due to everyone at CenSSIS -NU especially, Professor Michael Silevitch, Kristin, Brian, Deanna, Anne, Beeta, Greg and Mohamed. Cynthia Bates's amiable personality and her help with many administrative issues is noteworthy. Srikanth Vadde deserves a special mention for offering cookies, Indian sweets and for his advice on numerous issues.

I will forever be indebted to my parents who in addition to their words of encouragement, put extraordinary efforts toward setting very high standards for learning and infusing good work ethics. Nitya's good words have always worked wonders especially after the qualifying exams and before the proposal defense! Without my brother Basanth's good intentions, smart thinking and the much needed support and trust, this dissertation would never have come to fruition. His emphasis on quality rather than quantity, his advice on resilience and various demos on planning, organization and prioritization have led me to this day when I stand tall with my head held high, proud of my accomplishments!

Contents

List of Figures

List of Tables

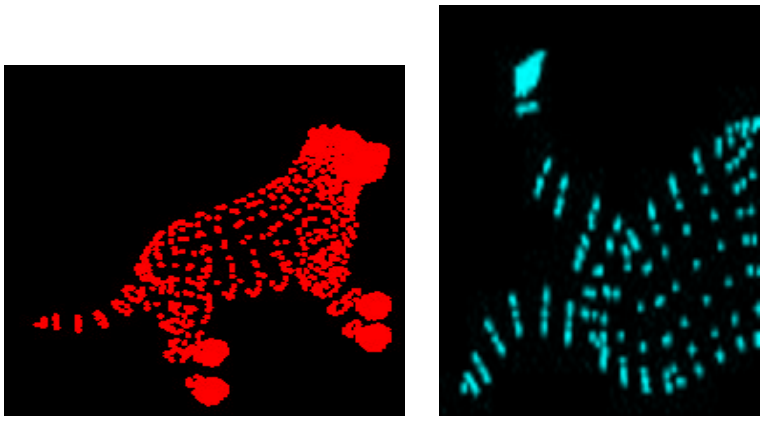
Chapter 1

Introduction

One goal of computer vision research is to design systems that provide human-like visual capabilities so that a certain environment can be sensed and interpreted to take appropriate actions. This involves the physical elements of illumination, geometry and image formation as well as the intelligent aspects of interpretation and understanding. Such a system if efficiently designed, can find use in a number of applications such as recognition of targets in defense applications, automated inspection of industrial assembly parts and autonomous vehicle navigation [11, 97]. Toward an efficient design of an automated system for the recognition of rigid, free-form objects which are characterized by 3D point clouds, this thesis addresses practical issues that arise in the segmentation and the matching aspects of the problem.

1.1 Automatic Scene Analysis

A 3D point cloud is an unstructured collection of points in the three dimensional space. Such point clouds are generated using range sensors [112]. As illustrated in Figure 1.1, a *3D model* or template of an underlying object is characterized its complete or near-complete unstructured point cloud. The *scene* (test or query data) consists of partial 3D point cloud of a known 3D object. An *interpretation* of the scene is then defined as knowing *which* model is located *where*



(a) Model (Template): Complete or near complete 3D point cloud

(b) Scene (Test or query data) Partial 3D point cloud

Figure 1.1: Scene and the model point clouds are sampled at different points in time, therefore the point sets are non-overlapping i.e., no two points correspond to the exact same location in the 3D coordinate space. Also, due to sensor inaccuracies the point clouds may be noisy. Recognition of the scene involves (1) classification of the scene as an instance of the stored object model (2) determination of orientation parameters that would align the scene with respect to the identified model (3) determination of one-to-one correspondence between scene and model points

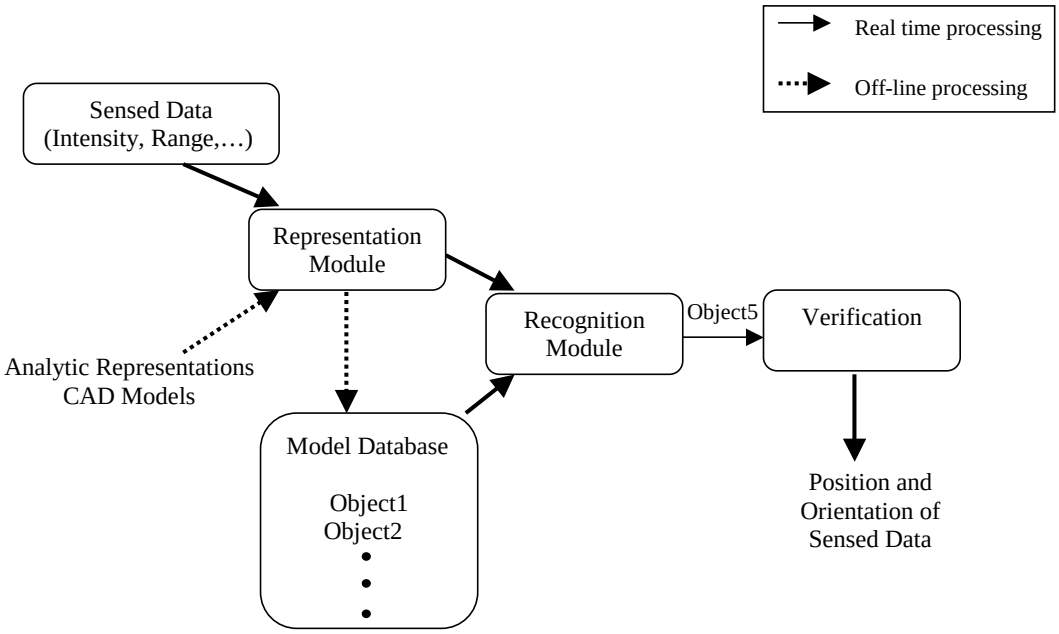


Figure 1.2: Components of a 3D vision system for representation and recognition. For the automated recognition system to be powerful, the representation and the recognition phases must be designed to work in tandem.

in the scene. Such an interpretation binds the entities in the scene to the models that we already have the knowledge about. Efficient interpretation of a scene requires the solution to three sub-problems. The first problem deals with *classification*, wherein the scene is classified as an instance of a stored object model. The second problem involves the determination of the orientation parameters (rotation, translation) that would align the scene with respect to the identified object model. The third problem involves the determination of the location of the scene within the identified model. Classification, orientation determination and localization together constitute the problem of *object recognition*.

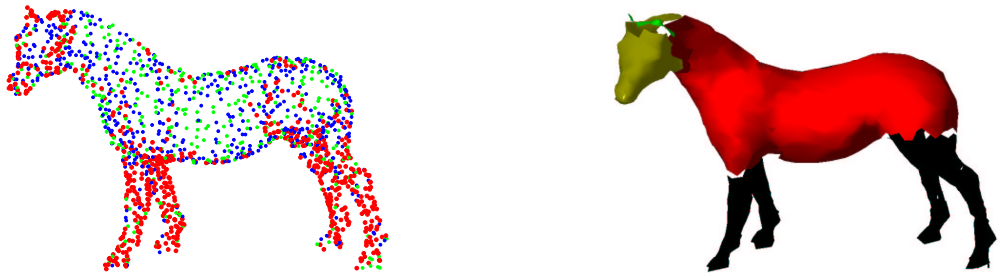
Scene analysis is indeed a difficult problem, as the recognition system needs to draw useful inferences from a point cloud, which in itself is not very informative. Therefore, instead of using a raw point cloud based information for recognition purposes, a rich, meaningful description of the object's shape/composition and connectivity information between different *parts* in the object is extracted from the point cloud data. This constitutes the task of object *representation*.

Stated succinctly, the design of a computer vision system involves a two stage processing:

1. Representation: The objective is to derive a rich, compact yet meaningful description of the object for efficient storage and for fast and accurate retrieval during recognition.

2. Recognition: The derived spatial and geometric descriptions of the partial point cloud from the scene are compared with stored models of objects in order to identify what is present in the scene. As mentioned before, this involves the tasks of classification, determination of alignment parameters and localization.

Figure 1.2 illustrates the various steps involved in building such a recognition system. A good representation scheme is required for efficient recognition since a poor representation scheme will put a heavy burden on the recognition system when it attempts to retrieve corresponding object models from the database, and there exists a very high possibility of incorrect retrieval/match.



(a) Colors on the vertices correspond to different curvedness thresholds intervals

(b) Segmentation using proposed algorithm

Figure 1.3: Curvedness [19] is a rotation, translation invariant shape descriptor which is popularly used for object representation/recognition. In (a), the vertices have been partitioned into three categories corresponding to three different curvedness threshold intervals. As shown in (a), the known segmentation approaches result in small fragments of connected vertices as opposed to regions representing the physical parts of the underlying object. Panel(b) illustrates a more practical segmentation.

1.1.1 General Challenges in 3D Object Recognition

For representation purposes, the point cloud is often triangulated and the resulting mesh (surface/volumetric) is used to obtain geometric descriptions of the underlying object. The intense research in the field of object recognition over the past two decades or so reflects the importance of certain representation and recognition challenges that need to be addressed by any vision system. We discuss below some emerging challenges that are of interest to us:

1. *Object shape characterization in the presence of noise*: In three dimensions, noise causes the perturbation of the data points. For the representation scheme to be effective, such point clouds need to be pre-processed to remove noisy artifacts, prior to computation of shape features. Also, it is a challenging to compute shape descriptors fairly accurately from surface triangulations and the inability to do so may result in an unreliable segmentation and representation as illustrated in Figure 1.3(a).

2. *Model and Scene Description*: Most existing vision systems represent a 3D object model as a collection of multiple 2D views. Also, the scene is a certain 2D view of the underlying

object or alternatively it is a projection of 3D data onto the 2D space. The recognition process in such vision systems, also referred to as view-based recognition [107, 97], involves classifying the 2D scene data as an instance of one of the 2D views in the model and further, determining the orientation of the scene with respect to the 2D model view. Although image-based approaches simplify the task of recognition, they struggle to address some other major issues such as sensitivity to lighting conditions. Object centered representations [97] on the other hand, are highly sensitive to noise.

3. Size of the object Database: Efficient retrieval i.e., accurate indexing into the database and classification of the scene as an instance of the stored object model, is strongly linked to the description and the organization of a model as a set of connected entities. Model databases for real-world applications contain tens of thousands of models. The matching cost increases as more and more models get added to the database because the input scene representation needs to be matched with all object models present in the database. Thus, there is an increasing need for the development of efficient pruning strategies and imposition of geometric constraints to enable real-time recognition.

4. Learning: Most all existing vision systems use a pre-compiled database of models for recognition of the input scene data. A system breakdown occurs when the scene is not an instance of one or more of the object models stored in the database. Ideally, a vision system needs to be able to learn from a new input as well as from the models in the existing database. For this to happen, the system must be able to learn the description of the unknown object and in addition, have the ability to represent and store such modeling information, not originally present in the object database. Noise is another issue that the existing vision systems have been grappling with; a failure to recognize noisy instances of models already present in the database.

5. *Articulated vs Rigid Objects*: Rigid objects such as a tea-pot have *immovable parts*. Conversely, objects with movable parts are referred to as articulated objects. In addition to these two categories, there exists a class of deformable objects which are purely non-rigid. While it is fairly straight forward to recognize articulated objects i.e., by incorporating the possible ranges of movement for the part(s) of objects that is (are) movable into the representation scheme, the representation and recognition of deformable objects is a very challenging problem.

6. *Recognition Methodologies*: An important step in recognition is the determination of correspondence between the scene and the model data points. While this problem is trivial when the scene point cloud provides a complete description of the underlying object, it is challenging when the scene provides only a partial description of the object of interest. Most existing systems assume that the scene and a subset of the points in the model overlap. In reality, different sensors can lead to different samplings of data points, and under such circumstances, the existing systems would fail to function. Also, the known matching strategies are either spatial or structural, but not both. The graph-based structural approaches [40, 71, 54, 12, 72], are highly sensitive to noise and perform well as long as the scene and the model graphs have the similar number of points. Information theoretic approaches [56, 88, 106], on the other hand, do not attempt to match the implicit structure connecting the data points i.e., matching is performed purely in the feature/spatial domain.

1.2 Proposed System

1.2.1 Assumptions

We are interested in the recognition of rigid, free-form objects described by 3D point cloud data. However, the study does not include statistically defined shapes such as textures, fractals or other objects such as trees and bushes. We restrict ourselves to scenes containing partial,

uncluttered point clouds of a single object. We assume that the model exists in the database prior to recognition. Also, we assume that the scene or the model point clouds are not sparse. But we do not assume that these point clouds are (1) uniformly sampled or, (2) overlapping.

1.2.2 System Design

There are a number of processing steps involved in our recognition system and they can be divided into two stages: model construction and recognition. During the model construction stage, the following steps are carried out (i) acquisition of a complete or near complete point cloud of an object using range sensors (ii) construction of a triangular mesh using the commercially available software, *Point2Polys* (iii) conversion of a non-manifold mesh into the corresponding manifold surface (iv) Partitioning a manifold mesh into multiple, disjoint sub-meshes using a graph-theoretic segmentation algorithm (v) building a graph-based representation using the sub-meshes identified in (iv).

During the recognition stage, a partial point cloud of an object is presented to the system. Given the sensed data, the following actions need to be taken (i) classification of the scene using its representation (ii) estimation of the rotation and the translation parameters that would align the scene with respect to the identified model (iii) determination of correspondence between the scene and the identified model. The performance of the recognition stage, in terms of its accuracy and speed, essentially determines the usefulness of the system in real-world situations.

1.2.3 Challenges Considered

We address the following important issues related to the design of a recognition system for rigid, free-form 3D objects obtained from unstructured point clouds:

- Given a 3D surface mesh constructed over the point cloud of an underlying object by using a commercially available software, how can we repair this mesh so that along every edge

in the mesh there are at most two triangles incident? This problem is considered in Chapter 3 of this thesis.

- How to segment the surface mesh into disjoint sub-meshes such that each sub-mesh corresponds to a certain physical part provided in Chapter 4.
- How to establish a one-to-one correspondence between the scene and a subset of the identified model points. This problem is considered in Chapter 5.
- How to coarsen model point clouds so that the object of interest can be described by a point cloud with fewer points. This problem is addressed in Chapter 6.
- How to determine the number of model points required for robust (scene-model) correspondence? A solution is proposed in Chapter 6.

We believe that these issues are inter-coupled in terms of the representation and the recognition schemes that can be used.

1.2.4 Contributions

Figure 1.4 presents an overview of the approach and a brief description of the modules listed in the figure is given below. The following are the key features that distinguish our 3D recognition system from other existing state-of-the-art systems in computer vision:

- *Conversion to manifold surface meshes*: Usually, the 3D surface meshes constructed from unstructured point clouds are non-manifold due to the presence of topological singularities caused by human or software induced 'bugs' [98]. In this dissertation, we are interested in repairing surface meshes wherein the reason for a non-manifold behavior is attributed to the presence of a large number of singular edges i.e., edges along which more than two triangles are incident. A greedy surface conversion algorithm based on geometry and topology is presented to convert a three dimensional non-manifold triangulation into the

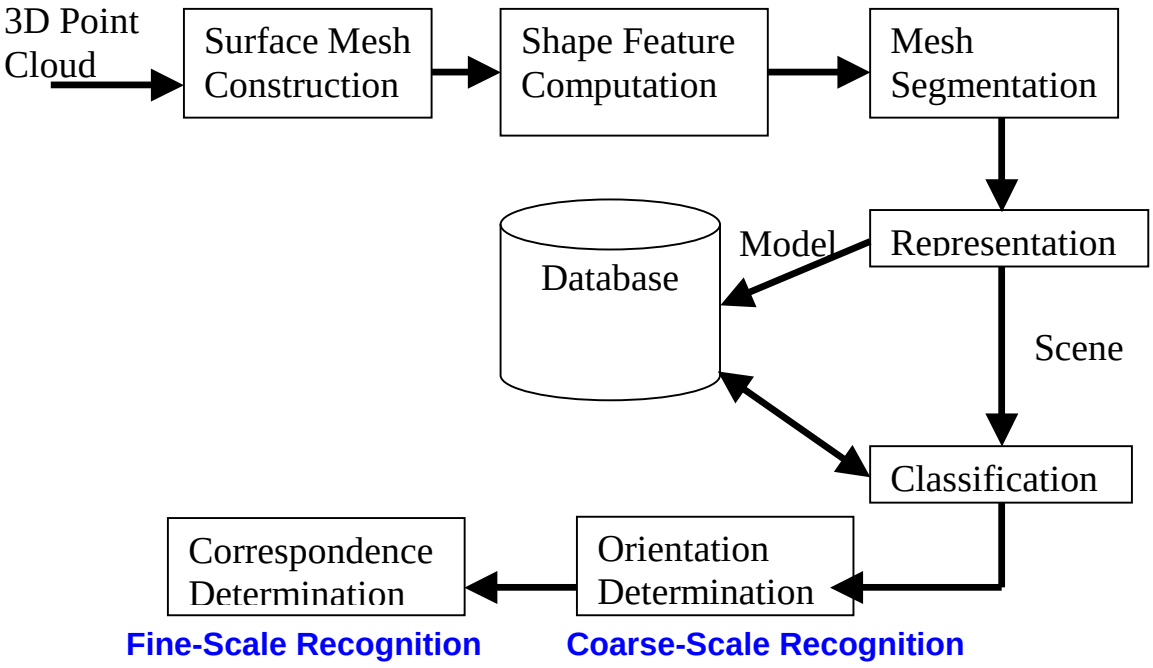


Figure 1.4: Outline of the proposed approach for representation and recognition

corresponding manifold surface. Since it is assumed that every vertex is reachable from every other vertex in the input triangulation, an arbitrary vertex is selected. A seed triangle incident on this vertex is then identified such that it is as *regular* [61] as possible and bears a close resemblance to its neighboring triangles with respect to certain geometric attributes. The input triangulation is then partitioned into the constituent triangles. The surface growing process works by *stitching* [98] optimal triangles along the boundary edges identified on the grown surface. The growing process terminates when all vertices present in the original non-manifold triangulation have been accounted for in the new manifold triangulation.

- *Mesh Segmentation* [125]: A new graph morphology based segmentation algorithm is proposed to address the problem of partitioning a 3D triangular mesh into disjoint sub-meshes that correspond to the physical parts of a particular object. Curvedness, which is a rotation

and translation invariant shape descriptor, is computed at every vertex in the input triangulation. Every sub-mesh is characterized by a pair of curvedness thresholds which are adaptively determined The fact that the geometric behavior of a vertex is influenced by its neighbors allows us to identify vertices with outlier curvedness values. Iterative graph dilation and morphological filtering of the outlier curvedness values result in multiple, disjoint, maximally connected sub-meshes such that each sub-mesh contains a set of vertices with similar curvedness values, and vertices in disjoint sub-meshes have significantly different curvedness values.

- *Correspondence determination* [123]: In the context of object recognition from point cloud data, a thermodynamically-inspired graph theoretic algorithm is presented to address the problem of matching the scene and the model point clouds, when the cardinalities of the two sets are orders of magnitude different. A thermodynamically inspired objective function is proposed to capture the structural nuances between a pair of graphs and the spatial differences between the underlying point sets. The desired correspondence is obtained by tackling a sequence of inexact graph matching problems that optimizes the proposed objective function.

- *Mesh Decimation* [124]: Since the object models contain highly dense point clouds of data points, not all of which are required for recognition, a shape- preserving mesh decimation algorithm is presented to obtain approximations of the fine scale surface meshes. The input mesh is segmented into multiple, disjoint sub-meshes to facilitate decimation. Given a sub-mesh, various shape clusters are identified and the vertices in those clusters are labeled as boundary/interior. Shape is preserved by considering only similar-labeled vertex pairs as candidates for a potential merge. Sub-mesh decimation is realized by merging a vertex pair that minimizes a certain graph energy based cost function.

- *Model Database*: The object model database used to test the strengths of our representation, recognition and simplification schemes consists of two categories. The first category consists of simulated point clouds of 8 different objects, specifically created to quantitatively evaluate the performance of our proposed algorithms. The second category includes unstructured point clouds of 15 objects obtained from an online repository.

1.3 Thesis Organization

The rest of the thesis details the key ideas outlined above. Chapter 2 presents the literature survey of the related work in object recognition and the technical background in graph theory and classical thermodynamics, required for understanding the rest of the dissertation. Chapter 3 describes the proposed algorithm for conversion of non-manifold triangulations into manifold surface meshes. In Chapter 4, the motivation for a new segmentation approach is presented and the proposed graph morphology-based segmentation algorithm is described in detail. The results of segmentation are then used to derive a two-tier representation scheme for rigid free form objects. In Chapter 5, motivated by classical thermodynamics, a point cloud matching scheme is described for the determination of a one-to-one correspondence between a partial scene point cloud and an identified model point cloud. In Chapter 6, we address the mesh decimation problem, the solution to which is cast within a graph theoretic framework. In Chapter 7, the proposed contributions to segmentation, decimation and correspondence problems are summarized and directions for future work are provided as well.

Chapter 2

Related Work and Technical Background

A free-form object is assumed to be composed of one or more non-planar surfaces. Typical examples of free form objects include sculptures, car bodies, human faces and terrain maps [55, 97]. General mathematical properties exhibited by object representation schemes are ambiguity, conciseness, uniqueness and locality [97].

Ambiguity or completeness measures the representation's ability to completely define the object in the model space. *Conciseness* represents how efficiently or compactly the description defines the object. *Uniqueness* is used to measure if there is more than one way to represent the same object, given the construction methods of representation. If the representation is unambiguous and unique, then there is one-to-one mapping from the object to the representation. *Locality* of an object representation is of interest in applications of recognition in the presence of occlusion. A representation that explicitly reveals local geometrical structure is characterized as occlusion tolerant and hence is better suited for such applications. However such representations are generally verbose. The importance of the above mathematical properties depends on the application context. In the case of object recognition applications, completeness and compactness are often sacrificed in favor of uniqueness [32]. The pragmatic issue of performance often makes such compromises appropriate. Depending on the type of the sensor used to capture the data and the method of registration, the points on the surface can be in one of the two basic forms. In

the first case, all points form a *point cloud* where no connectivity information is known. In the second case, the data are *structured* via relationships (e.g., image-plane connectivity) known a priori in each view [97]. For unstructured point clouds, the default computer representation is a polygonal mesh.

This chapter surveys the prior art in three important topics of computer vision and visualization that relate to this thesis: mesh segmentation, representation, pose estimation and localization for recognition.

2.1 3D Mesh Segmentation

3D mesh segmentation refers to the problem of partitioning a given 3D triangular mesh into large, homogeneous, disjoint sub-meshes usually based on certain geometric properties of the vertices that comprise the mesh. Typically, such meshes are obtained from unstructured point clouds of underlying 3D objects, which, in turn are generated by range sensors [26]. Examples of applications that benefit from mesh segmentation include vertex simplification, object recognition and scene understanding [75]. The success of several existing mesh segmentation algorithms, judged based on their ability to provide meaningful partitions, can be attributed to the specific applications for which they have been designed. Over the past decade or two, the problem of mesh segmentation has received significant attention and numerous algorithms have been proposed to solve the problem. We have grouped the related work into three categories.

The first category covers methods that use Reeb Graph ideas [20, 109, 110, 49], based on Morse theory, for segmentation of meshes. A Reeb graph, or a contour tree, represents the topological skeleton of the underlying 3D object, and uses height functions for determination of level-set curves. Each such curve represents a vertex in the graph. Segmentation is achieved by extracting edges that link different pairs of vertices. The main drawback of the basic Reeb graph approach is the determination of appropriate height functions that would lead to good segmentations. Also, the approach is highly sensitive to noise [109]. Various extensions to this

Segmentation Algorithm	Shape Descriptors	Perceptual Aspect	Post Processing Overhead	Stability to Noise
Basic Reeb Graph [20]	height functions; Affine-variant	No	No	No
Affine-Invariant Reeb Graph [109]	height functions; Affine-Invariant	No	No	Yes
Medial Axis Transformation [48]	radius function; Affine-Invariant	No	No	No
Watershed [75]	Total Curvature at every vertex	No	Yes; merging insignificant regions	No
Binary Morphology [83]	Curvature at every vertex	No	Yes;conversion to graphs	No
Fast Marching Watersheds [111]	principal curvatures	Yes	Yes;Merging small gaps	No

Table 2.1: A comparison of some state-of-the-art mesh segmentation approaches

approach have been proposed, which include: formulation of application dependent height functions [20], the discrete Reeb graphs [110], extended Reeb graphs [49] and affine-invariant Reeb graphs [109]. The affine-invariant Reeb graphs [109] provide a rotation and translation invariant segmentation. However, to obtain good results, the authors in [109] suggest that the input mesh be uniform. Also, [109] does not specifically address the perceptual aspect of segmentation.

The second category covers methods that extend classical segmentation approaches used in image analysis, to three dimensions. Mangan and Whitaker [75] propose a watershed algorithm for segmentation. They compute total curvature at every vertex and identify local curvature minima that represent thresholds. Adjacent vertices with uniform curvatures between two minima are labeled as belonging to the same region. The algorithm is designed to provide good segmentations only for uniform meshes. Rössl et.al in [83] propose a boundary extraction algorithm by extending binary morphology to three dimensions. They compute curvatures at every vertex in

the mesh and subsequently threshold them to obtain a binary feature vector. Further, they apply various binary morphological operators to obtain the skeleton of the feature region. Each skeleton is then post-processed to obtain a graph-based characterization. The Medial Axis Transformation (MAT) [48] provides an affine-invariant segmentation but is sensitive to noise [109]. In the planar case, the medial axis of the shape is a graph defined as the locus of the centers of all maximal discs contained inside the shape and having at least two points of contact with its boundary.

The third category covers algorithms that perform perceptual segmentation. These algorithms are based on minima theory that defines a framework for how human perception will decompose an object into its constituent parts [111]. Essentially, this theory defines boundaries as lines of negative minima curvature. Wu and Levine in [52] address the perceptual aspect by formulating an algorithm based on the simulated distribution of electrical charge across the surface of a mesh. Page et.al in [111] use the principal curvatures as shape descriptors and implement a variation of the watershed algorithm to identify regions that are bounded by lines of negative minima curvatures. Table 1 summarizes other key differences among some of these algorithms.

The performance of most of the existing state-of-the-art segmentation approaches is heavily reliant on the availability of uniformly sampled point clouds [75, 20, 109, 110, 49]. For the segmentation results to be useful for higher level tasks such as object recognition, it would be advantageous to develop an algorithm that best mimics the human visual system, in terms of isolating different physical parts in an object [52, 37]. In Chapter 4, we address the challenge of segmenting a 3D surface mesh into physical parts [52] without making any assumptions about the clouds being sampled uniformly. To meet this objective, we consider a graph-morphology based region growing algorithm which uses curvedness [55, 19, 24], computed at every vertex in the mesh as the similarity metric for segmentation purposes.

2.2 3D Object Representation

Representations used in computer vision can be fundamentally classified into *object-centered* and *view-centered* categories. Techniques that use an object-centered representation attempt either to describe the entire 3D volume occupied by the object or use view-independent features of objects such as corners and straight edges, projected onto a 2D image space, which are ultimately detectable by various image processing operations. On the other hand, view-centered representations rely on specifying the *appearance* of the object from a single or a set of multiple views and use features such as silhouettes of a shape that are not intrinsic to an objects. Representations can also be distinguished depending on whether they use local or global shape descriptors.

Among the object-centered representations are the boundary-based methods, volumetric descriptors and sweep representations (based on generalized cones). [97, 26] present comprehensive reviews of the object-centered representations. The local boundary-based methods describe an object by lists of faces, edges and vertices. Since polyhedral representations of curved objects require large amounts of space to adequately approximate them, both planar and quadric equations were used to approximate them. In [13], surfaces are classified into primitive shapes such as peak, pit, saddle etc., based on the signs of the gaussian and the mean curvatures. Typical global volumetric representations include voxels, octrees and superquadrics [26]. In a voxel representation, an object is described as the union of non-overlapping cubes, where the cubes occupy positions in a 3D lattice. Octrees describe objects in a hierarchical tree-like structure. A superquadric representation for an object is obtained by fitting an implicit equation to the set of data points that describe the object. The limited set of shapes represented by the superquadric primitives can be extended to represent more complex shapes by adding parameters and deformations to the implicit equations. Such an approach is computationally very expensive.

Spherical representations for describing 3D objects have a rich history in the vision community. Ikeuchi and Hebert [35] provide an excellent review of these representations. The representations include the class of orientation based descriptors such as extended Gaussian image(EGI), support-function based representations (SFBR), the complex EGI (CEGI), distributed EGI (DEGI), the generalized Gaussian image(GGI) and spherical attributed image (SAI). In the EGI representation, it is assumed that the object is evenly sampled into surface patches. At each surface patch, we define a surface normal with unit mass. The normals are then mapped onto the unit sphere. The disadvantage of this approach is that it cannot be used for representation of nonconvex objects, and further, while the representation allows for the determination of the orientation parameters, the position of the object cannot be determined. The CEGI representation addresses the latter issue, but it still can handle only convex objects. The DEGI representation can deal with non-convex objects but it cannot handle occlusion. The GGI representation can handle convex as well as non convex objects, but fails when parts of the objects are occluded (partial surface matching). The SAI addresses the shortcomings of many of these earlier spherical representations (EGI, DEGI, CEGI). The SAI representation maps points on an object's surface to vertices on a quasi-regular tessellated sphere. Local surface characteristics are stored at the vertices of the tessellated sphere that correspond with the vertex point. The surface point to vertex mapping is determined by deformation of the sphere on the object's surface. The representation provides rotational invariance and has to ability to deal with occluded objects. However, the entire representation assumes that the mesh is regular and uniform, which for practical purposes, is not a realistic assumption.

The view-based approach to 3D object recognition represents an object as a collection of 2D views. Popular view centered recognition systems include the parametric eigenface technique, aspect graphs [26], COSMOS (Curvedness Orientation Shape Map On Sphere) [55] and Spin Images [69, 68]. In the vision system proposed in [55], a histogram of shape index values is used to characterize the surface curvature of a view. The histogram bins store the amount of area on

a view that lies within the range of shape index values. During recognition, these histograms are quickly matched using moments. Their recognition system works well for single object scenes not containing polyhedral objects. Johnson and Hebert [69, 68], use spin images for recognition purposes. Their spin images are 2D histograms of the surface locations around a point. The spin image is generated by using a normal at a point and by rotating a cutting place around the point, using the normal as the axis of rotation. As the plane spins around the point, the points of intersection between the plane and the surface are used to index a 2D histogram. The drawback of their system is that they assume that the data points describing the object are uniformly sampled.

2.3 Matching Strategies

In the previous section, we discussed various approaches for representing objects. The next step is the classification and localization of objects that are present in the scene. Recognition is performed by matching features derived from the scene with those stored in the model database. The popular and important approaches to the recognition and localization of 3D objects are (i) graph matching (ii) information- theoretic matching (iii) interpretive tree search (iv) hypothesize and test and (v) Iterative model fitting.

Graph matching approaches [25, 16, 12, 7, 71, 89, 21, 73] capture the structural properties of objects for ease of recognition. The scene and the model are described using attributed graphs, where each vertex characterizes a scene or a model feature and the edge between vertices represents the relation between two features. Graph matching algorithms find the best match by minimization of the edit distance [108, 54, 66, 85, 74]. The idea behind edit distance is that it is possible to identify a set of basic edit operations (insertion/deletion) on the set of vertices and edges in a graph that could make it isomorphic with another graph. Associated with these operations is a cost and hence, the objective is to find a sequence of edit operations that will minimize the cost. Other algorithms for graph matching include graph isomorphism [12], subgraph isomorphism [54, 66, 81, 103, 118], matching using graduated assignment [40, 82, 104],

and matching using Markov chains [90]. The scene and the model can alternatively be modeled by their shock graphs. In [119, 96, 72], recognition is performed by the minimization of the edit distance computed over the shock graphs. While graph matching algorithms primarily differ with respect to the implicit graph structures used for matching purposes, recognition by minimization of edit distance is now a widely accepted paradigm. The edit distance approach has its drawbacks (i) it is computationally expensive (ii) in its current form, it does not accommodate edge weights. In [116], earth mover's distance is used to measure dissimilarity between graphs, which are represented using a view-centered approach. The advantage of this distance measure is that it can deal with edge weights as well as noisy data sets.

In the recent past, information-theoretic methods have been extensively used for matching purposes. Viola and Wells [56] proposed the mutual information (MI) matching approach, to align the scene image with the model image. They search over a set of possible transformations to find the transformation that maximizes the mutual information between the scene and the model. More recently, in [88], an entropic graph approach is proposed to align the scene image with the model. The advantage of entropic methods is that they can capture non-linear relations between the features (associated with the scene and the model) in order to improve the discrimination between poor and good matches. When combined with a highly discriminatory feature set and reliable prior information, entropic methods have shown very promising results [88, 120]. In [88, 120], the minimum spanning tree is used as an entropic graph, and it has been shown that the normalized total length of the MST is a consistent estimator of the entropy.

In an interpretive tree search [18], a search tree is constructed by pairing scene features with the model features. Instead of searching the entire tree for a complete match, local geometric constraints such as distance between features are used to prune the search tree. Upon completion of the search, a global transformation is computed to determine and verify the pose of the object.

In the hypothesize- and test paradigm [18, 26], a transformation from the model's coordinate

frame to the scene's coordinate frame is hypothesized. A set of non-linear equations is formulated to characterize the transformation. The equations are solved in an effort to minimize the squared error. The resulting transformation is then used to align the scene features with the model features. The hypothesis is accepted or rejected based on the matching error.

Iterative model fitting [18, 26] is used when 3D objects are described using parametric representations. There is no feature computation or correspondence determination between model and scene features. Object recognition and pose estimation reduce to estimating the orientation parameters of the model from the scene data, and matching with the stored parametric representations. Typically, the estimation of parameters is done by solving a set of non-linear equations in an effort to minimize the squared error.

In Chapter 6, we present a thermodynamically inspired algorithm to determine a correspondence between the scene and the model point clouds by combining the goodness of the graph-based structural approaches and the entropy-based spatial matching approaches. The maximization of the proposed objective function which captures the structural and spatial differences between point sets, leads to the desired correspondence.

2.4 Mesh Decimation

Laser range scanners and medical imaging devices generate highly detailed models of intricate 3D objects. In order to achieve acceptable processing times, often the original model needs to be substituted by coarser approximations. Polygonal decimation [33, 60] refers to the problem of transforming a three-dimensional polygonal model into a simpler version, by reducing the number of polygons required to represent the underlying object. Therefore, the primary aim of decimation is to produce a surface approximation which is *as similar as* possible to the original model. Polygonal decimation algorithms that need manifold triangulations as input, can be broadly classified as :

- **Multi-pass algorithms**: In the vertex decimation algorithm [58, 43, 23], multiple iterations are made over all the vertices present in the manifold surface. During an iteration, each vertex is a candidate for removal and if it meets the specified decimation criteria, then the vertex and all triangles incident on it are deleted. The resulting mesh is re-triangulated to remove holes, generated by the decimation process. The decimation process is repeated until some termination condition is reached. By far, the most popular hierarchical simplification algorithms are based on the optimization of certain energy functionals. Most of these algorithms, also known as iterative edge contraction algorithms [60, 42, 45, 62, 65], follow a *greedy* approach to select the sequence of edge contractions. Each vertex pair being considered for merging is assigned a cost. Typically, this cost, represents the error induced as a result of a potential merging of the vertex pair in question. At each iteration, the lowest cost pair is merged. This leads to a collapse of certain edges and also generates some degenerate triangular faces which are subsequently removed. The Progressive Meshes algorithm proposed by Hoppe et al [45, 62] performs iterative contraction based on the minimization of an energy function, which evaluates the geometric compactness of the resulting representation.

- **Single-pass Algorithms**: In [46], Kalvin and Taylor propose a single-pass algorithm that works by partitioning the surface into disjoint planar patches and simplifying each of the patches, before re-triangulating them, to obtain an approximation to the input triangulation. Each patch is determined by selecting a triangular face at random and merging the adjacent faces until the triangles in the patch can no longer be fit by a plane within some error tolerance. Degenerate or highly elongated patches are prevented by employing additional constraints. In a similar approach [29], patches of triangles with nearly parallel normal vectors are determined. Each of these patches is simplified by merging coplanar or nearly coplanar polygons into larger complex polygons. Finally, the patches are merged together to get an approximated version of the input triangulation. While the two approaches appear

similar at a higher level of abstraction, the difference lies primarily in how the patches are determined. In [29], Hinker and Hansen define patches based on angles between the normals while in [46], Kalvin and Taylor define the patches based on the distance to the plane. In [75], patches are determined in a way such that each patch has a relatively consistent curvature throughout and is bounded by areas of higher or significantly different curvatures.

In Chapter 5, we present a graph theoretic approach to hierarchical mesh decimation wherein, in the context of hierarchical vertex contraction, a cost function based on graph energy is proposed to identify the optimal vertex pair for merge.

2.5 Graph Theory [34, 64, 92, 63]

An attributed graph AG is denoted by $G[(V, V_a), (E, E_a)]$, where $V = \{v_1, ..., v_n\}$ is the set of vertices and $V_a = \{av_1, ..., av_n\}$ is the set of attributes associated with the vertices [5, 10, 38]. Note that av_i may represent a single attribute or a set of attributes for vertex v_i. Similarly, E and E_a define the set of edges and the set of attributes corresponding to edges, respectively. When $E_a = \phi$ and $V_a = \phi$, which implies that there are no attributes associated with the set of edges and the set of vertices, respectively, then the attributed graph $G[(V, V_a), (E, E_a)]$ can be re-written simply as a graph $G[V, E]$.

In a directed graph, the edge set *E* consists of edges connecting ordered pairs of vertices, while in undirected graphs, the set *E* contains edges that connect unordered pairs of vertices, i.e., in an *undirected* graph, $e_{ij} = e_{ji}$, where $(v_i, v_j) \in V$. Note that e_{ij} may also be denoted by $\langle v_i, v_j \rangle$. In an undirected graph, self-loops - edges from a vertex to itself - are forbidden and so every edge links exactly two distinct vertices. If e_{ij} is an edge in the graph G, then it implies that vertices v_i, v_j are adjacent and are neighbors. For undirected graph, the adjacency relation is symmetric while for undirected graph, the adjacency relation is not necessarily symmetric.

Further, e_{ij} is said to be incident on the vertices v_i and v_j.

The degree of a vertex in an undirected graph is the number of edges incident on it. A walk in G is a sequence of vertices $\langle v_1, v_2..., v_k\rangle, k \geq 1$ such that $\langle v_i, v_{i+1}\rangle \in E$ for $i = 1, 2, ..., k-1$. A walk without repeated vertices is called a path. A closed walk with no repeated vertices other than the first and the last one is called a cycle. A graph is connected if there is a path between any two vertices. A tree is a connected graph without cycles and, when the tree connects all vertices, it is called a spanning tree. A minimum spanning tree is the spanning tree whose sum of edge weights are the least among all possible weighted spanning trees for a given graph G.

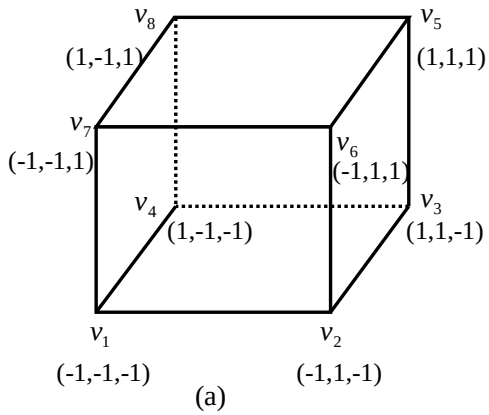
A complete graph is an undirected graph in which every pair of vertices is adjacent. A bipartite graph is an undirected graph $G(V, E)$ in which V can be partitioned into two sets V_1 and V_2 such that $e_{uw} \in E$ implies that either $v_u \in V_1$ and $v_w \in V_2$ or $v_w \in V_1$ and $v_u \in V_2$. That is, all edges go between the two sets V_1 and V_2. A *matching* M of a graph $G(V, E)$ is a subset of edges with the property that no two edges of M share the same vertex. When the cardinality of matching is $\lfloor \frac{|V|}{2} \rfloor$, the largest possible in a graph with $|V|$ vertices, the matching is said to be complete or perfect. A *weighted* undirected graph is an undirected graph for which each edge has an associated non-negative real weight, usually computed as the distance between the corresponding vertices.

A subgraph of a graph X is a graph Y such that $V(Y) \subseteq V(X)$, $E(Y) \subseteq E(X)$. If $V(Y) = V(X)$, then Y is called a spanning subgraph of X. Any spanning subgraph of X can be obtained by deleting some of the edges from X. A subgraph Y of X is an induced subgraph if two vertices of $V(Y)$ are adjacent in Y if and only if they are adjacent in X. Any induced subgraph of X can be obtained by deleting some of the vertices from X, along with any edges that contain a deleted vertex. A clique is a subgraph that is complete.

In an undirected graph, the weight along any edge satisfies the following properties:

1. $w_{ij} > 0$ if $e_{ij} \in E$ and $w_{ij} = 0$ if $e_{ij} \notin E$

2. $w_{ij} = w_{ji}$



(a)

Vertex	ρ(vertex)
v_1	(-1,-1,-1)
v_2	(-1,1,-1)
v_3	(1,1,-1)
v_4	(1,-1,-1)
v_5	(1,1,1)
v_6	(-1,1,1)
v_7	(-1,-1,1)
v_8	(1,-1,1)

(b)

Figure 2.1: (a) A cube with eight vertices (b) A possible representation in the three dimensional space

The *degree* d_i of a vertex $v_i \in V$ in a weighted undirected graph $G(V, E)$ is given by the sum of weights of all edges incident on vertex v_i. Mathematically, this is given as:

$$d_i = \sum_{v_j \in V} w_{ij} \tag{2.1}$$

Given a graph $G(V, E)$ with *n* vertices, its *weighted adjacency graph* $A(G) = [w_{ij}]$ is an $n \times n$ matrix with rows and columns indexed by V, whose entries are w_{ij} as defined above. A *diagonal matrix* $D(G) = diag(d_i)$ is an $n \times n$ matrix indexed by V and has the vertex degrees along the

principal diagonal of the matrix. The *weighted Laplacian L(G)* of the graph *G* is then defined as:

$$L(G) = D(G) - A(G) \tag{2.2}$$

A representation ρ of a graph *G* in $\Re^m$ is a mapping from V into $\Re^m$. Informally, a representation is the position of the vertices in an *m*-dimensional space.

Figure 2.1(a) provides an example of a possible representation for a cube in $\Re^3$.Here, $V = \{v_1, v_2, .., v_7, v_8\}$. A possible mapping for *V* is given in Figure 2.1(b).

2.6 Thermodynamics [2, 67]

Thermodynamics is the study of energy transformations in systems. A system may be homogeneous or heterogenous. The energy interactions in heterogeneous systems are analyzed by dividing the system into a number of homogeneous components. The state of a system is characterized by its pressure P, volume V, temperature T, and composition n. For a fixed amount or composition of a substance contained in a system, the state of the system can be completely determined by any two of the three quantities P, V or T. Energy transformations during which pressure, temperature and volume remains constant are called isobaric, isothermal and isochoric transformations, respectively.

2.6.1 Enthalpy

The total energy possessed by a substance is called its internal energy, U and is the total kinetic and potential energy of the molecules present in the substance. According to the first law of thermodynamics which essentially is the principle of conservation of energy for thermodynamical systems, the heat content in a system is given by its enthalpy H and is defined as:

$$H = U + PV \tag{2.3}$$

where P is the pressure and V is the volume occupied by the molecules. At constant pressure during substance conversion, the heat absorbed or produced by the system (or change in heat content) is characterized by its change in enthalpy ΔH.

$$\Delta H = \Delta U + P\Delta V \tag{2.4}$$

In a sense, ΔH results due to addition or deletion of bonds during the substance conversion process.

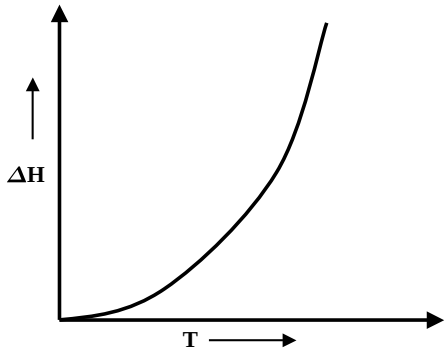
2.6.2 Entropy

A spontaneous direction of change is one that does not require work to be done to bring about the change, for instance, for a gas to be confined to a smaller volume some work needs to be done. Thus, a gas does not spontaneously contract because to do so the chaotic motion of its molecules would have to take them all into the same region of the container. Gas expansion on the other hand, is a spontaneous change; it is a consequence of increasing disorder. Spontaneous changes are always accompanied by a dispersal of energy into a more disordered form and this is the essence of the second law of thermodynamics. Entropy is a measure of the spatial disorder of a system. The second law uses entropy to identify the spontaneous changes among those permissible changes. The entropy of a system increases with increase in temperature. According to the third law of thermodynamics, the entropy change accompanying an physical or chemical transformation approaches zero as the temperature approaches zero: $\Delta S \longrightarrow 0$ as $T \longrightarrow 0$.

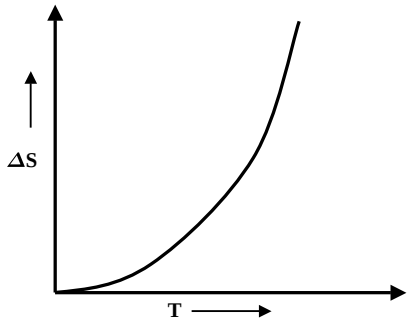
2.6.3 Gibbs Free Energy

The criterion for spontaneous change is defined by the Gibbs free energy, which at constant temperature is given by:

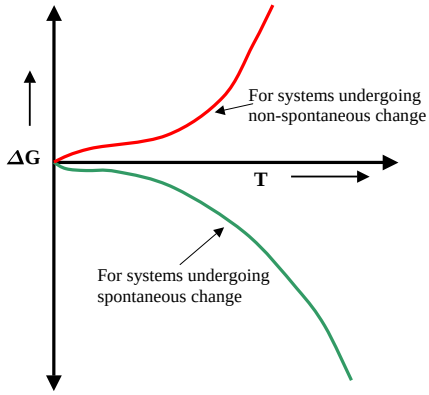
$$\Delta G = \Delta H - T\Delta S \tag{2.5}$$



(a) Change in Enthalpy as a function of temperature



(b) Change in Enthalpy as a function of temperature



(c) Change in Gibbs energy as as function of temperature

Figure 2.2: Variations in $\Delta H, \Delta S, \Delta G$ as a function of temperature T. Observe that as the temperature decreases and tends to zero, $\Delta H, \Delta S, \Delta G$ approach zero, leading to an equilibrium. At equilibrium i.e., when $T = 0$, $\Delta H = 0, \Delta S = 0, \Delta G = 0$. Note that ΔG must necessarily be less than zero for the reactions to be spontaneous or feasible.

The variations in enthalpy, entropy and Gibbs energy as a function of temperature T is shown in Figure 2.2. Since the entropy is always positive, it follows that the free energy decreases with increase in temperature, while the pressure is maintained constant. The free energy decreases most sharply when entropy of the system is large. Thus, entropic contributions are dominant at very high temperatures.

2.6.4 Thermodynamics of Heterogeneous Systems

A heterogenous system consists of more than one substance. For such systems, the Gibbs free energy is given by:

$$\Delta G = \mu_1 \Delta n_1 + n_1 \Delta \mu_1 + \mu_2 \Delta n_2 + n_2 \Delta \mu_2 + ... + \mu_j \Delta n_j + n_j \Delta \mu_j \tag{2.6}$$

subject to the constraint that

$$n_1 + n_2 + ... + n_j = n = const \tag{2.7}$$

Here, n_j, μ_j represent the composition and the partial Gibbs' energy (or chemical potential) of the jth component,

$$\mu_j = \left(\frac{\Delta G}{\Delta n_j} \right)_{P,T,n'} \tag{2.8}$$

Essentially, μ_j is the contribution of the jth component toward the ΔG of heterogeneous system. Mathematically speaking, the partial Gibbs energy of a component is the slope of the total Gibbs energy of the system with respect to the amount of substance of interest.

Due to the Gibbs-Duhem equation, which states that under conditions of constant temperature and pressure

$$n_1 \Delta \mu_1 + n_2 \Delta \mu_2 + ... + n_j \Delta \mu_j = 0 \tag{2.9}$$

the Gibbs free energy is then given by:

$$\Delta G = \mu_1 \Delta n_1 + \mu_2 \Delta n_2 + ... + \mu_j \Delta n_j \tag{2.10}$$

2.6.5 Phase Diagram: Relationship between Pressure and Temperature

Physical changes in a system are often analyzed using the phase diagram. As illustrated in Fig-

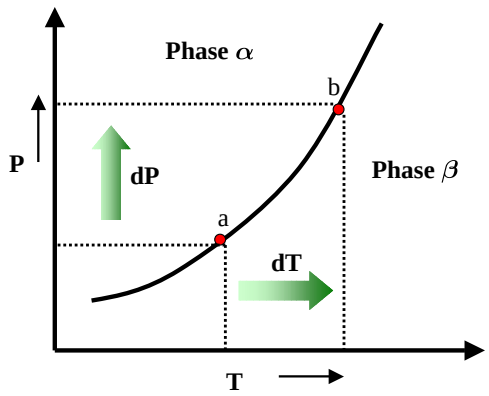


Figure 2.3: Let the system be in equilibrium at point (a). When pressure is applied to such a system, the equilibrium is disturbed. It can be restored by changing the temperature of the system. Thus the system moves to point (b)

ure 2.3, when pressure is applied to a system in which two phases are in equilibrium, the equilibrium is disturbed. It is then restored by changing the temperature. Thus, there is a relationship between pressure and temperature that ensures that they system remains in equilibrium as either parameter is changed and is given by the Clapeyron equation:

$$\frac{dP}{dT} = \frac{\Delta_{trs}S}{\Delta_{trs}V} \tag{2.11}$$

Here, $\Delta_{trs}S, \Delta_{trs}V$ are the entropy and volume changes associated with the phase transition.

2.6.6 Efficiency

Carnot efficiency η is defined as the maximum efficiency obtained when heat energy input to a system gets converted into work and is given by:

$$\eta = \frac{T_i - T_f}{T_i} \tag{2.12}$$

where T_i is the initial temperature and T_f is the final temperature.

2.7 Conclusions

In this chapter, we presented a review of the prior art concerning mesh segmentation, decimation, object representation and recognition. In the next chapter, we will present an algorithm for repairing triangulations containing a large number of topological singularities. In Chapter 4, we will present a new graph morphology based segmentation algorithm which can effectively deal with noisy point clouds. The proposed approach does not require an explicit pre-processing step for the removal of noisy artifacts. The sub-meshes resulting from the segmentation are then used to derive a two-level representation scheme. In Chapter 5, a graph theoretic mesh decimation algorithm is presented using the notion of the Laplacian of a graph and its energy. In Chapter 6, ideas from classical thermodynamics are applied to obtain robust correspondences for noisy scenes, for scenes with missing data and for scenes consisting of multiple parts.

Chapter 3

Conversion from Non-manifold to Manifold Surfaces

Three dimensional triangulations are a popular choice for representing underlying objects primarily because they provide easy solutions to a given application problem [13]. The applications span many research disciplines, from computer aided design and computer aided manufacturing to computer graphics and computer vision. Triangulations are non-manifold due to the presence of topological singularities caused by human or software induced 'bugs' [98]. Algorithms that strictly need manifold triangulations for processing can have undesirable results when used on non-manifold surfaces. Examples of typical algorithms that fail to work on non-manifold surfaces include the algorithms for vertex decimation, surface compression, surface smoothing and rapid prototyping [98]. For the purposes of reliable 3D object representation and recognition, a geometric modeling system must deal with such topological singularities to enable an efficient description of the underlying objects. In this dissertation, our particular interest is in 3D triangulations with a large number of topological singularities caused by the presence of singular edges, i.e., edges along which more than two triangles are incident. Further, we assume that the underlying surface is smoothly curving concave or convex. To address the problem, we propose a surface growing algorithm that uses topological as well as geometric characteristics to generate a

manifold surface.

The general approach for generating manifold surfaces from non-manifold triangulations involves the following steps:

1. **Decomposition to simpler parts**: Triangulations are 'cut' along the geometric elements (vertices, edges, faces etc.) where singularities occur [42, 98]. Such a decomposition results in singularity free components.

2. **Stitching**: Some of the components obtained in step (1) are grouped together at the geometric elements where singularities occur [98].

The algorithm presented in this chapter directly builds on the approach proposed in [98]. Guéziec et.al in [98] present a manifold surface conversion algorithm purely based on the topology. In their algorithm, the input triangulation is cut along singular edges. Stitching along the identified boundary edges results in a manifold surface. Also, the stitching process ensures that the boundary edges do not become singular. However, by their approach there can be many possible manifold surfaces, not all of which represent the underlying surface geometry. While following their overall approach of cutting and stitching, we employ additional geometric constraints for the selection of optimal triangles to stitch along the boundary edges. We have observed that such an approach, that of combining topology with geometry for stitching purposes, yields a manifold surface that is more representative of the underlying object geometry.

We now present an outline of our algorithm. To initiate the surface growing process, an arbitrary vertex is selected and a seed triangle incident on this vertex is determined based on the minimization of a cost criterion. The input non-manifold triangulation is then decomposed into unique triangular faces by cutting along all the edges present in the triangulation. The growing process primarily involves (1) identification of boundary edges on the grown surface (2) determination of optimal triangles to stitch along the boundary edges.

This chapter is organized as follows. In Sections 3.1 and 3.2, we provide the definitions and the necessary background required for understanding the proposed algorithm. The surface

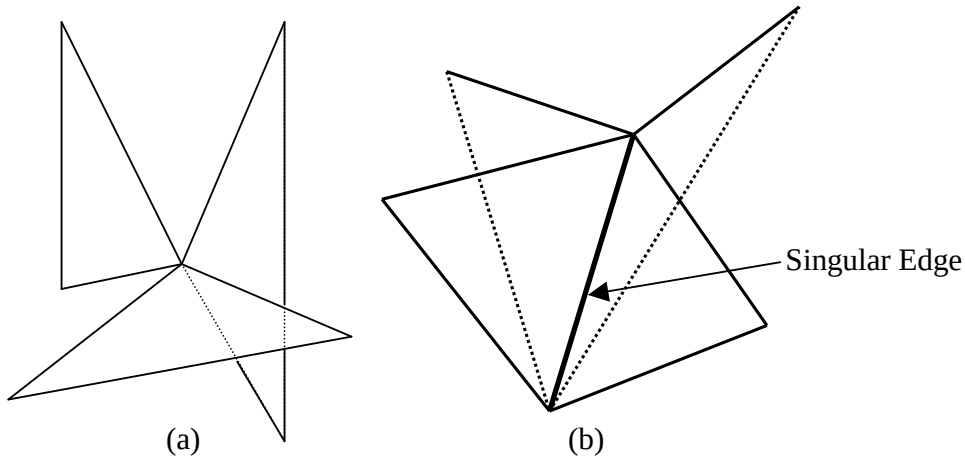


Figure 3.1: (a) Example of *lone* triangles (b) An example of a singular edge

growing algorithm is presented in Section 3.3. The results obtained by the algorithm are shown in Section 3.4.

3.1 Definitions

A 3D triangulation $T(V, F)$ is characterized by a set of vertices $V = \{v_1, v_2, ..., v_k\}$ and a set of triangular faces $F = \{f_{123}, f_{234}, ..., f_{ijk}\}$ such that a triangular face is an unordered subset of three distinct vertices. Every triangular face is said to be *incident* on its set of constituent vertices and corresponding edges [98]. A vertex with no incident triangular face is called a *lone* vertex and a triangular face is a *lone* triangle when there are no other triangular faces incident on any of its edges. Figure 3.1(a) provides an example of three lone triangles.

An edge is a *singular edge* if there are more than two triangular faces incident on that edge, else the edge is said to be *regular* [98]. The notion is pictorially represented in Figure 3.1(b). Both the vertices connected by a singular edge are necessarily singular [98]. However, there can be instances in a triangulation, when all the edges incident on a vertex are regular but the vertex on which they are incident is singular. Such a vertex is called an *isolated singular* vertex [98]. Isolated singular vertices are usually created due to holes in the triangulation. A regular edge in

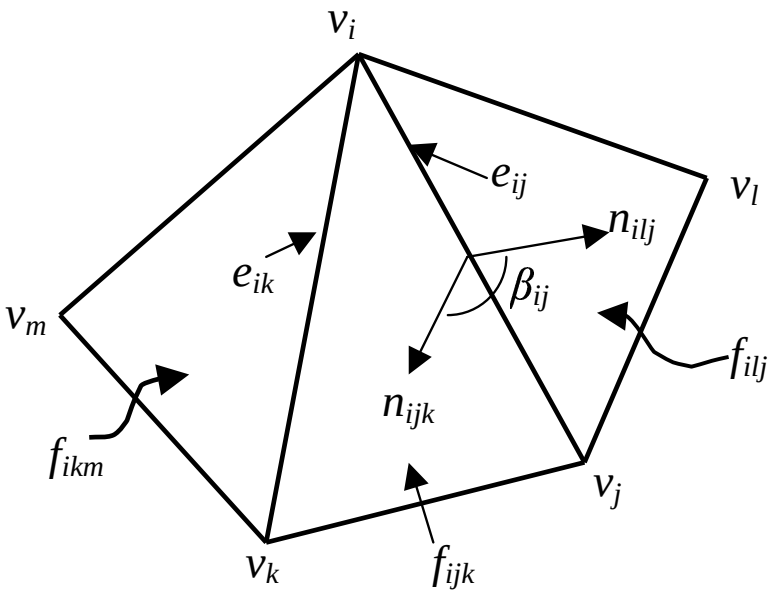


Figure 3.2: Definition of various parameters for the seed triangle computation.

a triangulation can be either a *boundary* edge or an *interior* edge. An edge with one and only one incident face is called a boundary edge, and is an interior edge otherwise. In a non-manifold triangulation either one of the following conditions are true:

- Presence of singular edges, which implies the presence of singular vertices
- Presence of isolated singular vertices

The algorithm proposed in this chapter is primarily useful in treating triangulations with a significantly large number of singular edges.

3.2 Shape Computation Basics

Consider Figure 3.2. A vertex in a three dimensional triangulation is represented as a triplet of coordinates i.e., a vertex v_i is denoted by $v_i = \{x_i, y_i, z_i\}$. A triangle is represented as a triplet of distinct vertices and is given by $f_{ijk} = \{v_i, v_j, v_k\}$.

With respect to vertex v_i present in triangle f_{ijk}, we define two edge vectors e_{ij} and e_{ik} as:

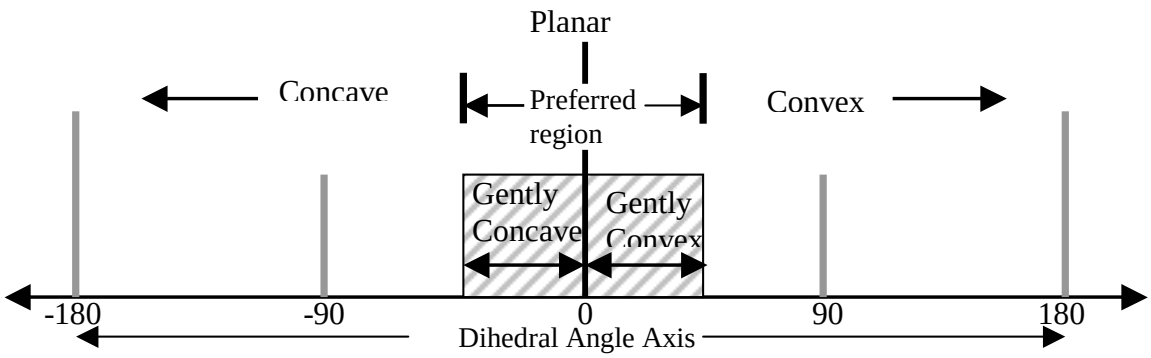


Figure 3.3: Shape interpretation of a surface based on dihedral angles

$e_{ij} = v_j - v_i$ and $e_{ik} = v_k - v_i$. The normal n_{ijk} to the triangle is defined as the cross product of the two edge vectors :

$$n_{ijk} = e_{ij} \times e_{ik} \tag{3.1}$$

The area of the triangle f_{ijk} is then defined as:

$$A(f_{ijk}) = \|e_{ij} \times e_{ik}\|/2 \tag{3.2}$$

The dihedral angle β_{ij} is defined as the angle between the normals of two adjacent triangles f_{ijk}, f_{ilj}, computed along the edge e_{ij}. Thus,

$$\beta_{ij} = cos^{-1}(n_{ijk} \cdot n_{ilj}) \tag{3.3}$$

The dihedral angle varies between $-180°$ and $+180°$.

Figure 3.3 shows various shape interpretations of a triangulation in terms of the dihedral angle. While positive angles represent a locally convex surface, negative angles represent a locally concave surface and zero dihedral angle implies a locally planar surface. A dihedral angle of $180°$ implies that the underlying surface edge is a sharp edge whereas a dihedral angle of $180°$ occurs when the surface folds back on itself [13]. To construct a smooth surface triangulation, the adjacent triangular faces should have small magnitude dihedral angles, positive or negative,

which are indicative of gently curving convex or concave surfaces respectively. Since the objects of interest are free-form,the need for smooth surface triangulations is justified.

3.3 The Algorithm

3.3.1 Assumptions

1. **Number of singularities**: We assume that the input triangulation has a significantly large number of topological singularities for removal.

2. **Vertex Accessibility**: We assume that every vertex is *reachable* from every other other vertex by walking a certain path along a set of triangular faces. Therefore, we do not deal with triangulations that have lone vertices.

3. **Nature of the underlying surface**: The underlying surface is assumed to be smoothly curving concave or convex.

4. **Nature of Point Cloud**: We do not assume sparse point clouds. This allows for the selection of an arbitrary seed vertex during the region growing process.

3.3.2 Seed Triangle Selection: Formulation of the Objective Function

An arbitrary vertex in the input triangulation is selected as the seed vertex. The next step in our algorithm is the determination of a *seed* triangle from a given three dimensional triangulation. Let $\{f_{ijk}\} \subset F$ be the set of triangular faces incident on an arbitrary seed vertex v_i. From a geometric perspective, a seed triangle should have the following characteristics: (i) it should be

as regular as possible (ii) it should be as similar as possible to the adjacent triangles, which in a sense, implies that it should be representative of other triangles in the neighborhood. . In order to achieve the above criteria, we select the seed triangle $\hat{f} \in \{f_{ijk}\}$, as the one that minimizes the total cost C_{ijk} defined below. Suppose, there are *I* triangular faces incident on the seed vertex v_i . Let the two edges of the triangle f_{ijk} that are incident on vertex v_i be e_{ij} and e_{ik}. Further, suppose, there are *J* adjacent triangles along the edge e_{ij} of triangle f_{ijk} and *K* adjacent triangles along the edge e_{ik}. With respect to each of the adjacent triangles is associated a dihedral angle, computed along e_{ij} or e_{ik} of the triangle f_{ijk}. The total cost C_{ijk} associated with triangle f_{ijk} (incident on the seed vertex v_i) is defined by (3.4).

$$C_{ijk} = \|e_{ij}\| + \|e_{ik}\| + |\cos(\hat{\beta_{ij}})\hat{A_{ij}} - A(f_{ijk})| + |\cos(\hat{\beta_{ik}})\hat{A_{ik}} - A(f_{ijk})| \tag{3.4}$$

Here, $A(f_{ijk})$ is the area of the triangular face f_{ijk}, $\hat{\beta_{ij}}$ is the optimal dihedral angle between f_{ijk} and a *similar adjacent* triangular face along edge e_{ij} and $\hat{A_{ij}}$ is the area of the adjacent triangle that results in the dihedral angle $\hat{\beta_{ij}}$. Similarly, $\hat{\beta_{ik}}$ is the optimal dihedral angle between f_{ijk} and a similar adjacent triangular face along edge e_{ik} and $\hat{A_{ik}}$ is the area of the adjacent triangle that results in the optimal $\hat{\beta_{ik}}$. Therefore, $\hat{\beta_{ij}}$ and $\hat{\beta_{ik}}$ are defined as:

$$\begin{aligned} \hat{\beta_{ij}} &= \min_{J} |\beta_{ij}| \\ \hat{\beta_{ik}} &= \min_{K} |\beta_{ik}| \end{aligned} \tag{3.5}$$

Rearranging terms in (3.4), for every triangle, we can associate the total cost in terms of the cost along its two constituent edges that are incident on the seed vertex v_i. This is defined as :

$$C_{ijk} = \underbrace{\|e_{ij}\| + |\cos(\hat{\beta_{ij}})\hat{A_{ij}} - A(f_{ijk})|}_{cost(e_{ij})} + \underbrace{\|e_{ik}\| + |\cos(\hat{\beta_{ik}})\hat{A_{ik}} - A(f_{ijk})|}_{cost(e_{ik})} \tag{3.6}$$

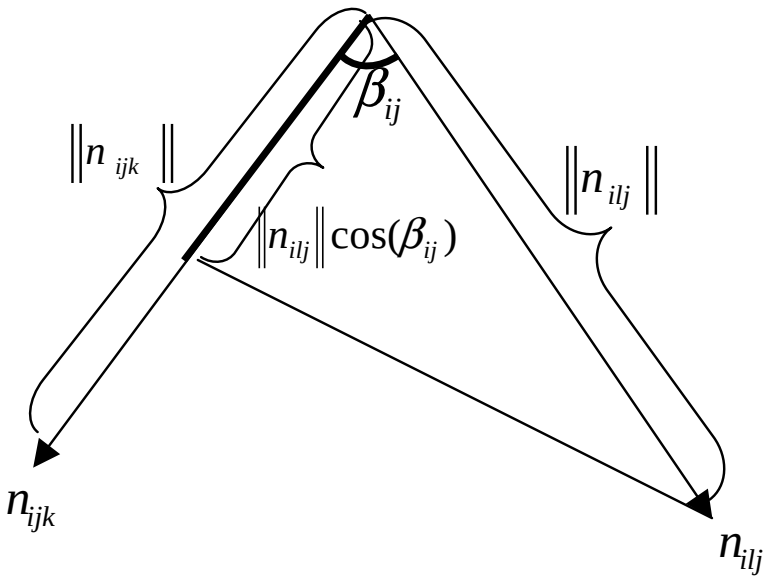


Figure 3.4: Geometric interpretation of the cost criterion

Among all the triangles incident on the seed vertex v_i, the seed triangle is the selected to be the one that minimizes the cost C_{ijk}.

3.3.3 A Geometric Interpretation of the Cost Criterion

The cost function defined in (3.6) reflects the aforementioned criteria. The norm of the edges ensures the regularity of the selected seed triangle. To explain the significance of the $|\cos(\hat{\beta}_{ij})\hat{A}_{ij} - A(f_{ijk})|$ term, consider Figures 3.2 and 3.4.

For a certain triangle to be representative of the other triangles in the neighborhood, the areas of the two triangles in consideration must be close to each other. Mathematically, this means that, for two adjacent triangles, incident on edge e_{ij},

$$\begin{aligned} Area(f_{ijk}) &\approx Area(f_{ilj}) \\ \Rightarrow Area(f_{ijk}) &= \alpha \cdot Area(f_{ilj}), \alpha \approx 1 \\ \Rightarrow \tfrac{1}{2}\|e_{ij} \times e_{ik}\| &= \tfrac{\alpha}{2}\|e_{il} \times e_{ij}\| \end{aligned} \tag{3.7}$$

From (3.1) and (3.7), we have:

$$\alpha \|n_{ilj}\| = \|n_{ijk}\| \tag{3.8}$$

From Figure 3.4, $\|n_{ilj}\| \cos(\beta_{ij})$ is the length of n_{ilj} as a projection. One can approximate the length of this projection with the length of n_{ijk}, for small values of β. Therefore, for the areas of the adjacent triangles to be similar $\cos(\beta_{ij}) = \alpha \approx 1$. Normally, this is true when β_{ij} is in the *preferred region*.

From the cost metric in (3.6), we can conclude that if two triangles have similar areas and further, if the underlying surface is smoothly curving concave ($\beta_{ij} < 0$) or convex ($\beta_{ij} > 0$) , then $|\cos(\hat{\beta_{ij}})\hat{A_{ij}} - A(f_{ijk})| \approx 0$, thereby satisfying one of the more important criteria for the selection of the seed triangle.

3.3.4 The Algorithm: Greedy Strategy

After selecting the seed triangle for surface growing purposes, the input non-manifold surface triangulation is decomposed into the constituent triangles. Stitching is essentially a grouping process [98] in which an optimal triangle is selected to *merge* along the boundary edges identified on the grown surface. Stitching for surface growing involves the following steps:

1. Identification of boundary edges on the grown surface
2. Constraint-based selection of triangular faces to stitch along the boundary edges identified in step (1)
3. Merging the triangular faces selected in step (2) along the corresponding boundary edges.
4. Updating the boundary edge and the vertex lists after stitching.

Figure 3.5 illustrates the stitching process. Clearly, the key elements of the stitching strategy that will ensure that the resulting surface is manifold are: (a) identification of the boundary edges (b) Optimal triangular faces to stitch along the identified boundary edges.

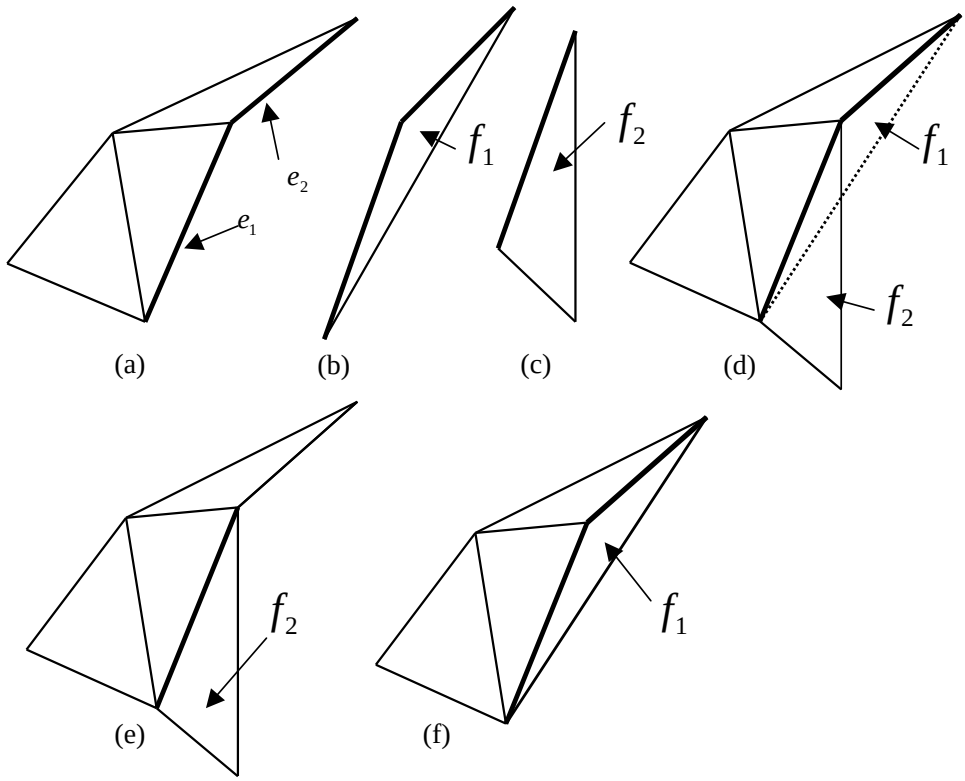


Figure 3.5: (a) Grown surface with boundary edges e_1 and e_2 (b), (c) Two optimal triangles identified for stitching (d)A surface that can never be generated by the surface growing algorithm (e),(f) Two possible stitches

Identification of boundary edges on a grown surface: The edges that constitute the triangular faces present on the grown surface can be classified either as interior edges or as boundary edges. The edges with exactly one incident triangular face are termed as boundary edges. Edges with at most two incident triangular faces are termed as regular interior edges. Stitching is possible only along boundary edges. In our algorithm, stitching along interior edges is clearly not allowed for it will render the resulting surface as non-manifold.

Selection of Optimal triangle for stitching: Having selected the seed triangle, other optimal triangles (for stitching along identified boundary edges) for the surface growing process are chosen from a set of potential candidates by employing additional constraints. Specifically, an *adjacent* triangular face which, if stitched, would minimize the error E_{abc} computed as the magnitude of the dihedral angle along a boundary edge e_{bc}, is selected as the best candidate. The cost

E_{abc} associated with a triangular face f_{abc} is given by:

$$E_{abc} = |\beta_{bc}| \tag{3.9}$$

Among the T triangles incident on edge e_{bc}, the one that minimizes the error computed above, is selected for stitching along the edge. The rest of the triangles that were evaluated for stitching along the edge e_{bc}, are now discarded from the list of triangles that are available for subsequent processing. Observe that the cost function formulated for the selection of the seed triangle is quite different from the one proposed for stitching along the grown surface.

After stitching a triangle along a boundary edge, the edge is subsequently labeled as interior. Further, when all the edges incident on a vertex are labeled as interior edges, the vertex consequently becomes an interior vertex. If however, there is at least one boundary edge incident on a vertex, the vertex remains a boundary vertex. The process of selection of the *optimal* triangular faces (for stitching) followed by stitching along the corresponding boundary edges is continued until all the vertices are labeled as interior vertices. Such a growing process ensures that isolated singular vertices are not created. Further,it guarantees that the resulting triangulation is manifold.

3.4 Experiments and Results

Figure 3.6 presents some results obtained by our algorithm. The non-manifold surfaces of various objects are shown in the left-hand column. These triangulations were obtained by processing the point clouds of the objects using commercially available mesh generation software. The singularity test [98] indicates that these triangulations contain a large number of singular edges. The results obtained by the surface growing algorithm are shown in the right hand column. The timing performance of the algorithm is indicated in Table 3.1. The approach can be used to repair non-manifold surfaces off-line.

Non-manifold surfaces

Manifold Surfaces via region growing process

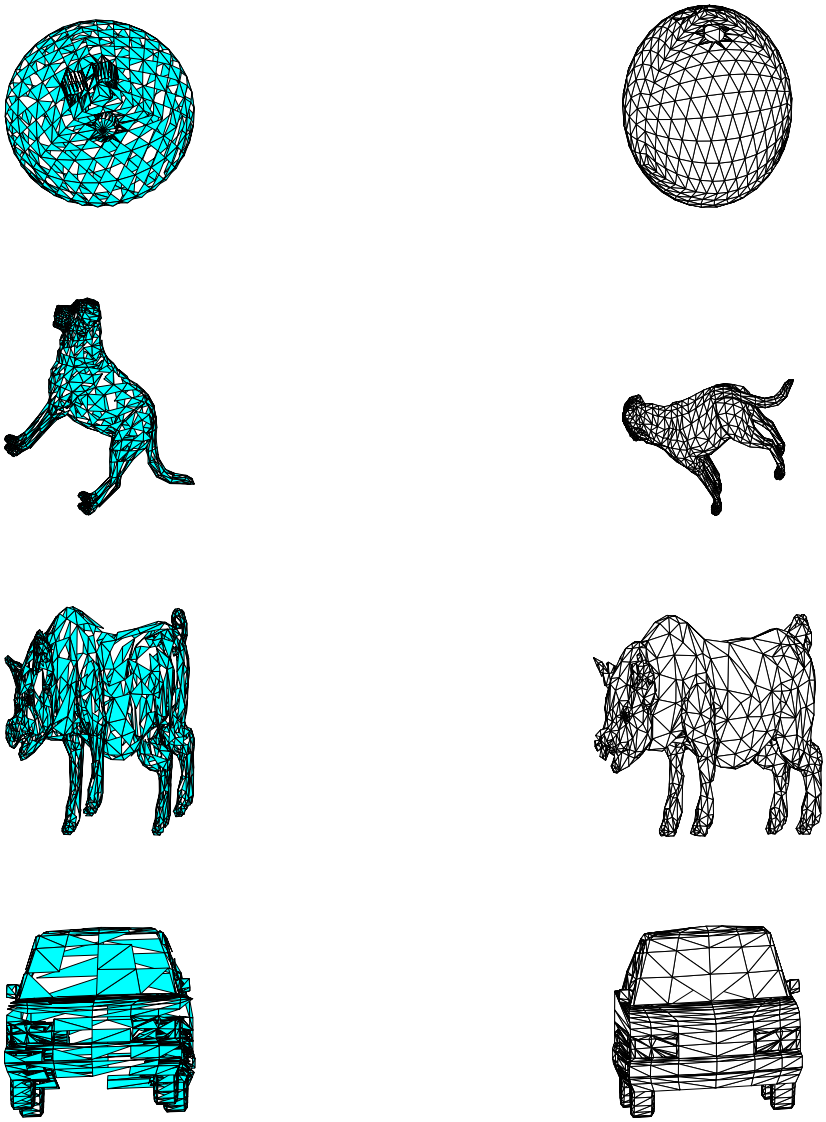


Figure 3.6: Manifold surface conversion

object	Number of vertices	Time (in minutes) for conversion to manifold surface
Ball	2932	3.9
Dog	4305	6.3
Pig	4332	6.35
Car	7401	10.4

Table 3.1: Timing Performance of the proposed mesh repair algorithm on various data sets

3.5 Conclusions

In this chapter, a surface growing algorithm is described to convert a 3D non-manifold triangulation into a manifold surface mesh. The surface hence generated is truly reflective of the underlying object's geometry. The results of the approach have been used in Chapter 4 for segmentation and representation. In chapter 5, the manifold meshes have been used for decimation purposes. Due to the propagating nature of the algorithm and due to the constraints employed on the triangles used for stitching, the algorithm generates a smoothly triangulated surface, free of any self-intersections. The algorithm is particularly useful, when a given triangulation has a significantly large number of singularities to be corrected. The method, however, does not address issues like filling gaps in disconnected triangulations. It may not be suitable if the underlying object has sharp concave or convex regions. Otherwise, the method is general and handles singularities without user intervention.

Chapter 4

Mesh Segmentation for Object Representation

Toward deriving an efficient representation scheme for free-form objects which are characterized by point cloud data, this chapter details a new approach for segmentation of surface meshes constructed over the point clouds. While we make very few restrictive assumptions about the shapes of the underlying 3D objects, we do not consider statistically defined shapes (such as textures, foams), infinitely detailed objects that are best described using fractals and non-orientable surfaces (such as Klein bottles, Moebius strips). Typical free-form objects of interest include vehicles, industrial assembly parts and animals.

Although existing free-form representations [55, 69] provide an efficient description of 3D objects to enable fast and accurate retrieval during recognition, they implicitly assume a uniform sampling of the point clouds. In practice, however, due to noise or sensor inaccuracies, point clouds are rarely uniformly sampled. Representations derived from non-uniformly sampled point clouds do not provide concise descriptions of underlying objects for the following reasons (1) shape descriptors computed from noisy point cloud data are often inaccurate, (2) segmentation of meshes constructed over such point clouds result in numerous fragments as opposed to sub-meshes corresponding to the physical parts in the underlying object. This work addresses the

challenge of segmenting a 3D surface mesh into physical parts [52] without making any assumptions about the clouds being sampled uniformly. Segmentation results are then used to derive a compact representation of the underlying object.

It is known that humans disambiguate objects in terms of their shapes. Shape is the visible or perceived form of an object that distinguishes a cylinder from a sphere, a human from a dog. The shape of a rigid object is independent of its position and orientation in space. In fact, as pointed out by Koenderink [19, 24], shape is independent of *scale*; a ping pong ball and a tennis ball have different radii, but are both described as spherically-shaped. Thus, shape plays an important role in differentiating one object from another. Most man-made objects are not composed of a single shape but are usually made of regions of different shapes. Therefore, an object can be effectively characterized by partitioning it into different shape categories. It may be noted that shape alone does not constitute a complete description of an object. One factor that distinguishes a ping-pong ball from a tennis ball is the scale or curvedness [19, 24]. Scale describes the amount of curvature present in an object and plays a conspicuous role in recognition [55]. We provide mathematical descriptions of shape and scale in the following section. Unless a representation scheme provides information about the orientation of an object or its parts in the 3D space, it will be extremely difficult to perform recognition of scenes, which primarily consist of the rotated versions of the stored object models. Hence, the orientation information complements the shape and scale characterization of an object.

In this work, an input point cloud is triangulated using a commercially available software, Points2polys [126]. For the efficiency and the ease of computation of the shape descriptors, manifold input triangulations [98] are considered. This means that there exist at most two triangles incident along any edge in the mesh. A non-manifold triangulation can be converted into a manifold surface mesh using the approach described in Chapter 3 or using other techniques [98]. Subsequently, the shape descriptors namely, the shape index and curvedness, are computed at every vertex in the triangulation, in order to capture the shape and the scale information

about the 3D object. The surface mesh is partitioned into multiple, disjoint sub-meshes using the graph-morphology based segmentation algorithm, described in Section 4.3.2. Curvedness serves as a similarity metric for segmentation purposes. The segmentation results are used to derive a two-tier representation wherein, at the coarser scale, adjacency relationships between disjoint sub-meshes are established by constructing a super-graph over these sub-meshes. At the finer scale, the orientation information about the object is captured by mapping the normals of each of the disjoint sub-meshes onto a unit sphere.

Specific objectives that are accomplished in this chapter are:

- description of a framework to deal with noisy point clouds without performing an explicit mesh smoothing operation,
- formulation of a new sub-mesh extraction algorithm that combines graph morphology and signal filtering ideas,
- design of an approach for the adaptive selection of curvedness thresholds that leads to disjoint sub-meshes that seem to match the human visual segmentation of the underlying object, and,
- design of a two-tier representation scheme for the description of free-form objects.

This chapter is organized as follows. In Section 4.1, we provide the background required for computation of shape descriptors at every vertex in the input triangular mesh. Section 4.2 presents the definitions and properties of various morphology based notions, all of which are required for segmentation purposes. Sections 4.3 and 4.4 provide a complete description of the algorithm and highlights its capabilities. In Section 4.5, we design a two-tier representation scheme for the description of rigid free-form objects. In Section 4.6, the proposed segmentation algorithm is compared with an existing state-of-the-art approach and results are provided to demonstrate the effectiveness of the approach.

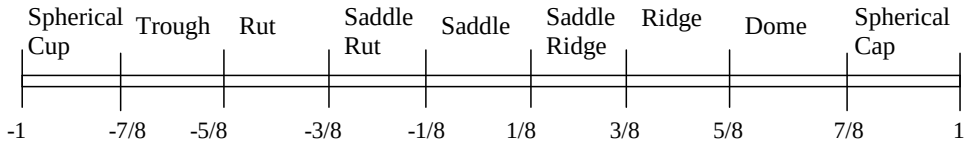


Figure 4.1: Shape Index scale is divided into nine shape categories. The two extreme shape thresholds $t_0 = -1, t_9 = +1$. The other eight thresholds are indicated on the scale. These thresholds are used to identify shape clusters present in a sub-mesh

4.1 Definitions and Notations

4.1.1 Shape Descriptor: Curvedness

Curvedness which is also known as the bending energy [19, 55], measures the intensity of the surface curvature and describes how gently or strongly curved a surface is. Mathematically, it is defined as:

$$C_v = \sqrt{(\kappa_{max}^2(v) + \kappa_{min}^2(v))/2} \tag{4.1}$$

where $\kappa_{max}(v), \kappa_{min}(v)$ are the principal curvatures of the surface at vertex v.

The shape index provides the quantitative definition of the shape of a surface at a vertex *v* and is defined as [19, 24]:

$$S(v) = -\frac{2}{\pi}\tan^{-1}\frac{\kappa_{max}(v) + \kappa_{min}(v)}{\kappa_{max}(v) - \kappa_{min}(v)} \tag{4.2}$$

As illustrated in Figure 4.1, in [19] a surface is classified into nine categories based on shape index. Vertices on a planar surface (with indeterminate shape index) are assigned a value outside the interval [-1,1].

The principal curvatures $\kappa_{max}, \kappa_{min}$ are defined as:

$$\kappa_{max}(v) = H(v) + \sqrt{H^2(v) - K(v)}; \kappa_{min}(v) = H(v) - \sqrt{H^2(v) - K(v)} \tag{4.3}$$

The mean curvature and the Gaussian curvatures denoted by $H(v), K(v)$ respectively, are computed by considering the triangular mesh as a piece-wise linear approximation of an unknown

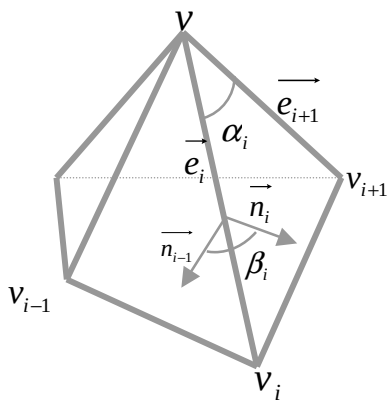


Figure 4.2: Definition of various parameters with respect to the triangular faces incident on vertex *v*. α_i is the angle subtended by the triangular face f_i at the vertex *v* and is computed as the angle between the corresponding edge vectors e_i and e_{i+1}; β_i is the dihedral angle between adjacent triangular faces and is computed as the angle between the corresponding normals.

smooth surface [78]. Therefore,

$$K(v) = \frac{2\pi - \sum_{i=1}^{k} \alpha_i}{A/3} \qquad (4.4)$$
$$H(v) = \frac{\frac{1}{4}\sum m_j \|e_j\|}{A/3}$$

where,

$$m_j = \begin{cases} \beta_j & \text{if edge } e_j \text{ is convex} \\ 0 & \text{if edge } e_j \text{ is planar} \\ -\beta_j & \text{if edge } e_j \text{ is concave} \end{cases} \qquad (4.5)$$

$A = \sum_{i=1}^{k} f_i$ is the sum of areas of k triangular faces incident on the vertex *v*, α_i denotes the angle subtended by a triangular face f_i at vertex *v*. β_j is the dihedral angle between two adjacent triangular faces f_j and f_{j+1} and is computed as the angle between the corresponding normals. Here, $||e_j||$ is the Euclidean norm. These quantities are illustrated in Figure 4.2.

In the proposed segmentation algorithm described in Section 4.3, curvedness will be used as the shape feature to partition an input surface mesh into disjoint sub-meshes.

4.1.2 Graphs

In this work, an input mesh is defined as an attributed graph $G[(V, C), E]$, where $V(G) = \{v_1, v_2, ..., v_n\}$ is the set of vertices comprising the mesh, $C(G) = \{C_{v1}, C_{v2}, ..., C_{vn}\}$ is the set of curvedness values associated with the vertices in the mesh and $E(G)$ is a set of edges connecting the vertices in $V(G)$. Vertices v_i and v_j are *adjacent* and are neighbors if there exists an edge e_{ij} connecting them [34]. The neighborhood $N(v_i)$ of a vertex v_i consists of a set of vertices that are adjacent to vertex v_i.

Given the graph G and a threshold interval $[t_i, t_{i+1})$, $t_i, t_{i+1} \in [C_{min}, C_{max}]$, where C_{min} and C_{max} are the minimum and the maximum curvedness values respectively, *a maximally connected attributed subgraph* (*MCASG*) Y is defined as:

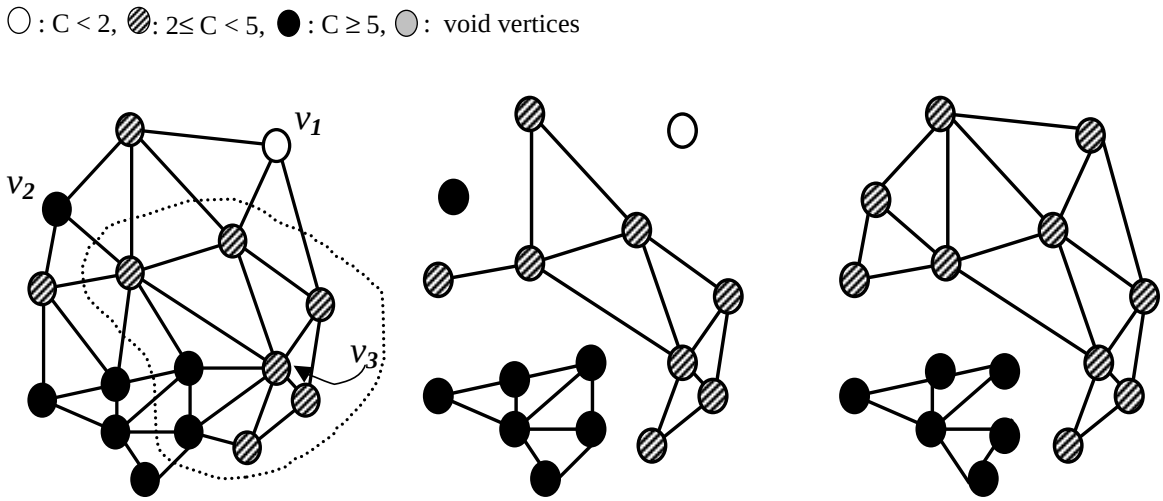
1. $V(Y) \subset V(G)$
2. $E(Y) = E(G) \cap (V(Y) \times V(Y))$ i.e., $E(Y)$ contains edges from the naturally generated edge set,
3. $C_{v_i}, C_{v_j} \in [t_i, t_{i+1}), \forall v_i, v_j \in V(Y)$
4. there exists a path *p* from v_i to v_j containing distinct vertices $v_0, v_1, ..., v_m \in V(Y)$, such that condition (3) holds true for every pair of vertices along the path *p*.

The details of the MCASG extraction algorithm are presented in Section 4.3.

In the following section we explain the need for new graph-based morphological operations and provide an overview of the proposed algorithm's characteristics.

4.2 Graph Morphology-based Segmentation: Overview

Non-structured graph morphology as well as soft morphology [28, 27, 14, 36, 39, 47, 87, 17] have been extensively applied in 2D image processing. In 2D morphology, a connected component is



(a) Input mesh *G[(V,C),E]* (b) Output1: Four Disjoint MCASGs (c) Output2: Two Disjoint MCASGs

Figure 4.3: The implication of not transforming the geometric properties of outlier vertices is shown by way of Output1. Output2 shown in (c) is obtained by our algorithm. The proposed algorithm performs graph dilation by identifying and morphologically filtering outlier vertices such as v_1, v_2. The outliers in the dilated graph are discarded formally by performing attributed graph matching.

extracted by the iterative application of the dilation and intersection operators to the input image [100]. For segmentation of meshes, this idea needs to be applied carefully because (1) extraction of multiple MCASGs requires that non-structured dilation [27, 14] be performed in restricted regions of the input mesh, and (2) the intersection operator must be specialized to handle graphs as opposed to an array of pixels.

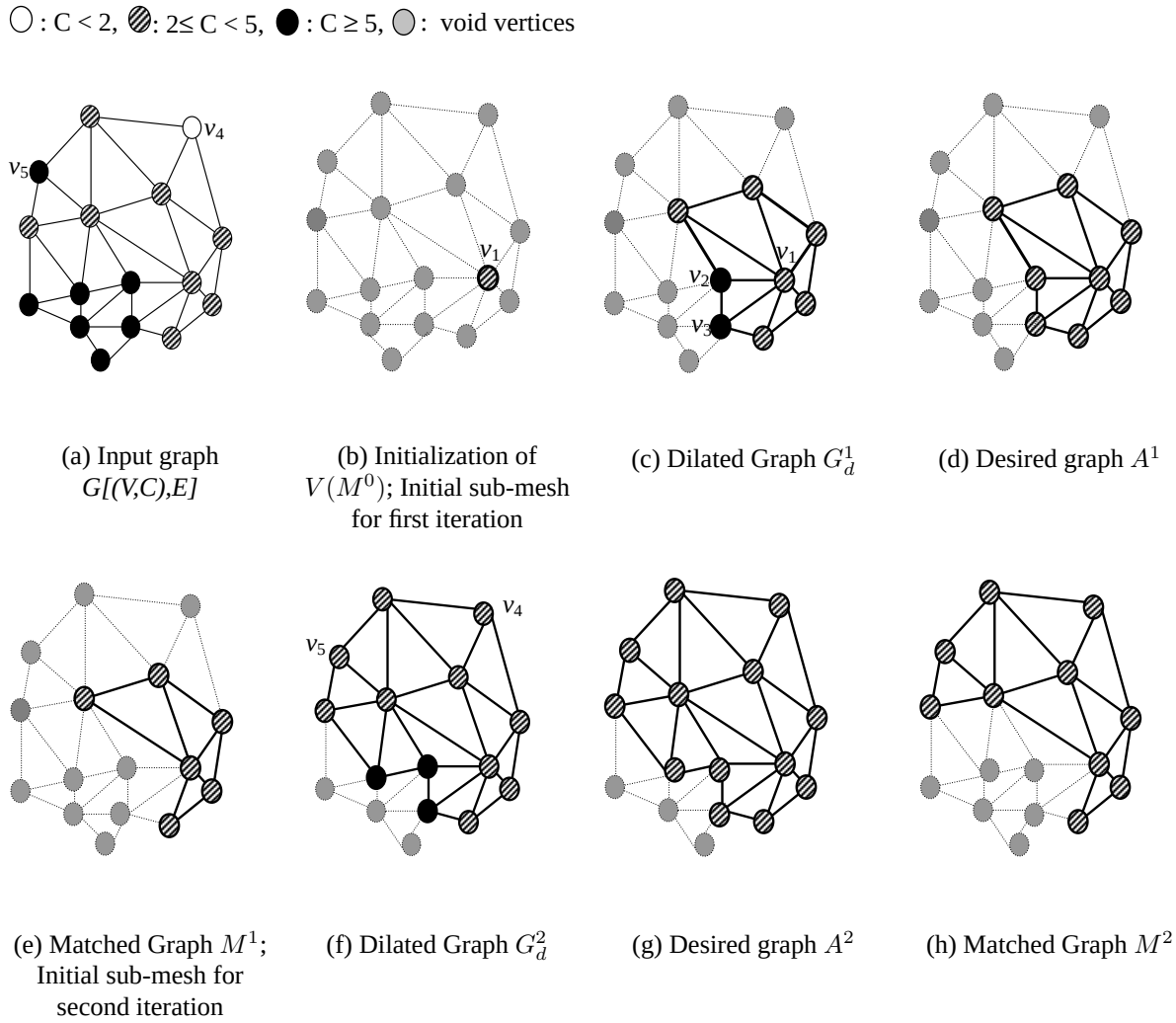
Figure 4.3 illustrates the need for a new graph theoretic formulation. The color on the vertices indicate the range of curvedness values assigned to them. The segmentation of *G* using the known algorithms will result in four MCASGs as shown in Figure 4.3(b). Such partitions are not acceptable for our purposes because the MCASGs correspond to small, discontinuous regions (possibly arising from the noisy data). On the other hand, the proposed morphology-based

processing of the graph *G* forces the outlier vertices v_1 and v_2 in Figure 4.3(a) to behave like their neighbors, thereby resulting in two MCASGs as indicated in Figure 4.3(c). The proposed algorithm has the following components:

- Adaptive threshold selection: Sub-meshes are obtained using adaptively determined curvedness threshold intervals. The details of the threshold selection process is presented in Section 4.3.1.

- Sub-mesh Growing: The MCASG extraction process is initialized by identifying a vertex from the input mesh with curvedness value in the specified threshold interval. For example, given the input mesh shown in Figure 4.4(a) and the curvedness threshold interval $[2, 5)$, the segmentation initialization is shown in Figure 4.4(b). At any iteration, the initial sub-mesh is expanded by graph dilation, which also implicitly identifies and filters the outliers in the expanded sub-mesh. Specifically, this approach exploits the idea that the geometric behavior of a vertex is influenced by its neighbors, so that an outlier vertex is transformed to be a part of the MCASG by replacing its curvedness value by the median computed over the curvedness values of its neighbors. The resulting expanded sub-mesh is scooped out of the input mesh to form the *dilated graph* for the iteration. As an example, graph dilation around the initialized vertex (Figure 4.4(b)) followed by the extraction of the expanded sub-mesh results in the dilated graph for the first iteration, as shown in Figure 4.4(c). Observe that the morphological filtering during this iteration has not transformed the outlier neighbors (v_2, v_3) of the initialization vertex v_1. On the other hand, during the second iteration, the dilation of M^1 morphologically filters the outliers v_4, v_5 (Figure 4.4(f)).

- Attributed Graph Matching to discard outliers: When only a proper subset of the vertices of the dilated graph have the curvedness values in the desired curvedness interval then, the outlier vertices are discarded formally by performing attributed graph matching with the *desired* graph. The motivation for this is drawn from 2D image processing, wherein



(a) Input graph *G[(V,C),E]*

(b) Initialization of $V(M^0)$; Initial sub-mesh for first iteration

(c) Dilated Graph G_d^1

(d) Desired graph A^1

(e) Matched Graph M^1; Initial sub-mesh for second iteration

(f) Dilated Graph G_d^2

(g) Desired graph A^2

(h) Matched Graph M^2

Figure 4.4: The steps involved in the extraction of two disjoint MCASGs for the given input mesh, G. The curvedness thresholds are specified as before.

a connected component is extracted through an iterative process of dilation and intersection [100]. These ideas are illustrated using Figure 4.4(c),(d) and (e). The matched graph, shown in Figure 4.4(e) is used as the sub-mesh for expansion during the second iteration. The resulting MCASG is shown in Figure 4.4(h).

In the following section, we formally describe the basic segmentation algorithm and then extend the approach for the extraction of multiple, disjoint MCASGs.

4.3 Algorithm for Extraction of MCASG

4.3.1 Adaptive Selection of Thresholds

The proposed segmentation algorithm is driven by the assumed knowledge of a pair of curvedness thresholds $[t_i, t_{i+1})$, that identifies the range of curvedness values allowed for the vertices in a MCASG. A threshold pair corresponds to the representative curvedness values for a MCASG. The motivation for the threshold selection process described here is derived from k-means clustering [93] and histograms [100]. For our problem, the selection of cluster centers resulting from a straightforward application of the k-means algorithm to the set of curvedness values does not result in the desired MCASGs. Also, the optimal number of classes needs to be specified. We have found that the use of curvedness histogram peaks as thresholds leads to over-segmentation. Such over-segmentations can be avoided by selecting a curvedness value (threshold) that is close to the identified peak.

We propose a hierarchical threshold determination technique based on the sub-bin processing of the histograms. Specifically, at a level i, we construct a curvedness histogram with an optimal bin width over the set of curvedness values in the interval $[t_i, C_{max}]$. Here t_i is the minimum of the N_i curvedness values available for processing at level i. When $i = 1$, $t_i = C_{min}$. Next, we identify the *first bin* with an appropriate peak i.e., the bin with at least 10% of the total number of

vertices in the input mesh. Sub-bin processing of this bin involves (1) partitioning of the curvedness values in the bin into two classes using k-means (2) assigning the cluster mean that is closer to the bin center as the threshold t_{i+1} for the extraction of the MCASG in the curvedness interval $[t_i, t_{i+1})$. The repetition of this process for $i = 2, 3, ...$ results in the determination of all the curvedness intervals $[t_i, t_{i+1})$ that are required for the extraction of the corresponding MCASGs. The threshold determination process is illustrated in Figure 4.5 and is formally described below.

For $i = 1, 2, ...$(representing various levels in the hierarchy), the selection of a threshold pair $[t_i, t_{i+1})$, involves

1. *Construction of Curvedness Histogram*: At a level i, a histogram is constructed over the curvedness values in the interval $[t_i, C_{max}]$. The set of curvedness values is partitioned into m *bins* such that the jth bin is the half-open interval $[t_i + (j-1)W_i, t_i + jW_i)$. Here, W_i is the optimal histogram bin width [6] computed as:

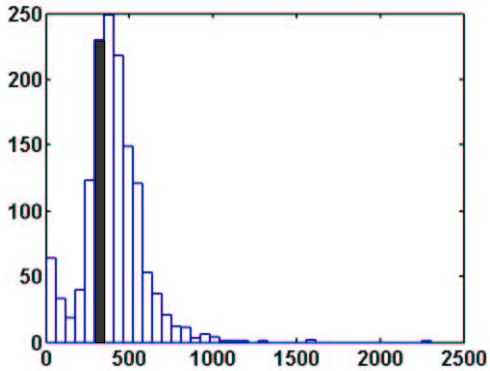
$$W_i = 3.49\sigma_i N_i^{-1/3} \tag{4.6}$$

   where σ_i is the standard deviation of the N_i curvedness values. The number of vertices whose curvedness value falls in the jth bin is given by $n_j = \sum_1^{N_i} \chi_k(C_{v_i})$ where χ is the characteristic function of the jth bin:

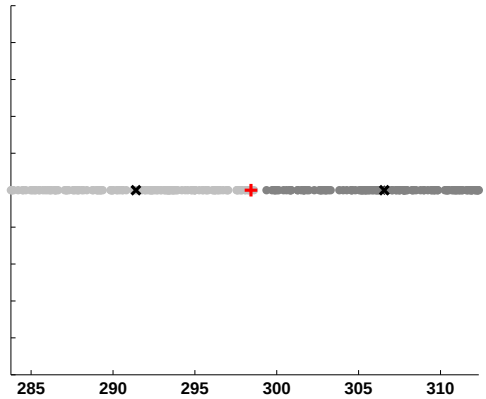
$$\chi_k(x) = \begin{cases} 1 & (t_i + (j-1)W_i) \leq x < (t_i + jW_i) \\ 0 & otherwise \end{cases} \tag{4.7}$$

   In Figure 4.5, for $i = 1$, the histogram is constructed by using *all* the curvedness values, whereas for $i = 2$ and $i = 3$, the corresponding histograms are constructed over the curvedness values in the intervals $[292.04, C_{max}]$ and $[382.84, C_{max}]$ respectively.

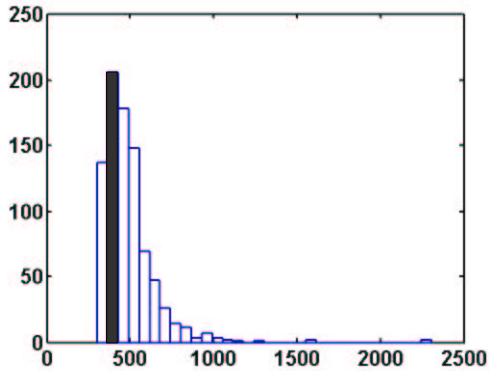
2. *Identification of the first bin of interest:* The kth bin is identified as the *first bin of interest*



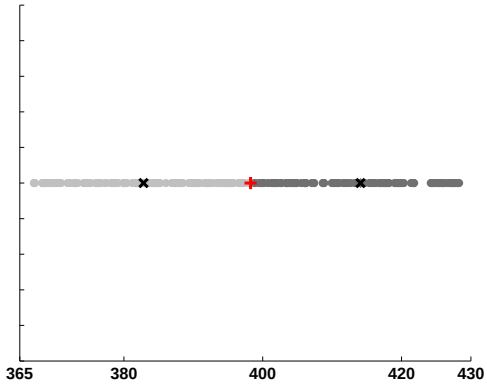
(a)$i = 1, W_1 = 56.02, k = 6, t_1 = 0$



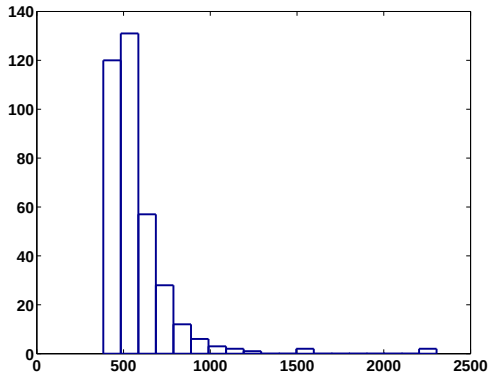
(b) $cen_1 = 292.04$, $cen_2 = 306.84$, $t_2 = 292.04$



(c) $i = 2, W_2 = 57.8, k = 2, t_2 = 292.04$



(d) $cen_1 = 382.84$, $cen_2 = 414.09$, $t_3 = 382.84$



(e) $i = 3, W_3 = 93.9, t_3 = 382.84$

Figure 4.5: Selection of curvedness threshold for the simplified horse consisting of 1548 vertices. The gray colored bin in (a) and (c) indicates the first bin of appropriate peakiness. All curvedness values in this bin are clustered into two classes using k-means with cluster centers cen_1 and cen_2. In (b) and (d) 'x' and '+' correspond to the cluster centers and the bin center respectively. The cluster center closest to the bin center is assigned as the threshold t_{i+1}

(a) Curvedness Interval [0 292.04) (b) Curvedness Interval [292.04,382.84) (c) Curvedness Interval [382.84,2304.4]

Figure 4.6: Segmentation of the simplified [23] horse with 1548 vertices into MCASGs corresponding to the threshold intervals shown in Figure 4.5

where $k = \min\{j \mid n_j \geq 10\%N\}$. In Figure 4.5(a), $k = 6$, as indicated by the gray colored bin.

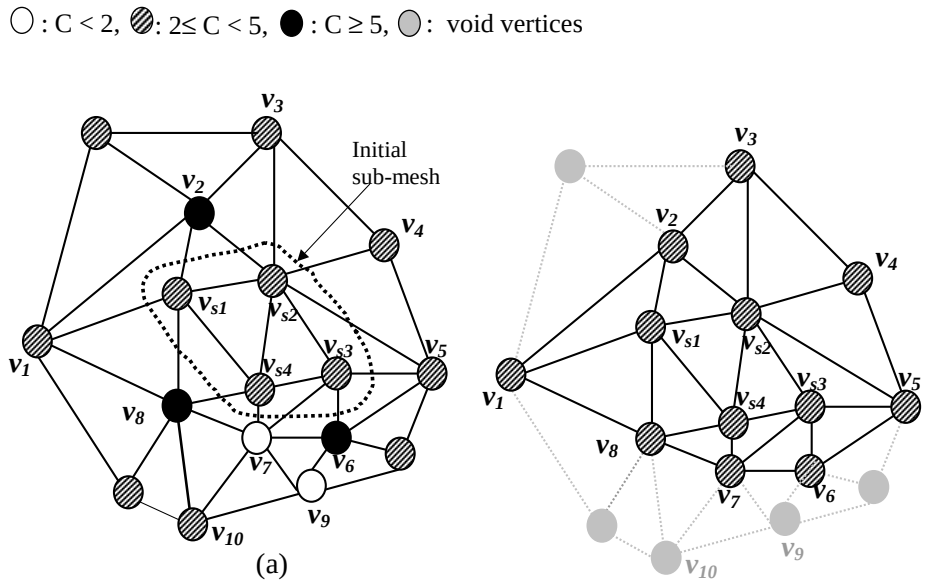
3. *Sub-bin processing for the determination of* t_{i+1}: While there exists the *first bin of interest* with bin center X_i, perform steps (a)-(c):

   (a) Using k-means, partition the curvedness values in this bin into two classes and identify the corresponding cluster centers $cen_1, cen_2; cen_1 < cen_2$. The cluster centers in Figures 4.5(b) and (d) are obtained by partitioning of the gray colored bins shown in Figure 4.5(a) and (c) respectively.

   (b) Set $t_{i+1} = cen_l$ where $l = \arg\min_{y=1,2} |X_i - cen_y|$. Thus, the cluster center that is closer to the bin center X_i is selected as a threshold. In Figure 4.5(b), $t_2 = cen_1$.

   (c) Extract MCASG corresponding to the threshold interval $[t_i, t_{i+1})$, using the segmentation algorithm described in Section 4.3.2. Figure 4.6 illustrates the various MCASGs obtained using the threshold intervals indicated in Figure 4.5.

4. Stopping Condition: Sub-bin processing terminates when none of the bins in the curvedness histogram satisfy the peakiness constraint. Then, the threshold interval for remaining



(a) Input: *G[(V,C),E]*, initial sub-mesh is shown in dotted lines; curvedness threshold range [2,5)

(b) Dilated graph: curvedness values of certain vertices are modified in the expanded sub-mesh

Figure 4.7: (a) Dilated graph extraction process involves expansion of the initial sub-mesh, identification and morphological filtering of outliers in the expanded sub-mesh. The idea exploited here is that a vertex geometrically behaves like its neighbors. Thus, it is reasonable to replace the curvedness value of an outlier by the median curvedness computed over its one-connected neighbors. For example, $C_{v7} =$ median $\{C_{v_{s3}}, C_{v_{s4}}, C_{v_6}, C_{v_8}, C_{v_9}, C_{v_{10}}\}$

unprocessed mesh is $[t_i, C_{max}]$. Such a histogram is shown in Figure 4.5(e).

Observe that, a new curvedness histogram is constructed at every level in the hierarchy. This is because segmentation (at the previous iteration) modifies the curvedness values of certain vertices. A fundamental advantage of such an approach is that the thresholds are selected without any user intervention and it does not require the specification of the desired number of sub-meshes.

4.3.2 Basic Segmentation Algorithm

Given an input mesh G and a curvedness threshold range $[t_i, t_{i+1})$, the extraction of a certain MCASG, say M involves:

1. *Initialization step:* Select an arbitrary vertex v from G such that its curvedness value $C_v \in [t_i, t_{i+1})$. Set $V(M^0) = v$. $E(M^0) = \emptyset$ and $C(M^0) = C_v$.

2. *Iteration Step:* For $k = 1, 2...$, perform the following:

   (a) *Dilated graph extraction:* Determine the neighbors of the vertices in $V(M^{k-1})$ and perform median filtering on their curvedness values if necessary. Extract the dilated graph G_d^k such that:

$$
\begin{aligned}
&V(G_d^k) = \{V(M^{k-1}) \cup v' | v' \in N(v), \exists v \in V(M^{k-1})\} \\
&C(G_d^k) = \{C'_v\}, where \\
&\forall v \in V(G_d^k), C'_v = \begin{cases} median\{C_{v'} | v' \in N(v)\}, & C_v \notin [t_i, t_{i+1}] \\ C(v), & otherwise \end{cases} \\
&E(G_d^k) = \{e_{uv} \in E(G) | u, v \in V(G_d^k)\}
\end{aligned}
\tag{4.8}
$$

   The dilated graph extraction process is illustrated in Figure 4.7.

   (b) *Desired graph A:* Using G_d^k, define desired graph A^k such that (a) $V(A^k) = V(G_d^k)$ (b) $E(A^k) = E(G_d^k)$ (c) $C(A^k) = \{\frac{t_i + t_{i+1}}{2}, \forall v_a \in V(A^k)\}$. By this definition, desired graph has the same sets of vertices and edges as the dilated graph. However, the curvedness values associated with the vertices in the desired graph are in the *desired* threshold interval. For example, corresponding to the dilated graphs shown in Figures 4.4 (c) and (f) , the desired graphs are illustrated in Figures 4.4 (d) and (g) respectively.

   (c) *Extraction of MCASG:* Attributed graph matching between dilated graph and desired graph will result in a *matched* graph M^k where (a) $V(M^k) = \{v \in V(G_d^k) | C_v \in [t_i, t_{i+1})\}$ (b) $E(M^k) = (V(M^k) \times V(M^k)) \cap E(G_d^k)$ (c) $C(M^k) = \{C_v, \forall v \in V(M^k)\}$.

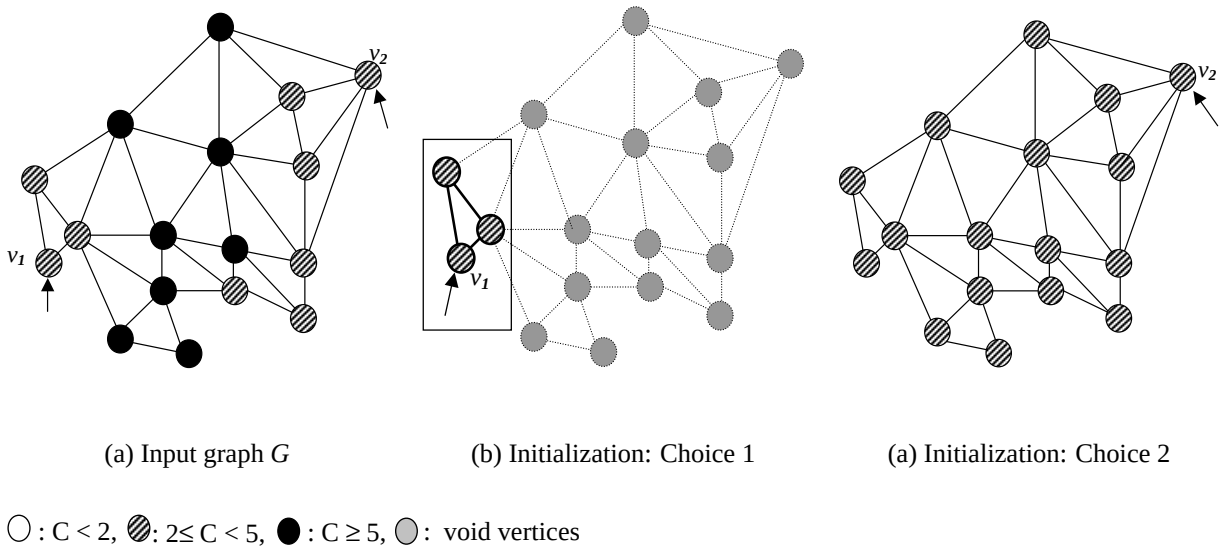


Figure 4.8: Algorithm exhibits robustness to bad initializations. Selection of vertex v_1 as the starting vertex doesn't allow the sub-mesh to grow as much. The vertices in such MCASGs are considered unprocessed. Vertex v_2 is a good choice for initialization.

The algorithm converges when $M^k = M^{k-1}$, thereby resulting in the MCASG corresponding to the threshold range $[t_i, t_{i+1})$.

It is observed that this approach causes the smoothing of a local surface shape, by modifying outlier curvedness values. We list below certain modifications to the basic algorithm that provide practical and robust segmentations.

4.3.3 Modified Algorithm

In Section 4.3.2, the segmentation initialization was done by arbitrarily selecting a vertex having its curvedness in the desired interval. However, if such a vertex has its neighbors outside the desired interval, then a reasonable sized MCASG may not be guaranteed. The implications of a bad initialization are illustrated using Figure 4.8. If we set $V(M^0) = v_1$ and implement our segmentation algorithm, we observe that the MCASG cannot grow as much as would have been expected, as shown in Figure 4.8(b). Since such MCASGs do not really represent *reasonably* large segmented regions, the vertices that comprise such MCASGs are considered 'unprocessed'

from a segmentation point of view. This leads to selection of another vertex for segmentation initialization. As shown in Figure 4.8(c), vertex v_2 is definitely a good starting point. It may be noted that the term *reasonable* is subjective, and for our problem, we drop MCASGs with fewer than 15 vertices, and consider these vertices as unprocessed. As we show in the examples, such an approach works well for quite a broad range of objects.our experience indicates that the algorithm is robust against bad initializations. For a given curvedness threshold range $[t_i, t_{i+1})$, due to the propagating nature of the segmentation algorithm, there will be a single MCASG at the output. In general, there may be several parts in any object with similar intensity of curvature, which are otherwise disconnected. We take into account such situations, and modify the basic segmentation algorithm to obtain multiple, disconnected, similar MCASGs.

Modified Segmentation Algorithm:

1. Obtain the list of all vertices, say *L*, satisfying the curvedness threshold criterion.

2. Select an arbitrary unprocessed vertex from L and implement the basic segmentation algorithm.

3. Drop the MCASG obtained in step 2, if it is not reasonably large (fewer than 15 vertices) and consider the corresponding vertices as unprocessed, else proceed to Step 4.

4. Repeat Step 2 until all vertices in *L* have been processed. This step ensures that all possible sets of MCASGs satisfying the given curvedness threshold criterion have been extracted.

Stopping Condition: The algorithm terminates either when all vertices in *L* have been processed or when only isolated regions with fewer than 15 vertices (which are not reachable by any propagations) are left.

4.4 Psychological Support

Algorithms that address the perceptual aspect of segmentation derive support from human vision theories, like the minima rule [111, 52](which defines boundaries along the lines of negative minima curvature) or Gestalt rules of organization [1, 15, 79, 80, 9, 50]. In this section, we show that in segmenting an input mesh into sub-meshes, our algorithm adheres to the following Gestalt rules of organization [1]. However, we do not provide a rigorous treatment of the perceptual aspect for that would require proofs from psychophysics, which are beyond the scope of this work.

- *Proximity Rule: Points that are close to each other are grouped together.*

  The proximity rule of perception is reflected in the definition of MCASG, that captures the adjacency relationship between any two vertices.

- *Similarity: Two points that are similar along a perceptual dimension (shape, color) are grouped together.*

  In the context of our problem, this rule implies that two vertices belong to the same sub-mesh if they have similar curvedness values. Clearly, this rule is incorporated in the definition of MCASG.

- *Good Continuation and Smoothness: Points that fit the path of a continuous curve are grouped together.*

  The formulation of dilation enforces the smoothness and continuation criteria. As explained in Section 4.2, dilation followed by morphological filtering results in smoothing of the local shape when the input data are noisy. This, in conjunction with the proximity rule, results in sufficiently large MCASGs (instead of small fragments) that delineate continuity. It may be mentioned that the minima rule does not deal with the *continuity* aspect [79] and hence, segmentation algorithms that are based on this rule also suffer from the drawback of lack of continuity.

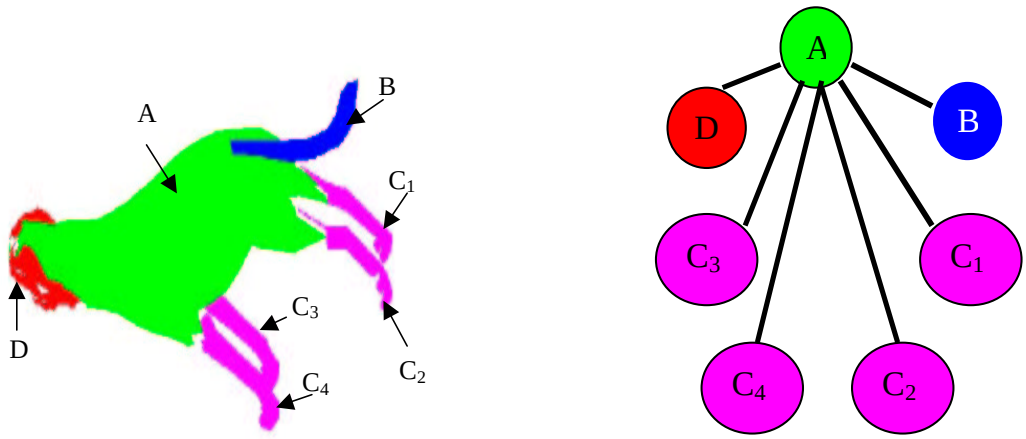
4.5 Two-tier Representation

For the sake of completeness, we describe below the design of our representation scheme. Due to the following key features that define an efficient representation scheme,

1. ability to represent all types of surfaces,
2. provision for accurate determination of orientation and translation parameters,
3. efficient representation for the surface described, in terms of storage and matching computation, and,
4. stable and sensitive in the sense that a small local change to a surface corresponds to a small, local change in its representation.

in this work, we are motivated to borrow ideas from the two-level Generalized Gaussian Image (GGI) representation scheme [31] because of the advantages it offers (1) rotation and translation paramaters are de-coupled; rotation of the object induces an equal rotation of normals on the sphere, (2) the GGI representation provides unique representations for convex and non-convex objects, and, (3) it allows us to uniquely determine a surface up to a translation thereby simplifying the recognition process considerably. In the GGI scheme, at the higher level, the connectivity between surface patches (where each patch consists of a set of vertices with constant Gaussian curvature) is established by constructing a graph linking the patches. At the lower level of representation, the surface normals of each one of the patches is mapped onto the unit sphere.

In our representation scheme, at the coarse scale, the global information about the object is provided by establishing adjacency relationship between various MCASGs. This is accomplished by constructing an attributed super-graph over the MCASGs. Each vertex in the super graph is a MCASG, the attributes associated with the vertices being the shape index thresholds and the range of curvedness values associated with the corresponding MCASGs. At the fine scale, the orientation information about each MCASG is preserved by mapping the corresponding normals onto a unit sphere. Figure 4.9 illustrates an example of the attributed supergraph



(a) MCASGs in a dog indicated by different colors

(b) Attributed super-graph over the MCASGs

Figure 4.9: An attributed supergraph representation of an object. An edge connects two vertices in the supergraph iff the corresponding MCASGs are adjacent each other

representation. It may be recalled that the normal to a triangle is defined as the cross product of the edge vectors. The normal at a certain vertex is then computed as the mean of the normals of the triangles that are incident on the vertex under consideration. Having computed the normals for all the vertices associated with a certain MCASG, each such normal is then mapped onto a unit sphere S^2. Thus, a normal vector at a vertex v in cartesian coordinates $\Re^3$ is denoted by $n(v) = \{n_x(v), n_y(v), n_z(v)\}$. When mapped onto a unit sphere S^2, it is represented using polar coordinates as $n(v) = \{\theta(v), \phi(v)\}$.

Since a MCASG describes a physical part in the underlying object e.g., the leg of a dog, the corresponding normals when mapped onto the unit sphere, will be spread across multiple regions on the sphere. As illustrated in Figure 4.10(a), the normals corresponding to the front and the rear faces of the leg of the dog map onto opposite regions on the sphere. These normals are then clustered based on the shape index values assigned to the corresponding vertices and the mean normal for each such cluster is determined. As shown in Figure 4.10(a), there are primarily two clusters of normals corresponding to the leg of the dog, i.e., MCASG C_1. For the two clusters, the mean normals are then computed as (θ_1, ϕ_1) and (θ_2, ϕ_2) respectively. As shown in Figure 4.10(b), it is possible that multiple MCASGs map onto the same point on the

(a) Clustering of the normals mapped on to a sphere

(b) Multiple folds on the sphere

Figure 4.10: Spherical Representation of the MCASG normals. Mean normals are determining by a clustering process. It is possible that multiple mean normals map onto the same point/neighborhood on the sphere resulting in multiple folds.

sphere, leading to multiple folds on the sphere. The attributed super-graph is then used to extract the connectivity information between various MCASGs, and thus, each fold can be identified uniquely. Referring to the block diagram illustrated in Figure 1.4, the derived representation for a model point cloud is stored in the database. The representation for the Scene point cloud is used for during recognition for classification and pose estimation. Specifically, the Scene is classified as an instance of a stored model by performing inexact graph matching between the Scene supergraph and the supergraphs corresponding to the stored models. Using the fine scale representation information, the rotation parameters that would align the Scene with respect to the identified model are then recovered. Determination of the translation parameters is then straightforward (we assume an affine model in this work). The challenge then is the determination of correspondence between the Scene and a subset of the model points. Our solution to the correspondence problem is described in Chapter 5 of this dissertation.

In the following section, we present experiments and results that validate the performance of the proposed segmentation algorithm.

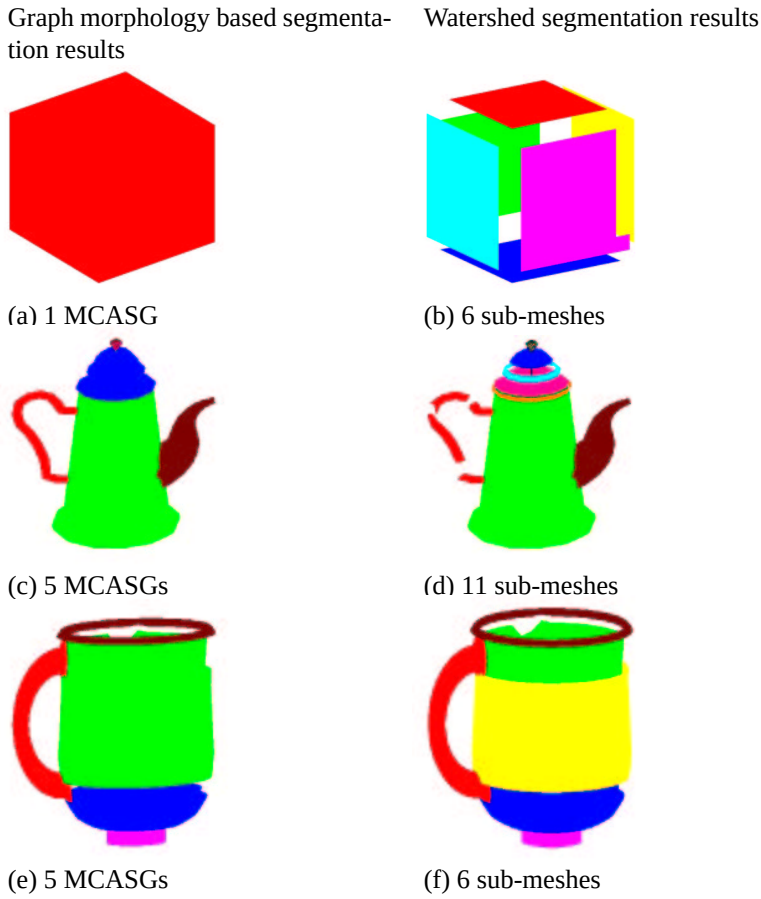


Figure 4.11: Proposed segmentation algorithm partitions the input mesh into submeshes corresponding to the physical parts of the underlying object. Since a cube consists of planar faces, the algorithm outputs only one MCASG whereas the watershed algorithm results in 6 sub-meshes.

4.6 Experiments and Discussion

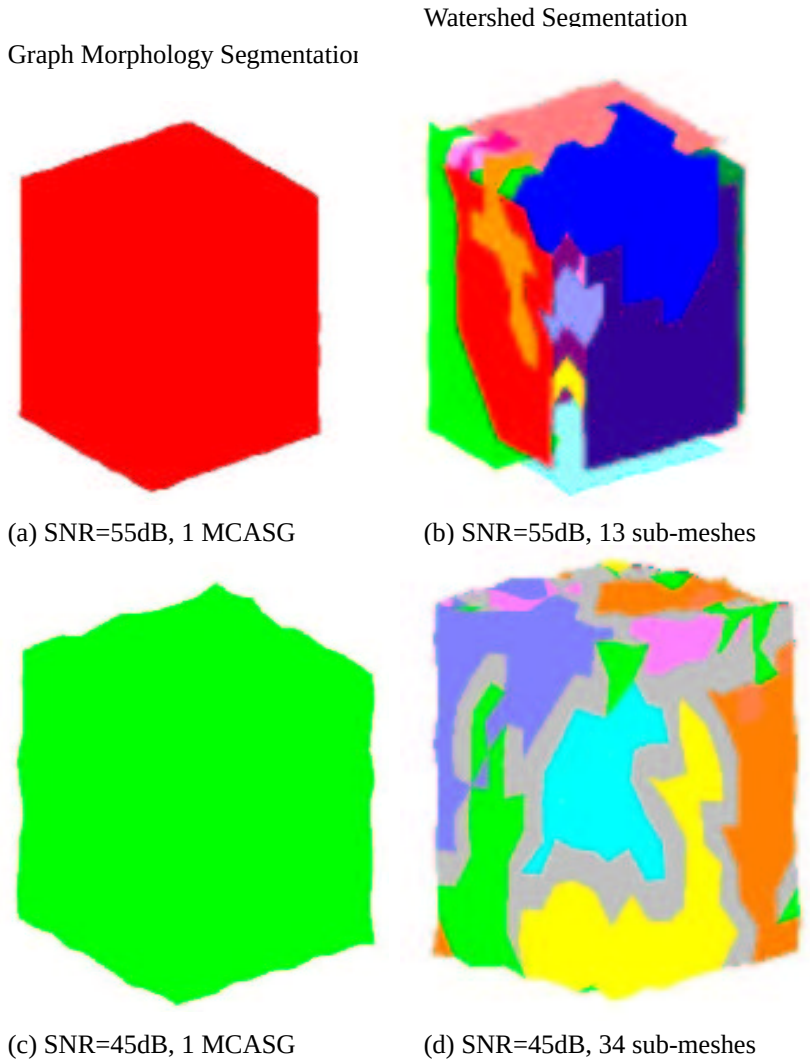
4.6.1 Comparison with the State-of-the-Art

A comprehensive comparison of our algorithm with all the existing state-of-the-art approaches is beyond the scope of this thesis. From the mesh segmentation literature, we selected the watershed algorithm [75] for evaluation purposes, primarily because it is also built on ideas borrowed from morphology. Our algorithm differs from [75] in the way the morphological operators are defined and applied.

Perceptual Aspect

To qualitatively analyze the segmentations, we simulated a uniformly-sampled point cloud of a cube, which was subsequently triangulated using the commercially available *Points2Polys* software. Our algorithm segments the cube into exactly one MCASG as shown in Figure 4.11(a). The watershed algorithm [75], on the other hand, segments the mesh into six sub-meshes (Figure 4.11(b)), wherein, each sub-mesh corresponds to a face in the cube. The input to both the algorithms is the same surface mesh. For a cube, the curvatures and hence the curvedness values associated with the vertices that lie along the edges are starkly different from the values associated with the interior vertices. The vertices along the edges are treated as outliers and the process of graph dilation forces the outliers to behave like their neighbors by modifying their curvedness values. Hence, segmentation results in the exactly one MCASG. On the other hand, the watershed algorithm [75], treats the vertices along the edges of the cube as points of minima. Hence gradient descent from vertices, lying on the interior of the cube, toward the minima, results in six connected components. It may well be argued that partitioning a cube into six sub-meshes (obtained using watershed algorithm) is more *meaningful* than a segmentation into a single MCASG. As stated in the beginning of the paper, the definition of *meaningful* is highly application dependent. For the purposes of recognition, we think that it is reasonable to partition a cube into a single MCASG. We point out that our results are similar to the perceptual segmentation results presented in [111].

Figures 4.11(c),(e) represents the segmentation results on a tea pot and tea cup using the proposed graph morphology based segmentation algorithm, while Figures 4.11(d),(f) represent corresponding segmentations obtained using the watershed algorithm. For the tea pot, our segmentation algorithm results in five MCASGs, while the application of the watershed algorithm results in 11 sub-meshes. We remark again that the segmentation results of the tea cup and tea pot, as provided by our algorithm are comparable to the perceptual segmentation results obtained by the fast marching watershed algorithm presented in [111] (Figures 2(a),(b) and (c), Figure

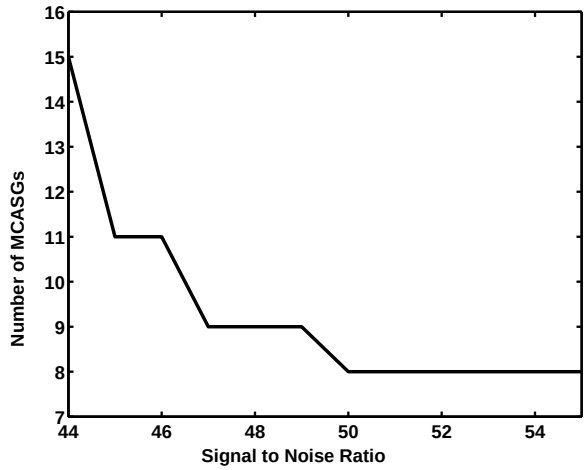


(a) SNR=55dB, 1 MCASG (b) SNR=55dB, 13 sub-meshes

(c) SNR=45dB, 1 MCASG (d) SNR=45dB, 34 sub-meshes

Figure 4.12: For reasonable noise levels, our proposed algorithm partitions the cube into exactly one MCASG. On the other hand, the watershed segmentation algorithm does not provide the desired results.

(a) SNR =44dB; 15 MACSGs obtained for Noisy Horse with 59547 vertices



(b) Plot of SNR vs. Number of MCASGs for the horse

Figure 4.13: The point cloud of the horse was subjecting to varying amounts of Gaussian noise (SNR between 44dB and 55 dB). Considerable amount of noise is required before the MCASGs results in patchy sub-meshes.

3(c) and Figure 9(b) in [111]).

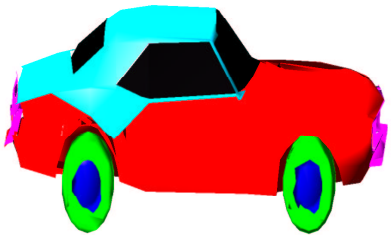
Effect of Noise

The point cloud of the cube was subjected to two different levels of Gaussian noise resulting in SNR=55 dB and 45 dB. SNR is the ratio of signal-to-noise energy on a logarithmic scale and is mathematically expressed as: $SNR = 20log(S/\sigma_n^2)$ where σ_n^2 is the variance of the Gaussian noise and S is the maximal signal strength. Since the noise causes a perturbation of the vertices that constitute the point cloud, the curvature estimates are not accurate, resulting in many more outlier vertices or minima (as compared to the noise-free point cloud). As illustrated in Figure 4.12(a) and (c), due to median filtering during dilation, our algorithm segments the noisy input mesh into exactly 1 MCASG. The watershed algorithm results in 13 and 34 sub-meshes for SNR=55 dB, and 45 dB respectively, as shown in Figure 4.12(b) and (d).

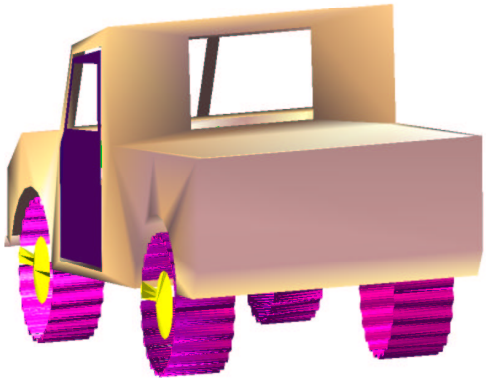
The effect of noise on more complex surfaces was analyzed by subjecting the point cloud of the horse to varying amounts of white Gaussian noise (SNR varying between 44dB and 55dB). Figure 4.13(a) indicates the segmentation obtained at SNR=44dB. From Figure 4.13(b), we

(a) Pig: 11 MCASGs

(b) Human: 15 MCASGs

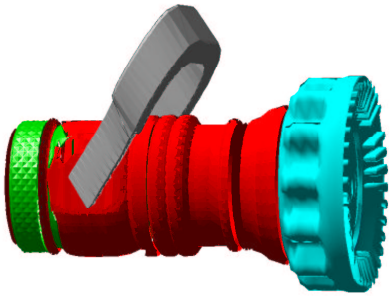


(c) Car: 19 MCASGs



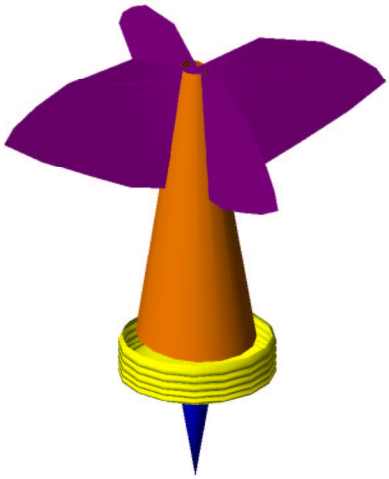
(d) Pickup Truck: 15 MCASGs

Figure 4.14: The algorithm allows for reconciliation between disjoint yet similar sub-meshes. Establishing such an association between similar, disjoint sub-meshes is vital for higher level tasks such as object recognition

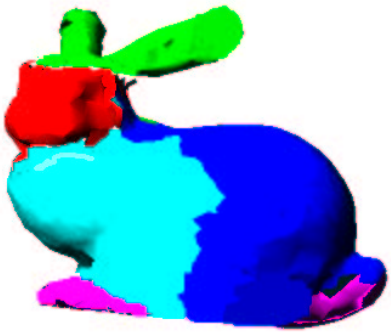


(a) fire hose nozzle: 7 MCASGs

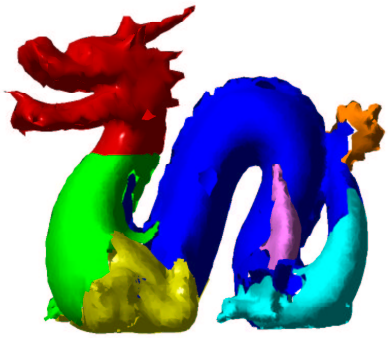
(b) Lamp: 5 MCASGs



(c) Dart: 4 MCASGs



(d) Bunny: 6 MCASGs



(e) Dragon: 7 MCASGs

Figure 4.15: Segmentation of complex surfaces

conclude that considerable noise is required before the algorithm results in patchy MCASGs.

4.6.2 Complex Data Sets

In order to demonstrate the effectiveness of the proposed algorithm, acceptable test cases from the realms of computer graphics and object recognition/machine vision were segmented into corresponding MCASGs. As the results in Figures 4.14 and 4.15 demonstrate, the proposed algorithm provides coarse yet clean segmentations for objects such as a car, a pickup truck, a dart and a lamp. For the textured surfaces such as the bunny and the dragon, the algorithm seems to over-segment certain regions of the mesh. These results confirm (1) the robustness of the adaptive threshold selection process and its applicability in a wide context (2) the extraction of multiple similar yet disjoint MCASGs using the modified segmentation algorithm. The reconciliation between such similar disjoint MCASGs can be used in higher level tasks such as object recognition.

Table 4.1 shows the timing performance of the proposed algorithm on various data sets. The algorithm was coded in Matlab, and tested on Pentium IV processor at 1.5 GHz, 256MB memory.

4.7 Conclusions

In this chapter, a graph morphology-based 3D mesh segmentation algorithm was presented to classify vertices into different categories based on their intensities of curvatures. The proposed threshold selection technique requires zero user intervention and provides robust segmentations for a wide variety of test cases. The segmentation process allows for extraction of multiple similar yet disconnected sub-meshes. The extracted sub-meshes seem to match the human visual segmentation of the underlying object. Results indicate that graph dilation together with morphological filtering of outliers can effectively deal with the noise The algorithm compares well with

object	Number of vertices	Number of sub-meshes	Time (in seconds)
human	2097	15	13.98
dart	1122	4	6.23
tea pot	2220	5	15.11
tea cup	11241	5	75.08
simplified horse	1548	8	11.59
horse	59547	8	451.31
Car	7401	19	43.30
Pickup Truck	4902	15	29.97
Bunny	35947	6	238.61
Pig	4332	11	27.89
Lamp	1954	5	14.25
Fire-hose nozzle	5885	7	38.78
Dragon	22998	7	189.56

Table 4.1: Timing Performance of the proposed segmentation algorithm on various data sets

the existing state-of-the art approaches, it suffers from zero pre-processing and post-processing overheads and can be effectively used for higher level tasks such as object recognition. To avoid over-segmentations of textured objects such as the Stanford bunny or the dragon and to obtain segmentation results for such objects that seem perceptual, texture-based features need to be considered in addition to the shape features. A two-tier representation scheme is then designed to aid in classification and pose estimation during recognition.

Chapter 5

Point Cloud Matching within Graph-theoretic and Thermodynamic Frameworks

The task of recognition of a given partial, unstructured point cloud of the *Scene* (query data) using a database of stored representations of the 3D Model point clouds involves (i) classification of the *Scene* as an instance of one or more stored Models (ii) alignment of the *Scene* with respect to the identified *Model* (iii) determination of the location of the *Scene* within the identified *Model*.

In our recognition system, given the two-tier representation for the *Scene*, classification is accomplished by performing inexact graph matching between the Scene super-graph and the stored attributed super-graphs of the models. The shape attributes (shape index and curvedness thresholds) associated with the super-graph nodes (MCASGs) allow for a rapid pruning of the model database. The rotation parameters are then computed as a function of the angular distance between the mean normals associated with the scene and the identified model GGIs. The translation parameters are determined by assuming an affine transformation model. This chapter addresses the problem of *location determination* i.e., determination of a set of points in the identified *Model*

that are structurally and spatially as similar as possible to the partial *Scene* point cloud.

There are two fundamental issues that make this problem challenging. First, the number of points in the *Model* and the *Scene* point clouds are orders of magnitude different. Secondly, due to sensor inaccuracies or because the *Scene* points are collected at different times, the two point sets may be non-overlapping i.e., no two points correspond to the exact same location in the 3D coordinate space.

As mentioned in Chapter 2, graph-based structural approaches and spatial location based algorithms have been reported in the literature on point matching. Graph-based algorithms establish the desired correspondence by matching configurations of *Scene* features to those of a *Model* [116]. In inexact graph matching, approximate solutions to the problem are obtained based on the minimization of the edit distance [115, 119], probabilistic optimization [117, 105], deterministic annealing [**?**], bipartite graph matching [21, 108] and spectral graph theory [22, 70, 84, 12]. Most existing graph matching techniques suffer from the inability to match graphs of largely varying cardinalities. Additionally, their performance severely degrades with small perturbations (positional jitter). Spatial matching approaches determine correspondence solely based on the spatial location of the points [88, 57]. While these algorithms are generally robust, they do not take into account the underlying structural information that exists between the points in a set. With non-overlapping point sets, this could be a problem, since there may be more than one subset of *Model* points that is spatially close to the *Scene*.

In this work, a thermodynamically inspired objective function is proposed to capture the structural nuances between a pair of graphs and the spatial differences between the underlying point sets. The desired correspondence is obtained by tackling a sequence of inexact graph matching problems that optimizes the proposed objective function.

We now provide a brief overview of our algorithm, which works in two stages. In the first stage, to facilitate inexact graph matching, the *Model* space is partitioned into *Model Clusters* (*MC*s), such that $|MC_i| = |Scene|$ ($|\cdot|$ represents the cardinality of a set). The change in

entropy, computed for every *MC* relative to the *Scene*, identifies the *MC* that is spatially the closest to the *Scene* and the model *neighborhood* around it. The advantage of such an approach is that the closest *MC* already provides a fraction of correct correspondences. In the second stage, the maximization of the free energy between the *Scene* and the *closest MC*, which is achieved by swapping certain points between the *closest MC* and the identified *Model neighborhood*, results in the desired correspondence. The reason for using different objective functions (i.e., based on entropy or free energy) during different processing stages is strongly motivated by the principles of thermodynamics. These ideas are non-trivially extended to deal with missing data. Also, we use ideas from thermodynamics of heterogeneous systems to address challenges in part-based matching approaches.

We contribute to the existing state-of-the-art by:

- defining graph enthalpy to quantify the underlying structural information in the point sets,
- deriving the Gibbs' free energy for the point sets based on the proposed formulation of graph enthalpy and existing notions of graph entropy,
- optimizing the free energy-based cost function to obtain the desired correspondence between the *Scene* and a subset of the *Model* points, and,
- proposing a part-based matching algorithm that uses ideas from the thermodynamics of heterogeneous systems

This chapter is organized as follows. In Section 5.1, we provide descriptions of various graph structures used in this chapter. In Section 5.2, we derive formulations for graph enthalpy and Gibbs free energy. Section 5.3 details the basic correspondence determination algorithm. In Section 5.4, we use ideas from thermodynamics of heterogeneous systems in the context of part-based matching. Experiments and results are provided in Section 5.5.

5.1 Definitions and Notations

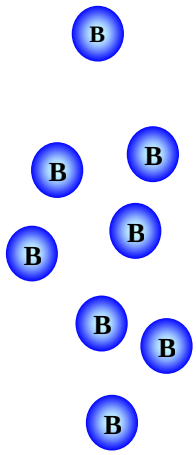
In Chapter 2, we defined the term *matching* in the context of graphs. Given a complete bipartite graph $G_{CB}((V_1, V_2), E)$, $|V_1| = u_1$ and $|V_2| = u_2$, a matching is *perfect*, when $|E_1| = \lfloor |u_1 + u_2|/2 \rfloor$ [64]. A minimum weight perfect matching bipartite graph $G_B((V_1, V_2), E_{V_1V_2})$, where $|V_1| = |V_2| = u$ and $E_{V_1V_2}$ denotes a minimum weight perfect matching, is obtained by the implementation of the Hungarian method on G_{CB} [64].

Let *Q* and *M* denote the query (*Scene*) and the identified *Model* point sets respectively. $|M| \gg\gg |Q|$. *MC* represents a certain model cluster, obtained by partitioning the *Model* space such that $|MC| = |Q|$.

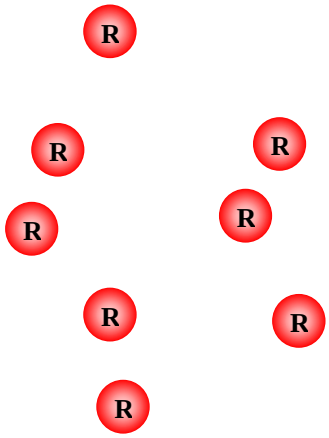
As illustrated in Figure 5.1, the graphs of interest are:

- $G_{MST}(Q, E_Q), G_{MST}(MC, E_{MC})$ represent the minimum spanning tree (MST) constructed over Q and a certain MC respectively.
- $G_B((Q, MC), E_{QMC})$ denotes a bipartite graph constructed over the point sets *Q*, *MC*, where, E_{QMC} represents the minimum weight perfect matching.
- The union of $G_{MST}(Q, E_Q)$ and $G_B((Q, MC), E_{QMC})$ is denoted by $G_{U1}((Q, MC), (E_Q, E_{QMC}))$. The union of $G_{MST}(MC, E_{MC})$ and $G_B((Q, MC), E_{QMC})$ is denoted by $G_{U2}((MC, Q), (E_{MC}, E_{QMC}))$.

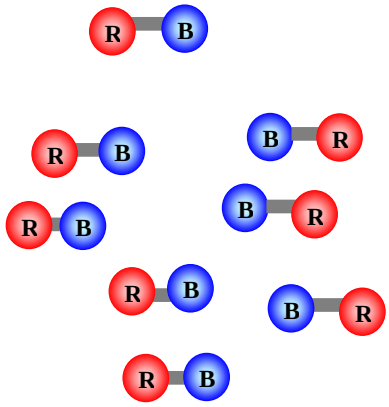
The above graphs, denoted by $G_{MST}(Q)$, $G_{MST}(MC)$, G_B, G_{U1}, G_{U2}, will be used in Section 5.2.1 to define graph enthalpy.



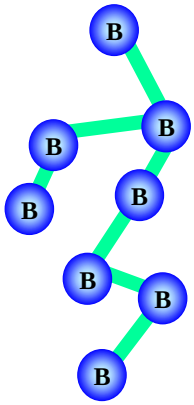
(a) Point Set Q



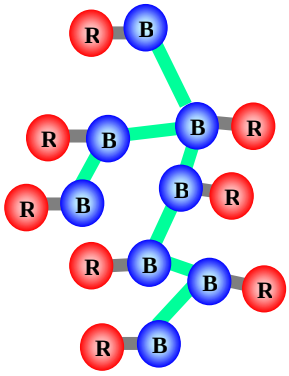
(b) Point Set MC



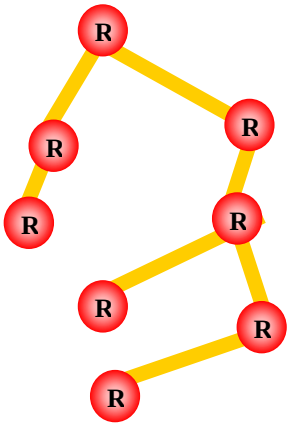
(c) Minimum weight perfect matching bipartite graph G_B



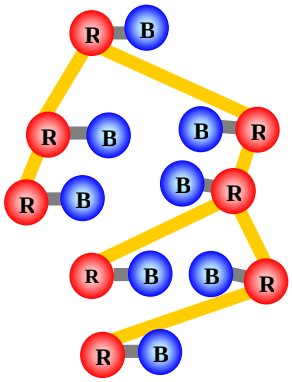
(a)$G_{MST}(Q)$



(d)$G_{U1} = G_{MST}(Q) + G_B$



(b)$G_{MST}(MC)$



(e)$G_{U2} = G_{MST}(MC) + G_B$

Figure 5.1: Given: Two point sets Q and MC; $|Q| = |MC|$. $G_B, G_{MST}(Q), G_{U1}, G_{MST}(MC), G_{U2}$ are the graph structures of interest in this work. Note that $G_B(Q, MC, E_{QMC}) = G_B(MC, Q, E_{QMC})$. When the weights of the edges in G_B are all zero, point sets Q and MC completely overlap. Then, $G_{U1}, G_{MST}(Q)$ have the same structure. $G_{U2}, G_{MST}(MC)$ have the same structure as well

5.2 Point Matching via Classical Thermodynamics- Theory

From Chapter 2, recall that in chemical thermodynamics, substance conversion results in the Gibbs' Free Energy ΔG, which quantifies the structural and the spatial differences as:

$$\Delta G = \Delta H - T\Delta S$$

where T is the temperature and ΔH is the change in *enthalpy*, resulting from the structural difference caused by the addition/deletion of chemical bonds between molecules. ΔS is the change in *entropy* due to the spatial disorder of the molecules involved [67].

Since graph edges are analogous to chemical bonds [4], a new formulation for graph enthalpy, quantifying the structural differences between a pair of graphs, is proposed in Section 5.2.1. The spatial differences between the point sets are estimated using the existing notions of graph entropy [88]. The Gibbs's free energy for the point sets is then derived based on these differences.

5.2.1 Enthalpy Change: Measure of Structural Difference

In classical thermodynamics, at a constant pressure P, the enthalpy H_{s_1} of a substance s_1 is given by:

$$H_{s_1} = U_{s_1} + PV_{s_1}$$

where, U_{s_1} is the internal energy and V_{s_1} is the volume occupied by the molecules in s_1. The structural differences associated with the conversion of s_1 to s_2 contributes to a change in enthalpy ΔH where:

$$\begin{aligned} \Delta H &= H_{s_2} - H_{s_1} \\ &= (U_{s_2} - U_{s_1}) + P(V_{s_2} - V_{s_1}) \end{aligned}$$

In a similar vein, the enthalpy of a graph *G* is computed as:

$$H(G) = U(G) + PV(G) \tag{5.1}$$

where $P = 1$ to indicate a constant pressure process. $V(G)$ is the volume occupied by the vertices in *G*, and is computed as the volume of the convex hull of the corresponding 3D points [51]. The internal energy $U(G)$ is computed as the minimum energy of a balanced orthogonal representation of G, using the following theorem:

Theorem 5.1 [92]: *For a graph G on n vertices and the corresponding weighted Laplacian L, let the eigenvalues of L be $\lambda_1 \leq \lambda_2 \leq ... \leq \lambda_n$ and that $\lambda_2 > 0$. The minimum energy of a balanced orthogonal representation of* G *in $\Re^m$ equals $\sum_{i=2}^{m+1} \lambda_i$.*

Consider Figure 5.1. In the context of matching of point sets *Q* and a *MC*, ($|Q| = |MC|$), the change in graph enthalpy ΔH is determined by considering $G_{MST}(Q), G_{MST}(MC), G_{U1}$ and G_{U2}. Specifically, we are interested in evaluating the structural differences between $G_{MST}(Q)$ and G_{U1} on one hand and the structural differences between $G_{MST}(MC)$ and G_{U2} on the other. In this regard, two interesting features of G_B need attention. First, the edges in G_B provide a one-to-one correspondence between *MC* and *Q*. Secondly, these edges are indicative of the extent of structural dissimilarity between $G_{MST}(Q)$ and G_{U1} or $G_{MST}(MC)$ and G_{U2} for the following reason. Since the weight of an edge in G_B is a measure of the Euclidean distance between a vertex in MC and the corresponding vertex in Q, when the weights of edges in G_B are all zero, then the point sets are completely overlapping. Then G_{U1} and $G_{MST}(Q)$ are structurally the same. G_{U2} and $G_{MST}(MC)$ will have the same structure as well. However, when the edge weights in G_B are not all zero, the structural difference ΔH_1 between $G_{MST}(Q)$ and G_{U1} is given by:

$$\Delta H_1 = H(G_{U1}) - H(G_{MST}(Q)) \tag{5.2}$$

and, the structural difference ΔH_2 between $G_{MST}(MC)$ and G_{U2} is given by:

$$\Delta H_2 = H(G_{U2}) - H(G_{MST}(MC)) \tag{5.3}$$

Then, ΔH quantifies the total structural difference for the two point sets *Q* and *MC* as:

$$\Delta H = \Delta H_1 + \Delta H_2 \tag{5.4}$$

Note that for completely overlapping point sets, $\Delta H_1, \Delta H_2, \Delta H$ are all zero.

5.2.2 Entropy Change: Measure of Spatial Difference

Given a set χ_n, consisting of *n* points, Ma *et.al* [88] estimate the entropy by the power weighted length of the MST constructed over the set of vertices as:

$$\hat{S}(\chi_n) = \frac{1}{1-\alpha}[\log \frac{L(\chi_n)}{n^\alpha} - \log \beta] \tag{5.5}$$

where, $L(\chi_n)$ is the length of the minimum spanning tree, β is a constant independent of the underlying density of the points and α is the fractional order of the density.

In this work, the change in entropy ΔS for the two sets of points *Q* and *MC*, is computed as

$$\Delta S = \Delta S_1 + \Delta S_2 \tag{5.6}$$

where,

$$\begin{aligned} \Delta S_1 &= \hat{S}(Q \cup MC) - \hat{S}(Q) \\ \Delta S_2 &= \hat{S}(Q \cup MC) - \hat{S}(MC) \end{aligned} \tag{5.7}$$

With the knowledge of $\Delta H, \Delta S$, the Free energy ΔG is computed for same-sized point sets *Q* and *MC*. The temperature *T* measures the degree of desired correspondence. High temperatures imply low degrees of desired correspondence and vice versa. The initialization of *T* is described in Section 5.3.4, within the context of the proposed matching algorithm.

5.2.3 Significance of Thermodynamic Quantities in the Context of the Problem

By the laws of thermodynamics,

(a) At very high temperatures, ΔG is dominated by entropic contributions, and at very low temperatures, it is dominated by enthalpic contributions.

(b) ΔH and ΔS decrease with decrease in temperature.

(c) At high temperatures, ΔG is a large negative number. As the temperature decreases, ΔG increases toward zero.

(d) At $T = 0$, $\Delta H = 0, \Delta S = 0, \Delta G = 0$.

Motivated by (a), our matching algorithm uses ΔS and ΔG as objective functions for the coarse scale and fine scale processing respectively. Property (b) provides a pruning strategy for minimization of ΔS during the coarse scale processing. The temperature's role in the algorithm is influenced by (c), and from a graph-theoretic perspective, the following theorem is in order.

Theorem 5.2 [92]: *Let X be a graph with n vertices and let Y be obtained from X by adding an edge joining two distinct vertices of X. Then $\lambda_i(X) \leq \lambda_i(Y)$ for all i.*

Theorems 5.1 and 5.2 imply that $\Delta H > 0$. Also, $\Delta S > 0$. Therefore, the temperature *T* decides whether $\Delta G < 0$ or not. In the proposed point matching algorithm, in order to ensure that $\Delta G < 0$, *T* is initialized to a large number. This is described in detail in Section 5.3.4.

Using property (d) in conjunction with the proposed definition of the temperature T, we can say that the desired correspondence between *Q* and a subset of the *Model* points is achieved when ΔG between the point sets reaches its maximum, at $T = 0$. In general, at any given *T*, the edges in G_B provide the correspondence between the points in a certain *MC* and *Q*.

5.3 Algorithm

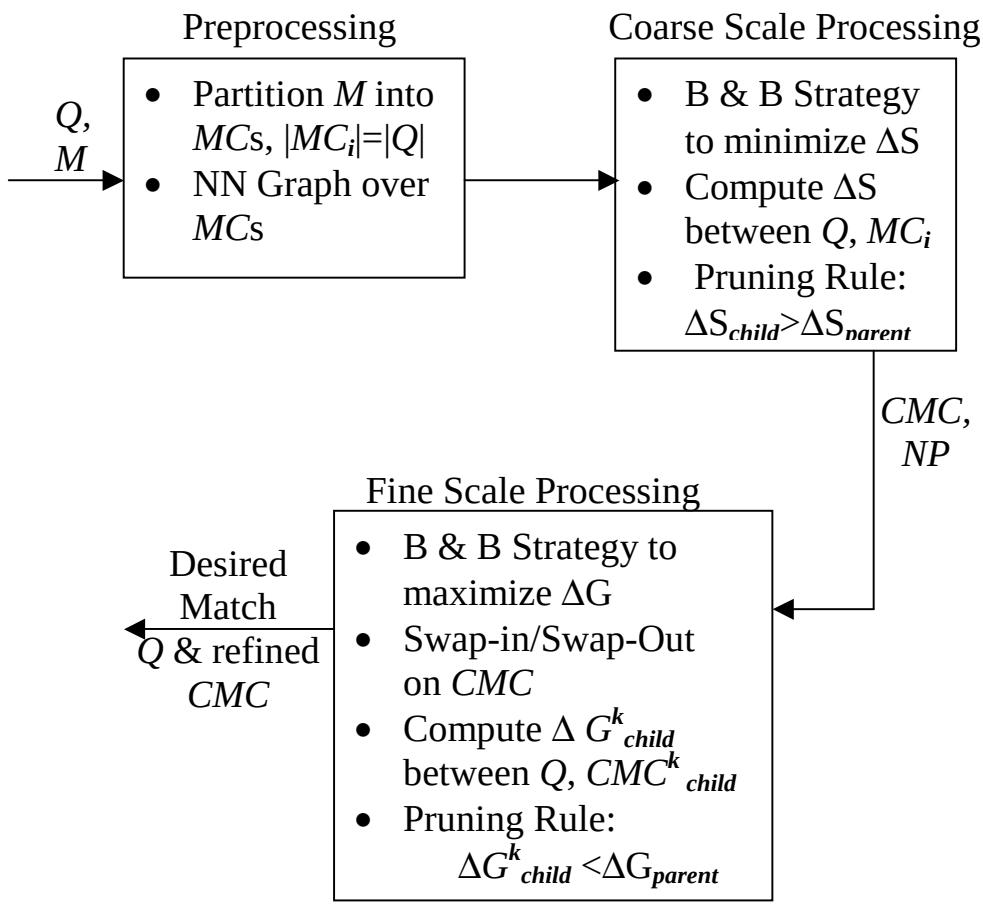


Figure 5.2: Block diagram of the proposed point matching algorithm: The preprocessing step involves the partitioning of the model space *M* into model clusters (*MC*s) and the construction of a Nearest Neighbor (*NN*) graph over these *MC*s. At the coarse scale, a Branch and Bound (B&B) optimization scheme for minimization of ΔS allows us to identify the *Closest Model Cluster* (CMC). At the fine scale, the desired correspondence is obtained by maximization of ΔG

Figure 5.2 provides an outline of the proposed algorithm. The preprocessing step involves

(i) the partitioning of the model space *M* into model clusters (*MC*s) ($|MC_i| = |Q|$) and, (ii) the construction of a Nearest Neighbor (*NN*) graph over these *MC*s. During the coarse scale processing step, the *MC* that minimizes ΔS is identified as the closest *MC* (*CMC*) and its one-connected neighbors form the neighborhood pool (*NP*). The immediate advantage of identifying the *CMC* is that it provides a fraction of correct correspondences. However, since the partitioning is in a sense blind, it is very unlikely that the *CMC* will provide the desired correspondence. This forces a fine scale processing step, wherein, the maximization of ΔG, by swapping certain vertices between the *CMC* and *NP*, leads to the desired correspondence.

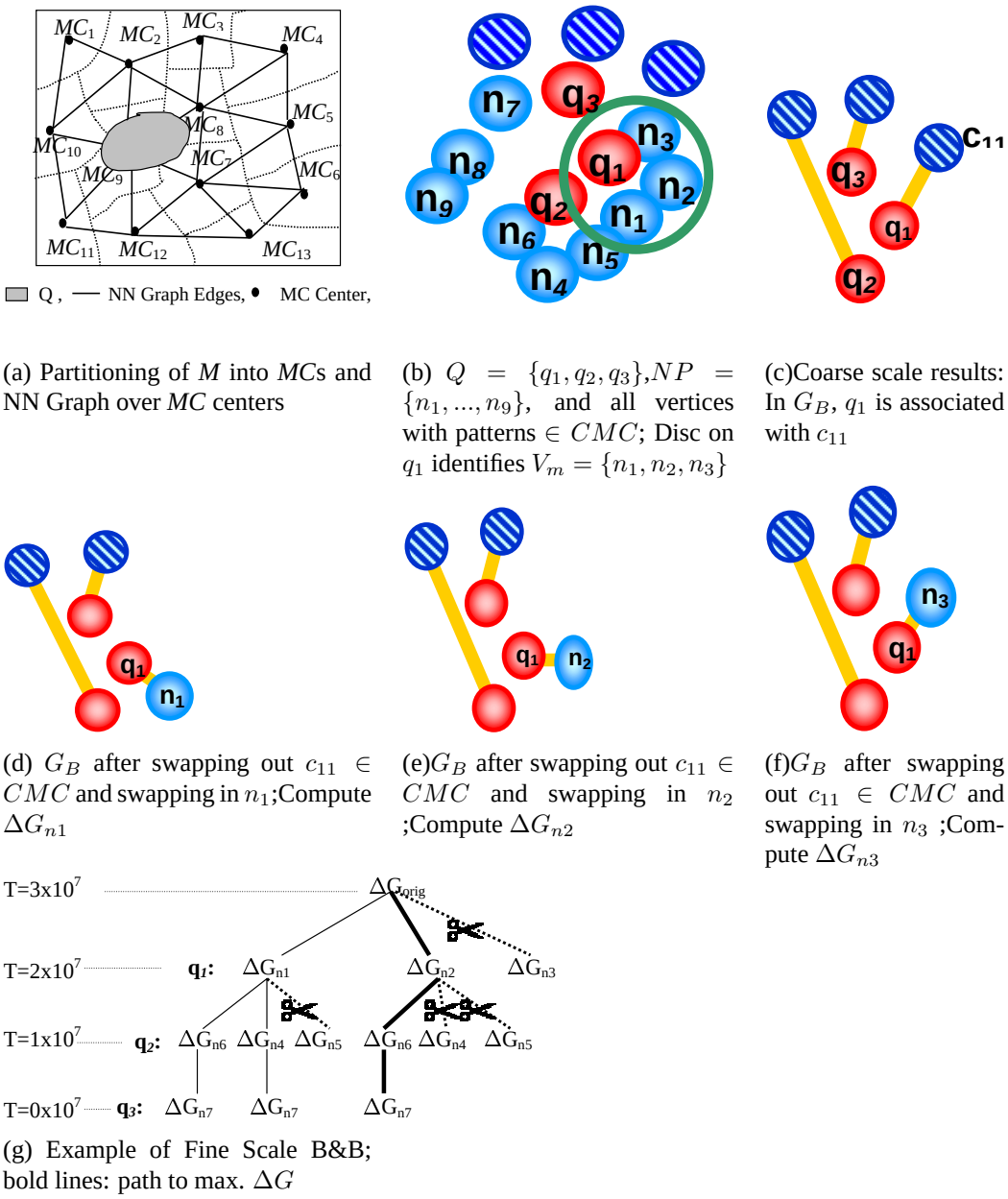
The optimization of the cost functions in both the processing stages is achieved using a Branch and Bound (B&B) approach, wherein, a thermodynamically inspired pruning strategy reduces the number of nodes and branches in the search tree that have to be explored. The processing modules are described below.

5.3.1 Preprocessing

Following [44], the n points that comprise a *MC* (where $n = |Q|$) are determined by performing a breadth-first search on a Delaunay triangulation [51] which is constructed over *M*. The center of a *MC* is then identified as the vertex with minimum eccentricity. Another Delaunay triangulation constructed over these centers serves as a *NN* graph for coarse scale processing.

5.3.2 Coarse Scale B&B Algorithm

The primary objective here is to identify the *MC* that minimizes ΔS, i.e., the *CMC*. For this, the *MC* corresponding to the vertex with the smallest degree (typically on the periphery) in the *NN* graph is assigned as the root node in the B&B search tree. In the search tree, the connected neighbors of a *MC* form its *children*, and at any node (*MC*), using (5.6) and (5.7), ΔS is computed between *Q* and the corresponding *MC*. While traversing the tree down to the leaves, if $\Delta S_{child} > \Delta S_{parent}$, then the sub-tree is automatically pruned off at such parent nodes. This is because an



(a) Partitioning of M into MCs and NN Graph over MC centers

(b) $Q = \{q_1, q_2, q_3\}$,$NP = \{n_1, ..., n_9\}$, and all vertices with patterns $\in CMC$; Disc on q_1 identifies $V_m = \{n_1, n_2, n_3\}$

(c)Coarse scale results: In G_B, q_1 is associated with c_{11}

(d) G_B after swapping out $c_{11} \in CMC$ and swapping in n_1;Compute ΔG_{n1}

(e)G_B after swapping out $c_{11} \in CMC$ and swapping in n_2 ;Compute ΔG_{n2}

(f)G_B after swapping out $c_{11} \in CMC$ and swapping in n_3 ;Compute ΔG_{n3}

(g) Example of Fine Scale B&B; bold lines: path to max. ΔG

Figure 5.3: In (a), MC_1 is the root in coarse scale B&B tree. *CMC* is identified as MC_9 and $NP = \{MC_2, MC_7, MC_8, MC_{10}, MC_{11}, MC_{12}\}$. (b)-(e) illustrate fine scale processing steps. Since we need to refine the correspondence for all three vertices in Q, therefore, in (e) the temperature at root is set to $T = 3 \times 10^7$.

increase in ΔS signals an increase in spatial dissimilarity (between a *MC* and *Q*), which further implies that we are moving away from *Q* rather than moving in a direction toward it. Such a pruning is consistent with the monotonicity property of the B&B approach [64]. The algorithm's output is the *CMC*, which together with its one connected neighbors i.e., *NP*, is used for the refinement of correspondences at the finer scale.

5.3.3 Fine Scale B&B Algorithm

The objective here is to maximize ΔG by determining the best correspondence for every point $q_i \in Q$ that has not already found its desired match $c_j \in CMC$. The coarse scale *CMC* forms the root node in the fine scale B&B search tree. With respect to each q_i, the set $V_m \subset (NP \cup CMC)$ comprising of *k* model points of interest, is identified by centering a disc of radius r on q_i. At any node in the search tree, ΔG^k_{child} is computed between CMC^k_{child} and Q, where CMC^k_{child} is obtained by *swapping out* a point $c_l \in CMC$ that was originally associated with q_i, and *swapping in* $V_m(k)$. That is,

$$CMC^k_{child} = (CMC_{parent} \backslash c_l) \cup V_m(k), k = 1, ..., |V_m| \tag{5.8}$$

Swapped out vertices are returned to the NP. Sub trees are pruned off at *parent* nodes when $\Delta G^k_{child} < \Delta G_{parent}$ (property (c),Section 5.2.3). Using $G_{MST}(Q)$, a connected neighbor of q_i generates the *children* in the search tree. Upon termination, the edges in G_B, associating the refined *CMC* and *Q*, provide the desired correspondence. The fine scale processing is illustrated in Figure 5.3(b)-(e).

5.3.4 Effect of Temperature on ΔG

By the laws of thermodynamics (Section 5.2.3, property (a)), entropic contributions are dominant only at high temperatures. This justifies the need to start the matching process at high temperatures. At the fine scale, the objective is to determine a one-to-one structural match between the *Scene* Q and a subset of the *Model* points (identified at the coarse scale). This is possible by emphasizing ΔH, which plays a dominant role only at low temperatures. For these reasons, we begin the optimization process at a high temperature and progressively decrease it to lower values, until the desired correspondence is reached.

At the level 0, i.e., at the root of the B&B search tree, the temperature *T* is initialized as $T = (|Q| - d) \times 10^7$, where d is the number of points in the coarse scale *CMC* that completely overlap with *Q*. $T = (|Q| - d - i) \times 10^7$ at the tree's level $i, 1 \leq i \leq |Q| - d$. Different levels in the tree, along the path leading to maximum ΔG, contribute to different fractions of the desired correspondence (due to the refinement of the *CMC*). The desired match between *Q* and *CMC* is reached at path's leaf, when $T = 0$.

5.3.5 Missing Data

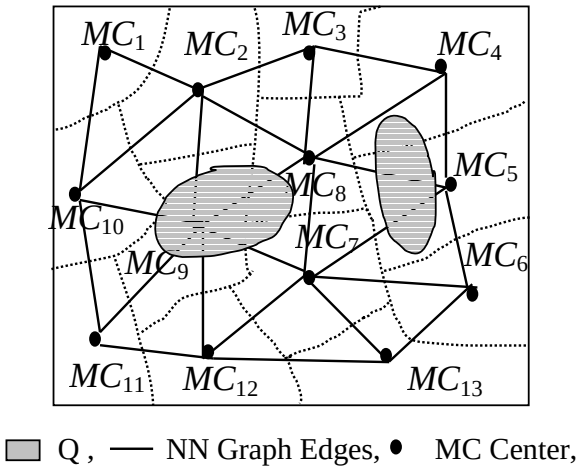


Figure 5.4: Coarse scale optimization for minimization of entropy will not be effective when the Scene is non-compact

Due to sensor inaccuracies or due to occlusion, the *Scene* is often non-compact, i.e., the *Scene* point cloud represents more than a single region on the surface of the underlying object. An example of such a scenario is illustrated in Figure 5.4. Under such circumstances, the coarse scale pruning strategy described in Section 5.3.2 will not be effective. To address such situations, we present a variation of the basic matching algorithm.

Given a non-compact *Scene*, the number of compact clusters n_c is identified using the histogram of the edge weights in $G_{MST}(Q)$. Specifically, the histogram of edge weights reveals the number of peaks and thus the number of inconsistent edges n_i. From a graph-theoretic perspective, the inconsistent edges are those edges whose weights are significantly larger than the average weight of the neighboring edges. Accordingly, the number of clusters n_c is given by $n_c = n_i + 1$. The points in the non-compact *Scene* Q are then clustered into n_c compact clusters using k-means technique i.e., Q is partitioned into $Q_1, Q_2, ..., Q_{n_c}$. With respect to each *Scene* cluster $Q_j, j = 1...n_c$, the corresponding CMC_j and NP_j are determined using the approach described in Section 5.3.2. Prior to the fine scale processing, the Scene points Q, the CMC and NP are aggregated as:

$Q = Q_1 \cup Q_2 \cup ... \cup Q_{n_c}$

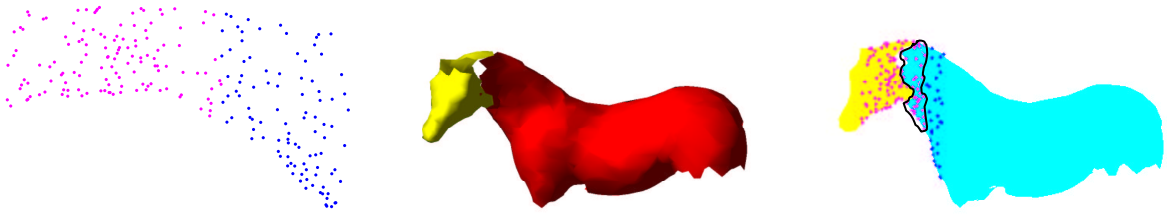
$CMC = CMC_1 \cup CMC_2 \cup ... \cup CMC_{n_c}$

$NP = NP_1 \cup NP_2 \cup ... \cup NP_{n_c}$

The fine scale processing strategy described in Section 5.3.3 is implemented for the determination of the correspondence.

5.4 Multi-part Point Cloud Matching

The algorithm described in Section 5.3 is useful for establishing a correspondence when all the points in a *Scene* are homogeneous in some sense (e.g., geometric sense). For heterogeneous *Scenes*, a part-based approach to point matching is followed wherein the *Scene* is segmented into multiple parts such that each part consists of a set of homogeneous vertices [97]. Thereafter, a



(a) Segmentation of Scene Q (b) Segmentation of Model M (c) The vertices insided the shaded region have been misclassified.

Figure 5.5: Need for the refinement of segmentation labels prior to point correspondence

point matching algorithm is used to establish correspondence between the points in every *Scene* part and the associated *Model* part. Given that the *Scene* has fewer points that the *Model* and that the geometry-based segmentation algorithms are sensitive to noise and the nature of the triangulation, the segmentation boundaries in the *Scene* are often inaccurate. Consequently, the resulting correspondence is unreliable. This is illustrated in Figure 5.5. Intuitively, a good correspondence between point sets can be obtained by refining the segmentation labels of *Scene* vertices prior to matching.

First, using a geometry-based segmentation technique [75], the *Scene* Q is segmented into j disjoint parts $Q_1, Q_2, ..., Q_j$, Then the labels of the boundary vertices of the various Scene parts are refined based on the structural and spatial similarity of the underlying *Model* points. For this, we derive the support from the thermodynamics of heterogeneous systems. During the coarse scale processing step, an entropy based minimization is performed to identify the *Model* points of interest. The fine scale processing, involves the maximization of ΔG at constant temperature T. This is because an optimal partition of a heterogeneous system into homogeneous components is obtained when ΔG reaches its maximum. The resulting correspondence allows us to re-label the boundary vertices (by using the labels of the associated Model points).

For the ease of understanding, we consider the case wherein the *Scene Q* has been segmented into two parts, Q_a, Q_b. Here, $Q_a \cup Q_b = Q$ while $Q_a \cap Q_b = \emptyset$. Note that the approach presented here can be trivially generalized to the case when three or more Scene parts share a boundary. As illustrated in Figure 5.5(c), since it is difficult to identify regions of misclassified vertices in

the case of 3D point clouds, we first consider only the vertices on the boundaries of the segments Q_a, Q_b. Depending on the fine scale refinement results (explained below), the one-connected neighbors of the re-labeled vertices are examined and if there exists an inhomogeneity in the class labels, then the coarse and the fine scale processing steps are repeated.

5.4.1 Coarse Scale Processing

To begin the coarse scale processing, the vertices $\{q_a\}$ and $\{q_b\}$ that are located on the boundaries of Q_a and Q_b respectively are identified. Using the point set $\{q_a\}$, we determine the corresponding CMC_a, NP_a from the underlying model. Similarly, we determine CMC_b, NP_b corresponding the point set $\{q_b\}$. Thus, the *Closest Model Cluster* is given by $CMC = \{CMC_a \cup CMC_b\}$ and the *neighborhood pool* is given by $NP = \{NP_a \cup NP_b\}$. CMC_a, CMC_b, NP_a, NP_b are obtained using the B&B optimization process described in Section 5.3.2.

5.4.2 Fine Scale Processing

The refinement of class labels of vertices is analogous to the change in composition of component(s) in a heterogeneous system. As mentioned earlier, an optimal partition of heterogeneous system into multiple parts can be obtained by maximizing ΔG with respect to the composition while keeping the temperature *T* and the pressure *P* constant. It is worth mentioning that basic correspondence problem (described in Section 5.3) is solved by maximizing ΔG with respect to the temperature T while keeping the pressure P and composition constant and is applicable only for homogeneous systems.

From an implementation point of view, the CMC determined during the coarse scale processing step described above forms the root node in the fine scale B&B search tree. We place a disc of radius r centered on the vertex $q_i \in (\{q_a\} \cup \{q_b\})$ and identify a set of corresponding *Model* points of interest say $V_m \subset (NP, CMC)$. The swapping and the B&B optimization strategies are similar to what is described in Section 5.3.3. Upon termination, the points in $\{q_a\}$ and $\{q_b\}$

are re-labeled based on the class labels of the corresponding *Model* vertices.

So far, we have considered only the boundary vertices during the coarse scale and the fine scale processing steps. A delaunay triangulation constructed over the points in each *Scene* part allows us to compare the class labels of the re-labeled vertices with their one-connected neighbors. If there exists an inhomogeneity in the class labels, then the coarse scale and the fine scale processing steps may be iterated by including the one connected neighbors as well. The point correspondence for each homogeneous part is then obtained using the algorithm described in Section 5.3.

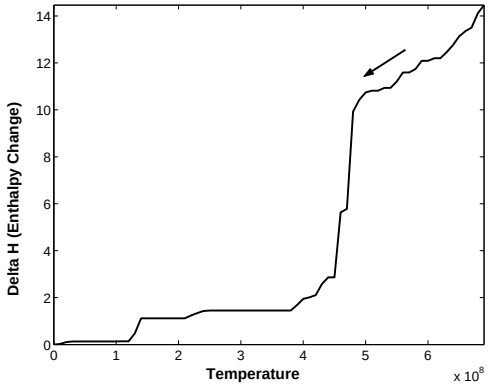
5.5 Experiments and Discussion

5.5.1 Point Matching Process: Validation of the Laws of Thermodynamics

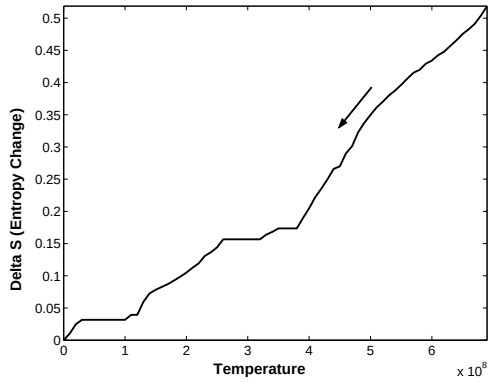
The adherence of the graph formulations of ΔH and ΔG to the laws of thermodynamics is experimentally proven by considering random 3D point sets where, $|Q| = 100$ points and $|M| = 5000$ points. Using M, Q_1, Q_2, Q_3, Q_4, Q_5 were generated, corresponding to 100%, 80%, 60% 40% and 0% overlaps respectively. The extent of overlap was decreased by adding white Gaussian noise to a subset of points in Q_1.

The graphs in Figure 5.6(a)-(c), correspond to the fine scale processing stage of Q_1, where *T* was initialized at 70×10^7, to indicate that 30 vertices from coarse scale *CMC* overlapped with Q_1. For the fine scale processing of Q_3 (graphs shown in Figure 5.6(d)-(f)), *T* was initialized at 88×10^7 (since 12 vertices in coarse scale *CMC* overlapped with Q_3). Such temperature initializations to large values ensured that $\Delta G \leq 0$ throughout the correspondence process.

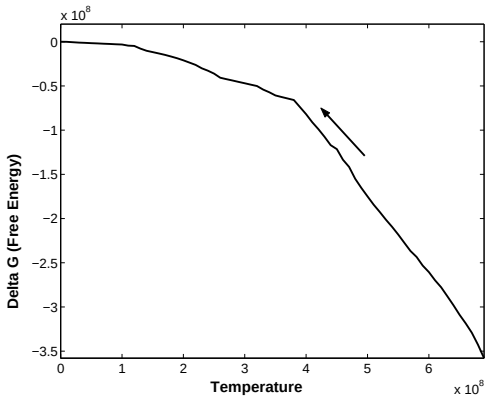
With all the point sets , it is observed that, ΔH as well as ΔS decrease with decrease in temperature, while ΔG increases toward zero. For Q_1 and Q_3, this is indicated by the direction of the arrows in Figure 5.6. As shown in Figure 5.6(a)-(c), for completely overlapping point sets, the desired correspondence is achieved when $\Delta H, \Delta S, \Delta G$ are all zero. For non-overlapping



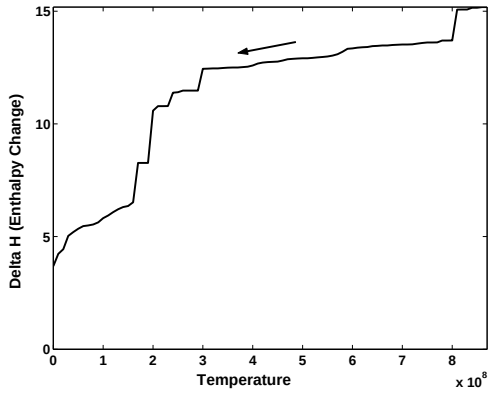
(a)ΔH: completely overlapping point sets



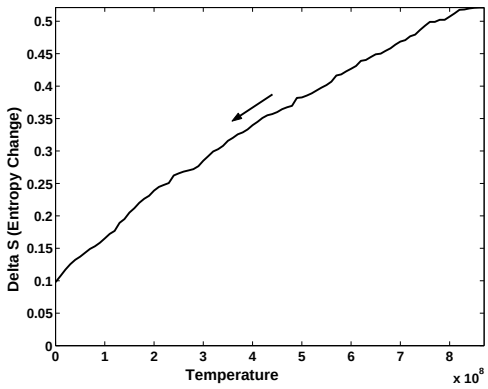
(b) ΔS: completely overlapping point sets



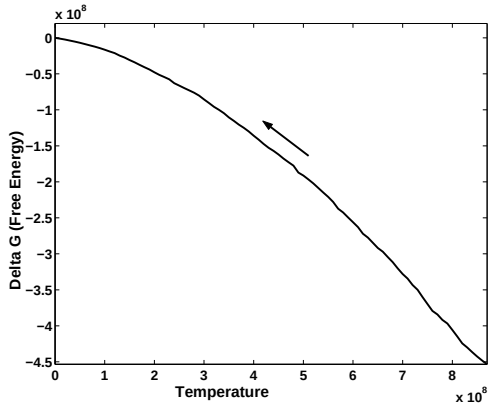
(c)ΔG: completely overlapping point sets



(d) ΔH: partially overlapping point sets



(e) ΔS: partially overlapping point sets



(f) ΔG: partially overlapping point sets

Figure 5.6: Variation of $\Delta H, \Delta S, \Delta G$ with temperature for different types of point sets

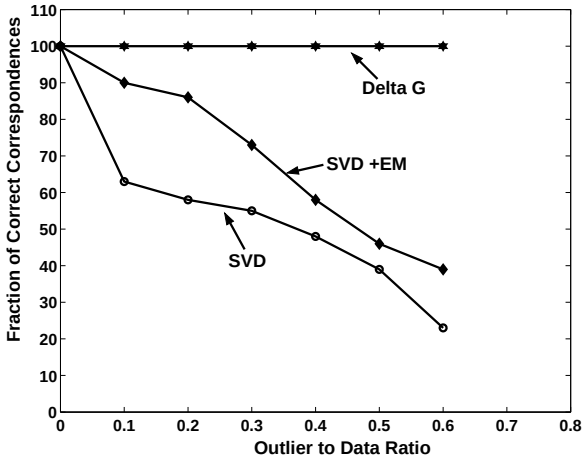
or partially overlapping point sets, the correspondence is recovered when ΔG reaches its maximum as shown in Figure 5.6(f). In Figure 5.6(f), at $T = 0$, the value of ΔG is dictated only by ΔH, implying that, among the possible matches, all with minimum ΔS, the one that minimizes the structural differences (with the least ΔH) provides the desired correspondence. These results experimentally prove the feasibility of the point matching process based on the laws of thermodynamics.

5.5.2 Comparison with an Existing State-of-the-art

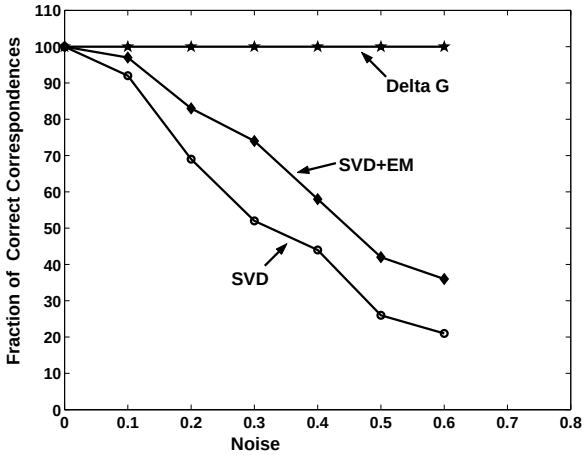
For comparison purposes, we implemented the SVD+EM [84] as well as the basic SVD algorithm [22], since they incorporate spectral graph theoretic ideas as well. An increasing weighted proximity matrix and a Gaussian weighted proximity matrix is generated for the SVD+EM [84] and the SVD approaches [22], respectively. These proximity matrices are then used to obtain corresponding modal matrices. In the SVD approach [22], a binary decision on the correspondence is made on the basis of the similarity of different rows of the modal matrices for the two point sets. For the SVD+EM approach [84], using the modal matrices, the probabilities are computed to assess the similarities between the elements of the point sets. The correspondence process is embedded within the EM framework.

Two sets of experiments on random same-sized 3D point sets were conducted to compare the algorithms. In the first series of experiments, we added "extra" *Model* points, which in [84] are termed as outliers. The outlier to data ratio ranged from 0 to 0.8. Although the performance of the SVD+EM algorithm is better than SVD approach, as Figure 5.7(a) indicates, it still is quite sensitive to the presence of outliers. Since our algorithm is designed to deal with outliers, a 100% correspondence is always achieved.

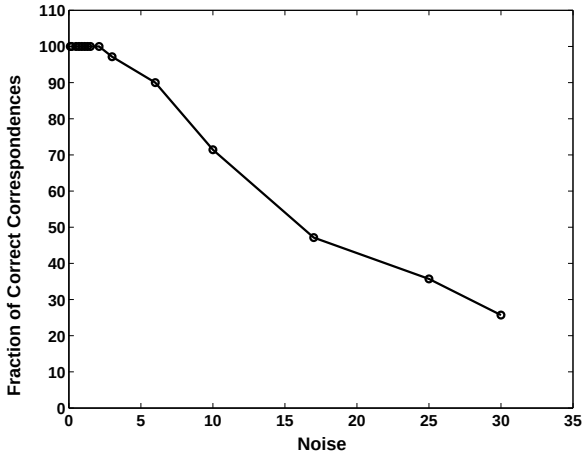
In the second series of experiments, our goal was to analyze the algorithms' tolerance to noise. To begin with, we considered completely overlapping, same -sized point sets. The scene point cloud was then progressively subjected to varying levels of Gaussian noise (standard deviation



(a) Effect of Outliers



(b) Effect of Noise



(c)Our algorithm's performance with considerable amounts of noise

Figure 5.7: Performance comparison between the proposed point matching algorithm and the existing state-of-the-art

ranging from 0 to 0.6). As Figure 5.7(b) indicates our algorithm provides 100% correspondence while the performance of the spectral correspondence algorithms degrade with increase in noise. As Figure 5.7(c) indicates, considerable positional jitter (manifested by large standard deviations of noise) is required for the performance of our algorithm to degrade.

5.5.3 Real Data Sets

We evaluated the performance of our algorithm on 15 real data sets. In Figure 5.9(a), $|M| = 4370$ points, $|Q| = 200$ and the *NP* consists of 4 model clusters (as opposed to 22 clusters that characterize the entire model space). Prior to the implementation of the fine scale B&B algorithm, 12% of the *CMC* points overlap with the scene. The desired correspondence between *Q* and the *CMC* is shown in Figure 5.9(b).

Figure 5.9(c) shows a more challenging example where the point sets are non-overlapping. The model *M* is a tank, $|M| = 18,897$. The scene *Q* consists of 300 points. Of the 63 *MC*s that describe the entire model space, only 6 *MC*s constitute the *NP*. Since none of the coarse scale *CMC* points exactly overlapped with *Q*, the temperature at the start of fine scale processing was set at 300×10^7. Figure 5.9(d) shows the desired correspondence obtained between *Q* and the refined *CMC*.

5.6 Conclusions

Our results indicate that (i) the newly proposed formulation of graph enthalpy efficiently captures the structural differences between non-overlapping point sets (ii) the Gibbs' free energy based optimization, by combining the spatial and the graph-based structural information, leads to stable and efficient matches, as opposed to simple graph matching. Additionally, the proposed approach is highly robust in the presence of noise and can handle missing data effectively. For $|Model| = 5000$ points and $|Scene| = 100$, our algorithm determines the desired correspondence

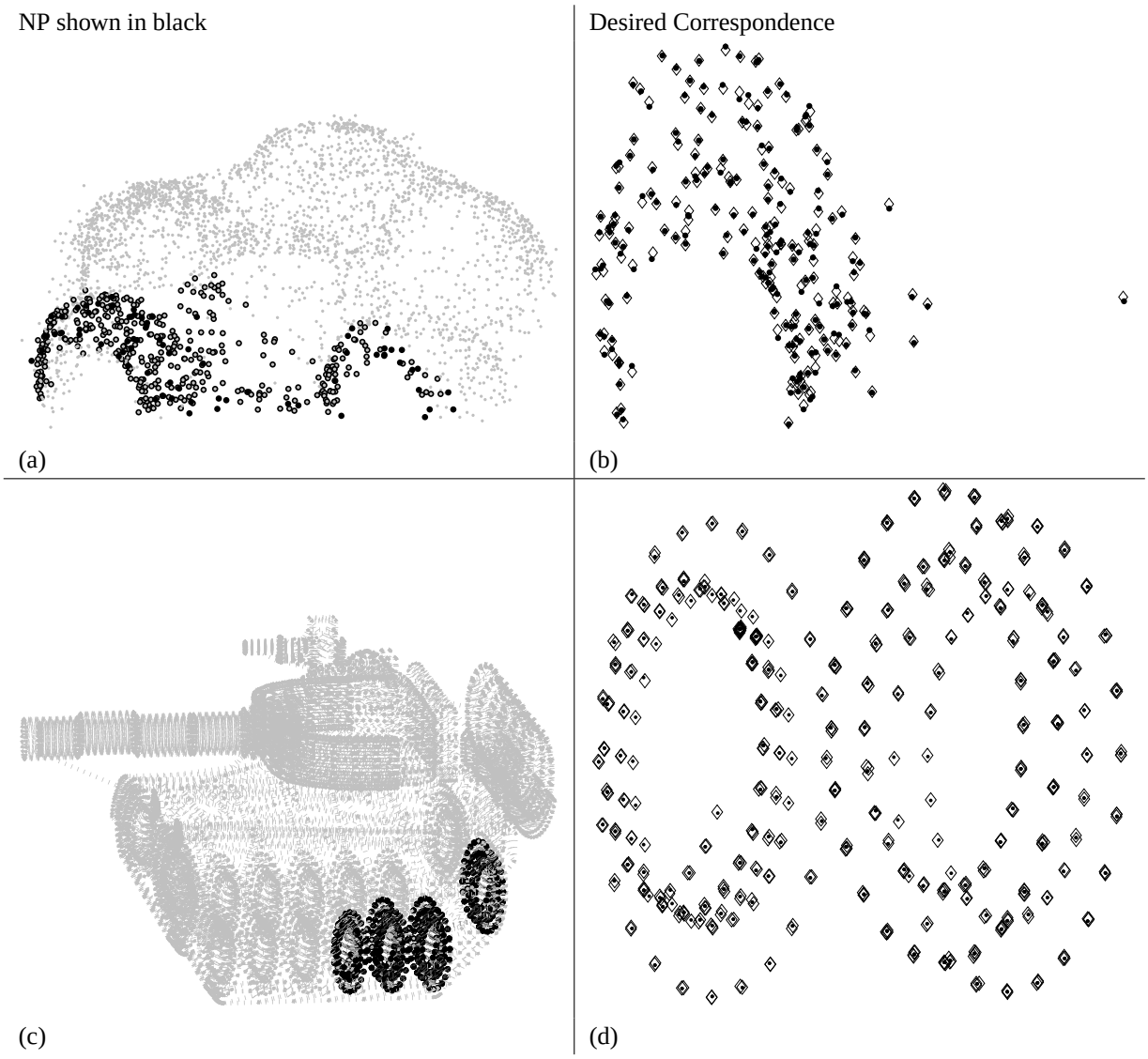


Figure 5.8: In (a),(c) the model points are indicated by gray dots while the points in the NP are shown in black. (b),(d) indicate the desired correspondence, where diamonds and the dots correspond to Q and refined CMC

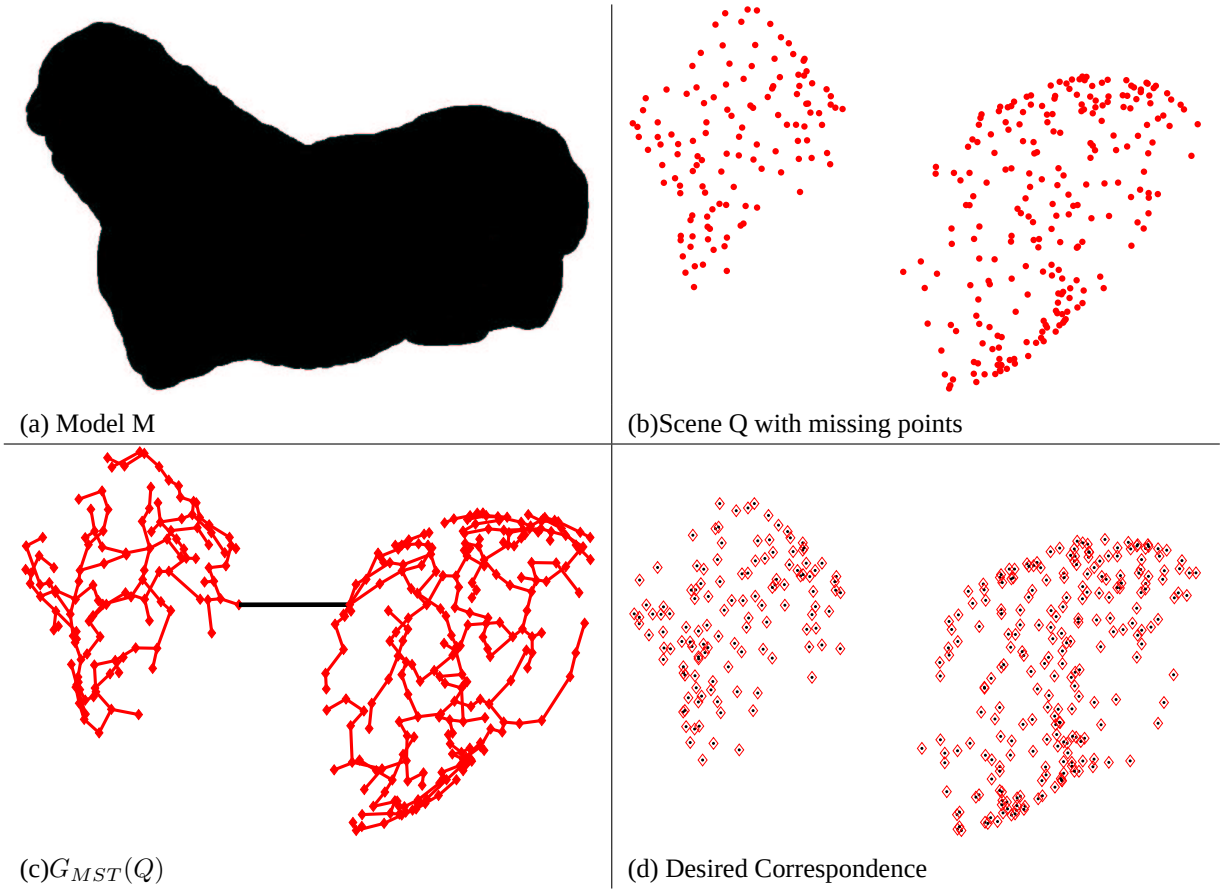


Figure 5.9: In (c) the inconsistent edge is indicated by the black line (d) indicates the desired correspondence, where diamonds and the dots correspond to *Q* and refined *CMC*

in approximately 1.2 minutes on a Pentium IV, 256MB memory, 1.5 GHz machine (with the code implemented in Matlab).

Chapter 6

Hierarchical Mesh Decimation for Multi-scale Correspondence

The point clouds of the models used to build the database are in some sense extremely over-sampled. Due to this, the stored model representations for a nontrivial number of objects require large amounts of disk space. This can also have a detrimental effect on recognition as one looks to real-time or close to real-time applications. To alleviate these problems, we are motivated to consider approximations of these finely sampled models which can be obtained through a process called mesh decimation.

For the results of mesh decimation to be useful for the recognition, there are two fundamental issues that need to be addressed. First, the geometric shape information captured by the fine scale mesh must be preserved at all coarser resolutions as well. Secondly, any graph-based decimation technique must avoid the computation of eigenvalues of the graph's Laplacian for such a computation can be prohibitively expensive for very large (of the order of 10^5 vertices) graphs [113, 94].

In Section 6.2 of this chapter, a graph energy based cost function is proposed for minimization within the framework of hierarchical edge contraction. The shape information is preserved by performing a curvature based classification of the vertices prior to decimation. In Section 6.4, we relate hierarchical decimation with multi-scale correspondence and propose a decimation metric

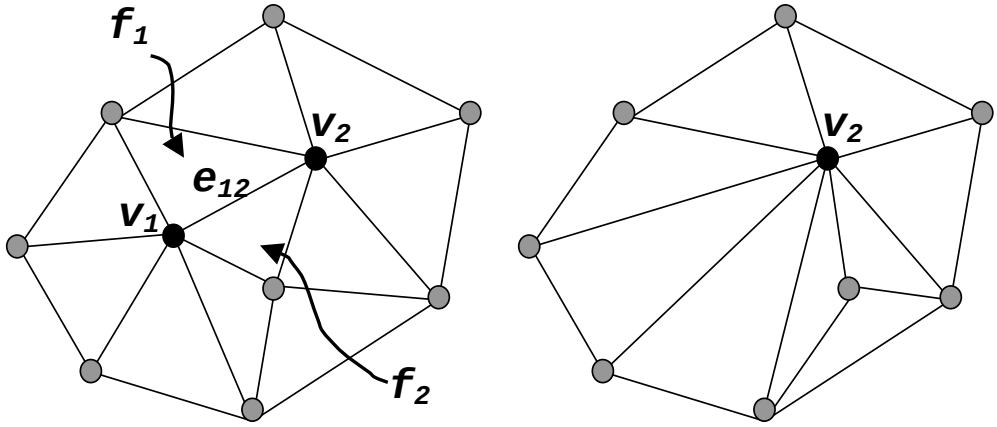


Figure 6.1: Vertex Contraction Process. When v_1 is merged with v_2 then the following happen (1) the edge e_{12} contracts, (2) triangular faces f_1, f_2 that are now degenerate, are removed, (3) the edges that were originally incident on v_1 are now incident on v_2. Vertex contraction affects the geometry and the topology of a surface

that captures the information about the degradation in correspondence as a result of decimation at multiple coarser scales.

6.1 Vertex Contraction

As illustrated in Figure 6.1, a vertex pair contraction modifies the surface in three steps [76]:

- A vertex v_i is *merged* with vertex v_k by replacing all occurrences of vertex v_i with vertex v_k.
- All the triangular faces that are now degenerate - that no longer have three distinct vertices- are removed.
- Edges from vertices that were originally incident on v_i are now incident on v_k.

The first step modifies the geometry of the surface. The second step simply removes the elements of the surface that are no longer needed. The final step modifies the connectivity of the mesh and in a sense, implicitly modifies the topology of the surface e.g., by closing the holes.

Most of iterative contraction algorithms [59, 42, 45, 62, 65], follow a *greedy* approach to select the sequence of edge contractions. Each vertex pair being considered for merging is assigned a cost which, typically represents the error induced as a result of a potential merging of the vertex pair in question. At each iteration, the lowest cost pair is merged. A fundamental advantage of iterative contraction is the hierarchical structure that it induces on the surface, thus leading to a multi-resolution surface representation.

Although the literature indicates the existence of general vertex pair contraction algorithms where the vertices v_i, v_k are not necessarily connected by an edge, in this work we are specifically interested in merging adjacent vertices with similar shape characteristics. Further we consider subset placements (where $v_i, v_k \in V(G)$) as opposed to optimal placements (where v_i, v_k are both moved to another optimal position $\bar{v}$ not necessarily in $V(G)$).

6.2 Graph-based Vertex Contraction Algorithm

6.2.1 Motivation

Toward formulating a cost function for vertex contraction, we are motivated to use ideas from spectral graph theory, which have been used for drawing *good* or *natural* graphs and in a different context, for the construction of stable physical mass models. In a natural graph (stable physical model), the vertices (masses) are connected by the edges (springs) that are minimally stretched [92], thus leading to the following theorems.

Theorem 6.1: Given a vector x and the graph's Laplacian L, $x^T Lx = \sum_{e_{uv} \in E(G)} ||x_u - x_v||^2$

Theorem 6.2 [92]: The graph energy $\mathcal{E}(G)$ is defined as $\mathcal{E}(G) = traceR^T LR$ where L is the weighted Laplacian and the *representation* R is a $(|V(G)| \times m)$ matrix that provides a mapping from V into $\Re^m$.

In this work, we perform mesh decimation by merging a pair of adjacent vertices that minimizes the change in graph energy between the fine scale mesh and its approximation. Clearly,

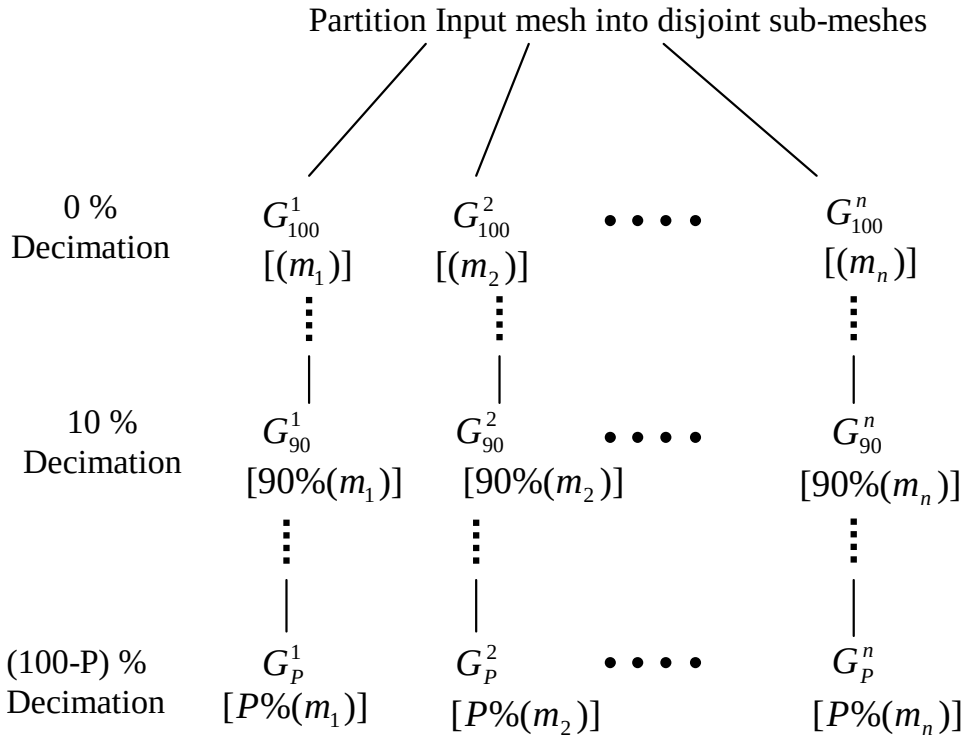


Figure 6.2: Illustration of the proposed hierarchical vertex contraction process. An input fine scale 3D mesh is partitioned into various disjoint sub-meshes. At any level in the hierarchy, every sub-mesh undergoes the same rate of decimation. In any sub-mesh, the vertex pair for contraction is identified based on the minimization of spectral graph energy. Finally, all the decimated sub-meshes are merged resulting in a shape preserving coarse scale approximation

such an approach avoids the prohibitively expensive computation of eigenvalues of the very large (of the order of 10^5) Laplacian matrices which is followed by some existing graph-based approaches [76]; only the computation of the eigenvalues of the (3×3) matrix $R^T LR$ is needed.

6.2.2 Proposed algorithm

Figure 6.2 provides an overview of the proposed algorithm. Let the segmentation of a fine-scale 3D mesh result in multiple, disjoint sub-meshes. At 100% resolution, the ith sub-mesh is denoted by SM^i_{100}. A hierarchical vertex contraction of SM^i_{100} involves:

- Computation of the sub-mesh energy $\mathcal{E}(SM^i_{100})$.

- Iterative Step: for $l = 99, 98, ...,$ perform the following:

  Step 1: Determination of boundary and interior vertices using Shape index: Recall that the shape index $S(v)$ [24] at a vertex $v \in SM^i_{l+1}$ is given by

$$S(v) = -\frac{2}{\pi}\tan^{-1}\frac{\kappa_{max}(v) + \kappa_{min}(v)}{\kappa_{max}(v) - \kappa_{min}(v)}$$

  where κ_{max} and κ_{min} are the principal curvatures of the surface at the vertex v and described in detail in Section 4.1.1, Chapter 4.

  Next,the shape thresholds are determined using a shape histogram and the shape scale illustrated in Figure 4.1. The number of vertices whose shape index value falls in the jth bin ($1 \leq j \leq 9$) is given by $n_j = \sum_1^{N_i} \chi_k(S_{v_i})$. Here, χ is the characteristic function of the jth bin:

$$\chi_k(x) = \begin{cases} 1 & t_{j-1} \leq x < t_j \\ 0 & otherwise \end{cases} \tag{6.1}$$

  and $t_0 = -1, t_9 = +1$ and the other shape thresholds t_i [24] are indicated on the shape scale in Figure 4.1.

  The region growing approach [100] is implemented to aggregate clusters containing vertices of the same shape type. Two vertices v_i, v_k belong the same shape cluster if (a) their shape indices $S(v_i), S(v_k) \in [t_{j-1}, t_j]$, and (b) there exists a path *p* from v_i to v_k containing distinct vertices $v_0, v_1, ..., v_m \in V(SM^i_{l+1})$, such that condition (a) holds true for every pair of vertices along the path *p*.

  Next, the vertices in every shape cluster are labeled as *boundary* or *interior* depending on whether they are located on the boundary/interior of the cluster. To ensure that the resulting approximation is shape preserving, only the similar-labeled adjacent vertices are considered as candidates for a potential merge during vertex contraction.

Step 2: Vertex Merge Pair Determination: Given the basic vertex contraction framework (illustrated in Figure 6.1), for every vertex $v_k \in SM^i_{l+1}$, $k = 1, 2, ..., |V(SM^i_{l+1})|$, a similar-labeled neighbor (boundary/interior), say v_r is identified that minimizes the energy difference between SM^i_{l+1} and the approximation resulting from the potential merge (v_k, v_r). The index of such a neighbor of v_k is given by

$$r = \underset{v_n \in N(v_k)}{\arg\min} |\mathcal{E}^i_{l+1} - C_{nk}| \tag{6.2}$$

C_{nk} is the energy of the graph obtained by merging the adjacent vertex pair (v_n, v_k). Of the potential vertex merge pairs (v_k, v_r) determined above, vertex contraction is implemented by using the pair that provides the closest approximation to SM^i_{l+1}.

Step 3: Energy Update: The energy of the contracted sub-mesh SM^i_l is given by $\mathcal{E}^i_l$.

6.3 Evaluating Surface Approximations

Shape differences between a model and its approximation are evaluated using application dependent error metrics such as the Haussdorff distance [60]. In this work, the quality of an approximation is assessed based on the misclassification error of the vertices involved. For this, the decimated sub-meshes at any level in the hierarchy are re-triangulated to obtain an approximation of the input 3D mesh. The approximated mesh is then segmented using the algorithm described in Chapter 4 and the number of misclassified vertices are accounted for.

6.4 Relationship between Hierarchical Decimation and Multi-scale Correspondence

6.4.1 Decimation: A Thermodynamic Viewpoint

In Section 6.2.2, a graph-theoretic approach to mesh decimation was presented. In this section, a thermodynamic interpretation is provided which is applicable to other iterative vertex contraction approaches [60, 42, 45, 62, 65] as well.

Let $l\%$ decimation of the fine scale *Model* M^1 result in the coarse scale approximation given by M^l. In order to compute the enthalpy and the Gibbs free energy for the *Model* at the coarser scale, we need to determine the change in pressure as a consequence of decimation.

From chemical thermodynamics, at constant volume, and for a constant number of particles n

$$P \propto T \tag{6.3}$$

where P, T are the pressure and the temperature respectively.

For our problem, let us denote the fine scale temperature and pressure by T_d^1 and P_M^1 respectively. Specifically, we assume $T_d^1 = 1$ and $P_M^1 = 1$. The pressure resulting from $l\%$ decimation can be determined as:

$$P_M^l = P_M^1 \frac{T_d^l}{T_d^1} \tag{6.4}$$

Here, $T_d^l = (1 + l\%)T_d^1$. Observe that as the rate of decimation (or decimation temperature T_d^l) increases, the pressure P_M^l increases as well. Also, the conclusions drawn about the pressure and temperature are independent of the decimation approach used. Thus, our definitions for pressure, volume and temperature are consistent with those provided in classical thermodynamics wherein these parameters are treated as state variables; the path to the state is immaterial.

6.4.2 Formulations for Enthalpy, Entropy and Free Energy at Multiple Scales

The definitions for enthalpy, entropy and Gibbs free energy for point sets at a single scale was provided in the previous chapter. Here, we extend these definitions to multiple scales. Enthalpy for a certain model cluster MC^l obtained from M^l is computed at pressure P_M^l. The pressure in the *Scene* points is given by P_Q. We assume that $P_Q = P_M^1$. The pressure for the point set $Q \cup MC^l$ is given by $P_{QM}^l = \frac{P_Q + P_M^l}{2}$.

Given the point sets Q and MC^l, the change in enthalpy ΔH^l at the level l is computed as by $\Delta H^l = \Delta H_1^l + \Delta H_2^l$ where

$$\Delta H_1^l = H(G_{U1^l}) - H(G_{MST}(Q)) \tag{6.5}$$
$$\Delta H_2^l = H(G_{U2^l}) - H(G_{MST}(MC^l))$$

Here, G_{U1^l} denotes the union of $G_{MST}(Q, E_Q)$ and $G_B((Q, MC^l), E_{QMC^l})$. G_{U2^l} denotes the union of $G_{MST}(MC^l, E_{MC^l})$ and $G_B((Q, MC^l), E_{QMC^l})$.

The change in entropy ΔS^l for the two sets of points Q and MC^l, is computed as

$$\Delta S^l = \Delta S_1^l + \Delta S_2^l \tag{6.6}$$

where,

$$\Delta S_1^l = \hat{S}(Q \cup MC^l) - \hat{S}(Q)$$
$$\Delta S_2^l = \hat{S}(Q \cup MC^l) - \hat{S}(MC^l) \tag{6.7}$$

The computation of the free energy ΔG^l for same-sized point sets Q and MC^l is then straightforward.

6.4.3 Performance Evaluation across Multiple Scales

Using the basic correspondence algorithm described in the previous chapter, one can determine the correspondence between a subset of M^l and the Scene Q. Then, the goal is to evaluate how the correspondence obtained at a certain coarse scale l compares with the correspondence obtained at the finest scale.

Following the thermodynamics of phase partitions and the clapeyron equation, we can say that the decimation pressure and the correspondence temperatures are related as :

$$\frac{P^l_{QM} - P^1_{QM}}{T^l_C - T^1_C} = \frac{\Delta_M S}{\Delta_M V} \tag{6.8}$$

where P^l_{QM} is the pressure in the point set $(Q \cup M^l)$. $P^1_{QM} = 1$. T^l_C is the correspondence temperature at the level l. T^1_C is set to a large constant. In this work, $T^1_C = 1 \times 10^6$ $\Delta_M S$ is the change in entropy computed over the point sets CMC^l and Q and $\Delta_M V$ is the associated change in volume. Here, $CMC^l \subset M^l$ is the subset of model points that provides the optimal correspondence at level l and is determined using the algorithm described in Section 5.3.

Motivated by Carnot's efficiency metric, in this work, the extent of correspondence degradation as result of decimation is given by the degradation metric η which is defined as

$$\eta = \frac{T^l_C - T^1_C}{T^l_C} \tag{6.9}$$

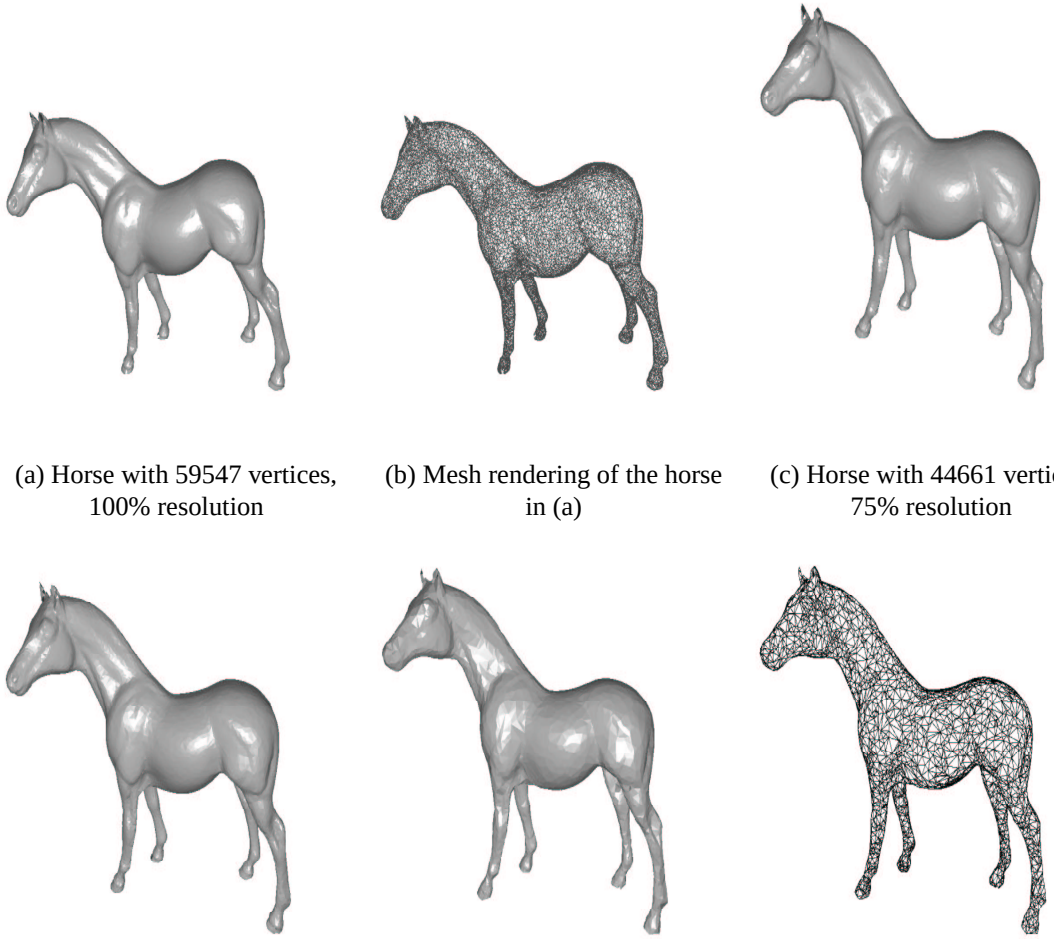
6.5 Results and Discussion

6.5.1 Graph-based Mesh Decimation

The proposed decimation algorithm was coded in Matlab and tested on a Pentium IV processor at 1.5GHz and 256MB memory. Decimation results for the fine scale mesh of a horse with 59547 vertices are shown in Figure 6.3. The horse mesh was segmented into 8 sub-meshes

object	Number of vertices	Number of sub-meshes	Time (in minutes) for 90% decimation
tea cup	11241	5	30.4
horse	59547	8	189.9
Car	7401	19	21.2
Pickup Truck	4902	15	13.7
Pig	4332	11	12.3

Table 6.1: Timing Performance of the proposed decimation algorithm on various data sets



(a) Horse with 59547 vertices, 100% resolution

(b) Mesh rendering of the horse in (a)

(c) Horse with 44661 vertices, 75% resolution

(d) Horse with 29773 vertices, 50%resolution

(e) Horse with 14886 vertices, 25% resolution

(f) Mesh rendering of the horse in (e)

Figure 6.3: Horse Data

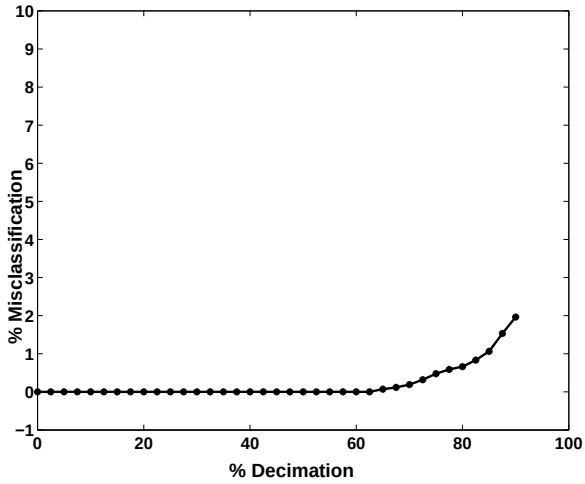


Figure 6.4: For horse mesh shown in Figure 6.3, an approximation is obtained by re-triangulating the decimated meshes at that level. This approximated mesh is segmented into multiple disjoint sub-meshes and the number of misclassified vertices are determined (relative to the sub-meshes at the finest scale).

using the segmentation algorithm described in Chapter 4. Each sub-mesh was subjected to the same rate of decimation. For example, the approximation at 75% resolution was obtained by decimating all the fine scale sub-meshes by 25%. The coarser approximation was then obtained by re-triangulating the decimated sub-meshes. Note that from a representation standpoint, such a re-triangulation is not necessary. The timing performance of the algorithm on various datasets is shown in Table 6.1. The graph in Figure 6.4 confirms the proposed algorithm's ability to produce the desired shape-preserving coarse approximations.

6.5.2 Hierarchical Decimation Vs. Multi-scale Correspondence

At the finest scale, the simulated Model and Scene point clouds ($|M| = 5000, |Q| = 100$) were used to identify the model points comprising the neighborhood around the scene. To evaluate the effect of decimation on correspondence, we performed two separate experiments. In the first case, the points outside the fine scale model's NP were decimated. In the second case, points inside the fine scale model's NP were decimated. The graph relating the decimation pressure with correspondence temperature is shown in Figure 6.5. From the graph, it can be seen that

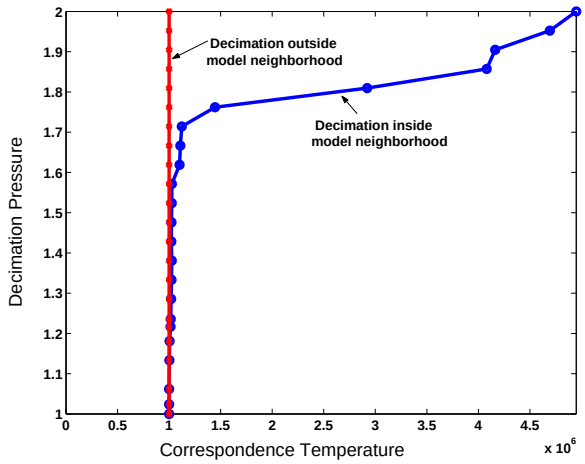


Figure 6.5: Relationship between decimation pressure and correspondence temperature

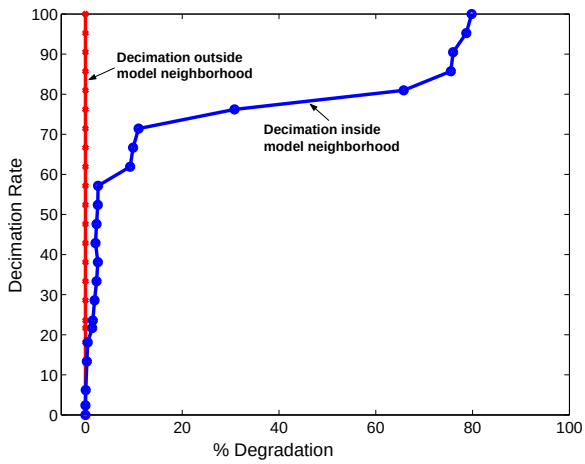


Figure 6.6: Rate of decimation vs. % degradation: Extent of degradation as a result of decimation

decimation outside the model's NP does not affect the correspondence i.e., although the pressure changes as a result of decimation, the correspondence temperature does not change. On the other hand, decimation of points inside the model's NP tends to degrade the correspondence. The decimation pressure and the correspondence temperatures are related to the rate of decimation and the % degradation (as captured by the degradation metric η) respectively and the corresponding graph is provided in Figure 6.6. As shown in this graph, the quality of the desired correspondence degrades (as manifested by an increase in % degradation) with increase in the rate of decimation.

6.6 Conclusions

In this chapter, a new mesh decimation algorithm is proposed to address the problem of obtaining shape-preserving coarser approximations of highly detailed 3D surface meshes. The input mesh is segmented into multiple, disjoint sub-meshes to facilitate decimation. Given a sub-mesh, various shape clusters are identified and the vertices in those clusters are labeled as boundary/interior. Shape is preserved by considering only similar-labeled vertex pairs as candidates for a potential merge. Sub-mesh decimation is realized by merging a vertex pair that minimizes the proposed graph energy based cost function. Low misclassification error on various datasets indicate the algorithm's ability to produce shape-preserving approximations. De-coupling of the input mesh into corresponding sub-meshes prior to the hierarchical approach to decimation suggest that parallel implementations of the algorithm can provide a significant computational speedup.

A thermodynamic interpretation for hierarchical decimation allows for the analysis of the relationship between decimation and correspondence. The degradation in the desired correspondence as a result of decimation is captured by the proposed formulation for the degradation metric. Results indicate that % degradation increases with increase in the rate of decimation.

Chapter 7

Conclusions and Future Work

In the context of object recognition from 3D point cloud data, this dissertation presented solutions to two fundamental problems (1) segmentation of surface meshes toward deriving an efficient representation of the underlying object (2) determination of a one-to-one correspondence between a partial Scene and a complete Model point cloud. A robust solution to the segmentation problem was obtained by considering manifold surfaces meshes as input. Typically, the construction of surface meshes over 3D point clouds using commercially available software induces topological "bugs" in the triangulation, thereby causing the triangulation to be non-manifold. In this dissertation, a simple approach to converting a non-manifold triangulation into the corresponding manifold mesh was presented in Chapter 3. In addition, a new mesh decimation algorithm was presented in Chapter 5 which aided in evaluating the relationship between hierarchical decimation and multi-scale correspondence. The major and the minor contributions of this thesis are summarized below.

7.1 Major Contributions

- Mesh segmentation: In Chapter 4, in the context of object representation, a graph morphology based segmentation algorithm was presented to partition an input 3D manifold surface

mesh into disjoint sub-meshes corresponding to the different parts of the underlying object. Given a manifold mesh, curvedness, a curvature based shape descriptor, is computed at every vertex in the triangulation and it serves as a similarity metric for segmentation purposes. A sub-mesh consists of a set of vertices whose curvedness values are in a certain range as specified by a pair of curvedness thresholds. Such curvedness thresholds are determined using a robust technique which combines ideas from 2D histogram based processing and k-means based clustering. The extraction of a certain sub-mesh is an iterative two-step morphological process which involves (a) dilation and morphological filtering of vertices (b) attributed graph matching of the *dilated* graph with a *desired* graph.

The performance of the algorithm was tested using simulated point clouds as well as using surface meshes of objects that are well accepted test cases in the fields of machine vision and computer graphics. Results indicate that graph dilation together with morphological filtering of outliers can effectively deal with noise, thereby avoiding the need for any preprocessing step to deal with noisy point clouds. Also, it is observed that the selection of adaptive curvedness thresholds leads to robust segmentation. The algorithm compares well with the existing state-of-the art approaches and provides robust segmentations for a wide variety of objects.

- Point cloud matching: In Chapter 5, a thermodynamically-inspired graph theoretic algorithm was presented to address the problem of establishing a one-to-one correspondence between the scene and the identified model point clouds, when the cardinalities of the two sets are orders of magnitude different. Such an approach determines a subset of points from the model that is structurally and spatially as similar as possible to the set of points in the scene. A new formulation for graph enthalpy characterizes the structural differences between point sets, which together with the existing notions of graph entropy quantifies the Gibbs' Free Energy. A two-scale approach is proposed, wherein, at the coarse scale, a set of

points that comprise the model neighborhood around the scene is identified by minimization of entropy. At the fine scale, the desired correspondence was achieved by a refinement process, aimed at maximizing the Gibbs' Free Energy. In order to deal with missing data in the scene, a variation of the basic matching algorithm is presented. In the context of part-based correspondence, ideas from thermodynamics of heterogeneous systems were used to refine segmentation labels prior to correspondence.

The proposed definitions of graph enthalpy and Gibbs free energy were validated using random 3D point sets and it was observed that these definitions abide by the laws of classical thermodynamics. Extensive experiments on real data indicate that (i) the newly proposed formulation of graph enthalpy efficiently captures the structural differences between non-overlapping point sets (ii) the Gibbs' free energy based optimization, by combining the spatial and the graph-based structural information, leads to stable and efficient matches, as opposed to simple graph matching. Additionally, the proposed approach is highly robust in the presence of noise and can efficiently deal with missing data.

7.2 Minor Contributions

- Conversion from non-manifold to manifold surface meshes: In order for a surface mesh to be used by the proposed segmentation algorithm, it must be void of any topological singularities, i.e., the surface mesh must be manifold. Often times however, the use of commercially available software for the triangulation of point clouds induces 'bugs' that cause the triangulation to be non-manifold. In Chapter 3, a greedy surface growing algorithm is presented to convert a three-dimensional non-manifold surface triangulation into the corresponding manifold surface mesh by employing topology as well as geometry-based constraints. The algorithm specifically addresses the case when there are a large

number of singular edges present in the triangulation and does not consider sharply convex/concave surfaces. The singular edges are identified as those edges along which more than two triangles incident. The region growing process is initialized by identifying a seed triangle that satisfies certain topological and geometric constraints. The algorithm subsequently breaks the non-manifold triangulation into its constituent triangles. A manifold triangulation is then grown by stitching triangles that minimize a dihedral angle-based cost function. A number of non-manifold triangulations were converted into the corresponding manifold surfaces using the proposed algorithm. It was observed that the resulting manifold surface efficiently captured the underlying object's geometry. Due to the propagating nature of the algorithm, this mesh repair process needs to be done off-line.

- Mesh Decimation: The realization that the derived representations of very finely sampled models not only require a large amount of storage space but also cause a significant slow down during recognition, motivated us to propose a graph-theoretic mesh decimation algorithm in order to obtain shape preserving coarser approximations of the given fine scale input mesh (Chapter 6). For this, the fine scale mesh is segmented into disjoint parts using the mesh segmentation algorithm proposed in chapter 4. In the proposed hierarchical decimation approach, every sub-mesh undergoes the same rate of decimation and the vertex pair that minimizes a certain graph energy based cost function is considered as the best pair for merging/contraction. The shape information is preserved by performing a curvature based classification of the vertices (in a submesh) prior to decimation. At a certain level in the hierarchy, the quality of an approximation is assessed by re-triangulating the decimated sub-meshes and by computing the the misclassification error. The performance of the algorithm was tested using a number of fine scale meshes. A low misclassification error suggests that the proposed algorithm preserves the shape of the underlying object at various coarser scales.

7.3 Ongoing Work

In Chapter 5, Section 5.4, in the context of part-based matching, we described the need for refinement of segmentation labels prior to the determination of one-to-one correspondence between the scene parts and the associated model parts. Inspired by the thermodynamics of heterogeneous systems, an algorithm was presented to refine segmentation labels of the scene that involved the maximization of the Gibbs free energy. As part of ongoing work, extensive experiments are being conducted to understand how misclassification affects the performance of the algorithm in terms of the accuracy of the correspondence obtained when the scene point cloud is subject to varying levels of noise. The algorithm is also being tested on scenes containing multiple objects, wherein each object is characterized as a homogeneous part.

In Chapter 6, we presented a relationship between hierarchical mesh decimation and multiscale correspondence by borrowing ideas from classical thermodynamics. A finely sampled model point cloud was subject to varying rates of decimation. A one-to-one correspondence was then established between a certain coarse model and the input scene. Although subjective, such an analysis demonstrates that coarser approximations of an input model are sufficient to obtain the desired correspondence. From a thermodynamic standpoint,while on the one hand, we established a link between different values of pressure and varying rates of decimation, on the other hand, we established a link between temperature and desired correspondence obtained at multiple scales. It was observed that with increase in the model pressure (manifested by increasing rates of decimation), the accuracy of scene-model correspondence decreases. The conclusions drawn are independent of the decimation algorithm used. As part of ongoing work, sensitivity analyses are being conducted to evaluate the performance of the algorithm under the conditions of noise. Also, effect of model decimation on multiscale correspondence for scenes with missing data and clutter is being evaluated.

7.4 Future Work

- Selection of shape descriptors: While the segmentation algorithm described in Chapter 4 provides good segmentation results for smooth man-made objects, it tends to over-segment textured surfaces. Hence, for the mesh segmentation algorithm to work well on a larger variety of surfaces, textural information needs to be incorporated into the algorithm, in addition to the existing shape descriptors. Also, while the existing shape descriptors are useful in distinguishing objects such as a pick-up truck and a horse, they are not as useful to discriminate two similar objects for example a Toyota Corolla vs. a Honda Civic. For such problems, additional shape descriptors need to be considered, while keeping in mind the curse of dimensionality.

- Extension of thermodynamic ideas for many-to-many graph matching and deformable point matching: Arguably, deformable point matching is a much difficult problem. The thermodynamic ideas described in this work are useful for rigid point matching. For the purposes of deformable point matching, the formulation for graph enthalpy needs to utilize graph structures other than those considered in this dissertation. In this regard, it may be advantageous to include shape based ideas such as shape context [108]. Many-to-many graph matching is also a challenging problem of interest to the computer vision community.

- Thermodynamics for computer vision: In this dissertation, ideas from classical thermodynamics were used to solve the correspondence problem. The formulation for graph enthalpy encoded the structural or topological information in the point sets. Physicists have established a relationship between concepts in differential geometry such as curvature and the thermodynamics [3]. The question then is: is it possible to apply geometrical thermodynamics to computer vision? If yes, to what problems and what are the advantages thereof?

- Performance Evaluation: In this dissertation, we considered point cloud matching as an independent problem by assuming efficient solutions to the classification and the alignment problems. Results presented in Chapter 6 indicate that robust desired correspondence is obtained using the proposed thermodynamic formulation to point cloud matching. Toward designing practical systems, however, the rate of false positives/false negatives, or in general the receiver operating characteristic (ROC) curve under various sensitivity conditions, can best be evaluated by combining the performance of every step of representation and recognition.

- Algorithm evaluation for efficient implementations: From a practical standpoint, a recognition system must not only provide the desired accuracy but speed is also an important factor. Thus, from an algorithm development perspective, it is useful to analyze the performance of various optimization algorithms and the effect they have on the algorithms' performance in terms of the speed and accuracy. For instance, a computational speedup may be obtained by considering variations of the Branch and Bound algorithm [122] rather than the traditional B&B algorithm for point cloud matching. In a similar way, efficient ways for computing a minimum weight perfect matching bipartite graph have also been cited in the literature [108]. A parallel implementation of the proposed graph-based decimation algorithm can lead to a considerable decrease in the computational complexity.

- Learning System: Our recognition system assumes that the model already exists in the database prior to recognition. As part of future work, one needs to design a 'smart' system that can learn the description of an unknown object and in addition have the ability to represent and store such modeling information, not originally present in the database.

Bibliography

[1] M. Werthheimer, “Laws of Organization in Perceptual forms”, *A sourcebook of Gestalt Psychology*, 71-88, 1950

[2] E. Fermi, *Thermodynamics*, Dover Publications Inc., 1956

[3] F. Weinhold, “Metric geometry of equilibrium thermodynamics”, *Journal of chemical physics*. 63, 1975

[4] A.T. Balaban (ed), *Chemical Applications of graph theory*, Academic Press, 1976

[5] W. H. Tsai and K. S. Fu, “Error-correcting isomorphisms of attributed relational graphs for pattern analysis”, *IEEE Transactions on System, Man and Cybernetics*, 9(12), 1979

[6] D. Scott, “On optimal and data-based histograms”, *Biometrika*, 66, 605-610. 1979

[7] L. G. Shapiro and R. M. Haralick, “Structural Descriptions and inexact matching”, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 3,504-519,1981

[8] D. H. Ballard and C. M. Brown, *Computer Vision*, Prentice Hall, 1982

[9] T. H Hong and A. Rosenfeld, ”Compact region extraction using weighted pixel linking in a pyramid”,*IEEE Transactions on Pattern Analysis and Machine Intelligence*, 6(2), 1984

[10] A. K. C. Wong and M. You, “Entropy and distance of random graphs with application to structural pattern recognition”, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 7(5), 1985

[11] P. J. Besl and R. C. Jain, “Three-dimensional object recognition”, *ACM Computing Surveys*, 17(1), 1985

[12] S. Umeyama, “An eigen-decomposition approach to weighted graph matching problems”, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 10(5), 1988

[13] P. J. Besl, *Surfaces in Range Image Understanding*, Springer Verlag, 1988

[14] L. Vincent, “Graphs and mathematical morphology,” *Signal Processing*,16,365-388,1989.

[15] P. Parent and S. Zucker, “Trace Inference, Curvature Consistency and Curve detection”, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 11(8), 1989

[16] T. Fan, G. Medioni and R. Nevatia, “Recognizing 3D Objects using surface descriptions”, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 11(11), 1140-1157, 1989

[17] I. Pitas and A.N. Venetsanopoulos, “Morphological Shape Decomposition,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*,12(1),38-45,1990

[18] W.E.L. Grimson, *Object Recognition by Computer*, The MIT Press, 1990

[19] J.J. Koenderink, *Solid Shape*, The MIT Press,1990

[20] Y. Shinagawa and T.L. Kunii, “Constructing a Reeb Graph from Cross Sections,” *IEEE Computer Graphics and Applications*, 44-51, 1991.

[21] W. Kim and A. C. Kak, “3D object recognition using bipartite matching embedded in discrete relaxation”, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(3), 1991

[22] L.S. Shapiro and J.M. Brady, “Feature based correspondence- an eigenvector approach”, *Image and Vision Computing*, 283-288,1992

[23] W. J. Schroeder, J. A. Zarge and W. E. Lorenson, “Decimation of Triangular meshes”, *Proceedings of ACM SIGGRAPH*, 65-70, 1992

[24] J.J. Koenderink and A.J. Van Doorn, “Surface Shape and curvature scales,” *Image and Vision Computing*,10,557-565,1992

[25] F. Stein and G. Medioni, “Structural Indexing: Efficient 3D Object Recognition”, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(2), 1992

[26] M. Suk and S.M. Bhandarkar, *Three Dimensional Object Recognition from Range Images*, Springer-Verlag,1992.

[27] H. Heijmans, P. Nacken, A. Toet and L. Vincent, “Graph Morphology,” *Journal of Visual Communication and Image Representation*,3(1),24-38,1992.

[28] E.R. Dougherty,(ed.), *Mathematical Morphology in Image Processing*, 1st ed.,1993.

[29] P. Hinker and C. Hansen, “Geometric Optimization”, *IEEE Conference on Visualization*, 189-195, 1993

[30] M. Lindenbaum, “On the amount of data required for reliable recognition”, *International Conference on Pattern Recognition*, 1994

[31] P. Liang and C. H. Taubes, “Orientation-based Differential Geometric Representations for Computer Vision Applications”, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 16(3), 1994

[32] P.Flynn and A.K. Jain, “Three Dimensional Object Recognition”, *In the handbook of Pattern Recognition and Image Processing:Computer Vision*, 497-541, 1994

[33] A. Varshney, *Hierarchical Geometric Approximations*, PhD Thesis, Dept. of Computer Science, University of North Carolina, Chapel Hill, 1994

[34] T.H. Cormen, C.E. Leiserson and R.L. Rivest, *Introduction to Algorithms*, MIT Electrical Engineering and Computer Science, 1994

[35] K. Ikeuchi and M. Hebert, “Spherical Representations: From EGI to SAI”, *Object Representation in Computer Vision, Springer-Verlag*, 327-345, 1995

[36] C.C. Pu and F.Y. Shih, “Threshold Decomposition of Grey-Scale Soft Morphology into Binary Soft Morphology,” *CVGIP-Graphical Models and Image Processing*,57 (6),522-526,1995.

[37] K. Siddiqi and B.B. Kimia, “Parts of Visual Form: Computational Aspects”, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17(3), 239-251, 1995

[38] W. J. Christmas, J. Kittler and M. Petrou, “Structural matching in computer vision using probablistic relaxation”, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17(8),pp. 749-764, 1995

[39] P. Kuosmanen and J. Astola, “Soft Morphological Filtering,” *Journal of Mathematical Imaging and Vision*,5 (3),231-262,1995.

[40] S. Gold and A. Rangarajan, “A graduated assignment algorithm for graph matching”, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 18(4), 1996

[41] D. B. West, *Introduction to Graph Theory*, 1996

[42] R Ronfard and J Rossignac, “Full-range Approximation of triangulated Polyhedra”, *Eurographics: Computer Graphics Forum*, 15(3), 1996

[43] M. Soucy and D. Laurendeau, “Multiresolution surface modeling based on heirarchical triangulation”, *Computer Vision and Image Understanding*, 63(1), 1-14,1996

[44] M. T. Dickerson and D. Eppstein, “Algorithms for proximity problems in higher dimensions” *Computational Geometry: Theory & Applications*,5(5), 277-291, 1996

[45] H. Hoppe, “Progressive Meshes”, *Proceedings of ACM SIGGRAPH*, 99-108, 1996

[46] A. D. Kalvin and R. H. Taylor, “Superfaces: Polygonal mesh simplification with bounded error”, *IEEE Computer Graphics and Appln.*, 16(3), 1996

[47] J.M. Reinhardt and W.E. Higgins, “Efficient Morphological Shape Representation,” *IEEE Transactions on Image Processing*,5(1),89-101,1996.

[48] E.C Sherbrooke, N.M. Patrikalakis and E. Brisson, “An algorithm for medial axis transform of 3D polyhedral solids,” *IEEE Transactions on Visualization and Computer Graphics*,2(1),1996

[49] S. Takahashi, Y.Shinagawa and T.L. Kunii, “A feature-based approach for smooth surfaces,” *Proceedings of Solid Modeling*, 97-110,1997

[50] J. Shi and J. Malik, “Normalized cuts and image segmentation”, *IEEE Conference on Computer Vision and Pattern Recognition*, 1997

[51] J. E. Goodman and J. O’Rourke, *Handbook of Discrete and Computational Geometry*, CRC Press, 1997

[52] K. Wu and M. D. Levine, “3D part segmentation using simulated electrical charge distributions”, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(11), 1997

[53] D.A. McQuarrie and J. D. Simon, *Physical Chemistry: A Molecular Approach*, University Science, 1997

[54] H. Bunke, “On a relation between graph edit distance and maximum common subgraph”, *Pattern Recognition Letters*, 18(8),689-694, 1997

[55] C. Dorai and A.K. Jain, “COSMOS: A representation scheme for 3D free-form objects,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(10):1115-1130, 1997

[56] P. Viola and W.M. Wells, “Alignment by maximization of mutual information”, *International Journal of Computer Vision*, 24(2), 137-154, 1997

[57] S. A. Nene and S. K. Nayar, “A simple algorithm for nearest neighbor search in higher dimensions”, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(9), 999-1013, 1997

[58] W. J. Schroeder, “A topology modifying progressive decimation algorithm”, *IEEE Conference on Visualization*, 205-212, 1997

[59] M. Garland and P. S. Heckbert,“Surface simplification using quadric error metrics”, *Proceedings of ACM SIGGRAPH*, 209-216, 1997

[60] P. S. Heckbert and M. Garland, “Survey of polygonal surface simplification algorithms”, *Proceedings of ACM-SIGGRAPH*, 1997

[61] G. Barequet and S. Kumar, “Repairing CAD Models”, *Proceedings of IEEE Conference on Visualization*, 1997

[62] F. P. Preparata and H. Hoppe, “Progressive Simplicial Complexes”, *Proceedings of ACM SIGGRAPH*, 217-224, 1997

[63] B. Mohar, “Some applications of Laplace eigenvalues of graphs”, *Graph Symmetry: Algebraic Methods and Applications, Kluwer*, 497, 227-275, 1997

[64] C. H. Papadimitriou and K. Steiglitz, *Combinatorial Optimization: Algorithms and Complexity*, Dover Publications, 1998

[65] P. Lindstrom and G. Turk, “Fast and memory efficient polygonal Simplification”, *IEEE Conf. Visualization*, 279-286,1998

[66] H. Bunke and K. Shearer, “A graph distance metric based on maximal common subgraph”, *Pattern Reconginition Letters*, 19(3), 255-259, 1998

[67] P. Atkins, *Physical Chemistry*, Oxford University Press, 1998

[68] A. E. Johnson and M. Hebert, “Control of Polygonal Mesh resolution for 3D computer vision”, *Graphics Modeling and Computer Vision*, 1998

[69] A. E. Johnson and M. Hebert, “Using Spin Images for efficient object recognition in cluttered 3D scenes”, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21(5), 433-449, 1999

[70] K. Siddiqi, A. Shokoufandeh, S. Dickinson and S. Zucker, “Shock Graphs and Shape matching”, *International Journal of Computer Vision*, 30, 1-24, 1999

[71] M. Pellili, K. Siddiqi and S. Zucker, “Matching Hierarchical Structures using association graphs”, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21(11), 1105-1120,1999

[72] K. Siddiqi, A. Shokoufandeh, S. Dickinson and S. Zucker, “Shock Graphs and Shape Matching”, *International Journal of Computer Vision*, 35(1), 1999

[73] T. Liu and D. Geiger, “Approximate Tree Matching and Shape Similarity”, *Proceedings of International Conference on Computer Vision*,456-462,1999

[74] H. Bunke, “Error Correcting Graph Matching: On the influence of the underlying cost function”, *IEEE Transactions on Pattern Analysis and Machine Intelligence* 21(9), 1999

[75] A.P. Mangan and R.T. Whitaker, “Partitioning 3D surface meshes using watershed segmentation,” *IEEE Transactions on Visualization and Computer Graphics*, 5(4):308-321, 1999.

[76] , M. Garland, “ Multiresolution Modeling: Survey and Future Opportunities”, *Eurographics*, 113-131, 1999

[77] I. Guskov and W. Sweldens and P. Schröder, “Multiresolution signal processing of meshes”, *Proceedings of ACM SIGGRAPH*,325-334, 1999

[78] N. Dyn, K. Hormann, S.J. Kim and D. Levin, “Optimizing 3D triangulations using discrete curvature analysis,” *Mathematical Methods for Curves and Surfaces*,135-146,2000.

[79] P. Rosin, “Shape Partiotioning by Convexity”, *IEEE Trans. Systems, Man, and Cybernetics, part A*, 30(2),202-210, 2000

[80] K. L. Boyer and S. Sarkar (ed.), “Perceptual Organization for Artificial Vision Systems”, *Kluwer Academic*, 2000

[81] H. Bunke X. Jiang and A. Kandel, “On the minimum common supergraph of two graphs”, *Springer Verlag*, 65(1),13-25, 2000

[82] H. Chui and A. Rangarajan, “A new algorithm for non-rigid point matching”, *IEEE Conference on Computer Vision and Pattern Recognition*, 2, 44-51, 2000

[83] C. Rössl, L. Kobbelt and H.P. Seidel, “Extraction of feature lines on triangulated surfaces using morphological operators,” *Proceedings of AAAI Symposium on Smart Graphics*, 71-75, 2000.

[84] M. Carcassoni and E.R. Hancock, “Point Pattern Matching with Robust Spectral Correspondence”, *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, 2000

[85] R. Myers, R. Wilson and E. Hancock, “Bayesian Graph Edit Distance”, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(6), 628-635, 2000

[86] M. Boshra, B. Bhanu, “Predicting Performance of Object Recognition”, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(9), 956-969, 2000

[87] C.D. Ruberto and A.G. Dempster, “Attributed Skeleton Graphs using mathematical morphology,” *IEEE Electronics Letters*,37(22),2001.

[88] B. Ma, A. O. Hero, J. Gorman, O. Michel, “Image Registration with minimum spanning tree algorithm”, *Proceedings of International Conference on Image Processing*, 2001

[89] B. Luo and E. R. Hancock, “Structural Matching using EM algorithm and Singular Value Decomposition”, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23, 1120-1136, 2001

[90] A. R. Kelly and E. R. Hancock, “Graph Matching using adjacency matrix markov chains”, *Proc. British Machine Vision Conference(BMVC)*, 2001

[91] H. Cantzler and R.B. Fisher, “Comparison of HK and SC curvature description methods,” *Proceedings of IEEE International Conference on 3D Digital Imaging and Modeling*,285-291,2001.

[92] C. Godsil and G. Royle, *Algebraic Graph Theory, Springer-Verlag*, 2001

[93] R. O. Duda, P. E. Hart and D. G. Stork, *Pattern Classification*, Wiley-InterScience, 2001

[94] Z.Karni and C. Gotsman, “3D Mesh Compression Using Fixed Spectral Bases”, *Proceedings of Graphics Interface*, 2001

[95] L. Shams, S. Schaal, “Graph-matching vs. entropy-based methods for object detection”, *Neural Networks*, 14(3),345-354, 2001.

[96] C. M. Cyr and B. Kimia, “3D object recognition using shape similarity based aspect graph”, *Proceedings of International Conference on Computer Vision*, 254-261, 2001

[97] R.J. Campbell and P.J. Flynn, “A survey of free form object representation and recognition techniques,” *Computer Vision and Image Understanding*,81(2), 2001.

[98] A. Guéziec, G. Taubin, F. Lazarus and B. Horn, “Cutting and Stitching: Converting sets of Polygons to Manifold Surfaces,” *IEEE Transactions on Visualization and Computer Graphics*,7(2),2001.

[99] G. Hetzel, B. Leibe, P. Levi, B. Schiele, “3D object recognition from range images using local feature histograms,” *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*,2,394-399,2001.

[100] R.C.Gonzalez and R.E. Woods, *Digital Image Processing*,2nd ed., Prentice Hall,2001.

[101] R.J. Campbell and P.J. Flynn, “Recognition of free-form objects in dense range data using local features,” *Proceedings of 16th International Conference on Pattern Recognition*,3, 607-610, 2002.

[102] R. J. Campbell and P.J. Flynn, “Recognition of free-form objects in dense range data using local features”, *Proceedings of International Conference on Pattern Recognition*,3,607-610, 2002.

[103] S. Kosinov and T. Caelli, “ Inexact Multisubgraph matching using graph eigenspace and clustering models”, *Proceedings of SSPR/SPR, Springer-Verlag*, 2396, 133-142, 2002

[104] A.R. Ahmadyfard and J. Kittler, “Using Relaxation techniques for region-based object recognition”, *Image and Vision Computing*, 20(11), 769-781, 2002

[105] E. R. Hancock and R. C. Wilson, “Graph based methods for vision: A yorkist manifesto”, SSPR & SPR, LNCS 2396, 2002

[106] J. Li and A.O. Hero, “A spectral approach to statistical polar shape modeling”, *Proceedings of International Conference on Image Processing*, 2002

[107] D. Macrini, A. Shokoufandeh, S. Dickinson, K. Siddiqi, S. Zucker, “View-based 3-D object recognition using shock graphs” *International Conference on Pattern Recognition*,3, 24-28, 2002

[108] S. Belongie, J. Malik and J. Puzicha, “Shape Matching and object recognition using shape contexts”, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(24), 509-522, 2002

[109] M. Mortara and G. Patanè, “Affine invariant skeleton of 3D shapes,” *Proceedings of Shape Modeling International*,1-8, 2002.

[110] D. Bespalov, A. Shokoufandeh, W.C. Regli and W. Sun, “Scale-Space representation of 3D Models and topological matching,” *Proceedings of Solid Modeling*, 208-215, 2003

[111] D.L. Page, A.F. Koschan and M.A. Abidi, “Perception-based 3D triangle mesh segmentation using fast marching watersheds,” *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*,2,27-32, 2003.

[112] R. Hauge, “Ladar puts the puzzle together”, *SPIE OE Magzine*, 18-20 April 2003

[113] C. Gotsman, “On Graph Partitioning, Spectral Analysis, and Digital Mesh Processing”, *Proceedings of Solid Modeling International*, 2003.

[114] M. Kazhdan, T. Funkhouser and S. Rusinkiewicz, “Rotation invariant spherical harmonic representation of 3D shape descriptors”, *Proceedings of Eurographics symposium on Geometry processing*, 2003

[115] E. R. Hancock, M. Vento (eds.), *Graph based Representations in Pattern Recognition*, Springer Verlag, 2003

[116] Y. Keselman, A. Shokoufandeh, M. F. Demirci and S. Dickinson, “Many-to-Many Graph Matching via Metric Embedding”, *IEEE Conference on Computer Vision and Pattern Recognition*, 2003

[117] T.S. Caetano, T. Caelli and D.A.C. Barone, “Graphical Models for Graph Matching”, *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, 2004

[118] T. Caelli and S. Kosinov, “An eigenspace projection clustering method for inexact graph matching”, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(4), 515-519, 2004

[119] T. Sebastian, P. Klein and B. Kimia, “Recognition of Shapes by editing their Shock Graphs”, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(5), 550-571, 2004

[120] H. Neemuchwala, A.O. Hero and P. Carson, “Image matching using alpha-entropy measures and entropic graphs”, *European Journal of Signal Processing* (Special issue on content-based visual information retrieval), March 2004

[121] D. Huber, A. Kapuria, R. Donamukkala, M. Hebert, “Parts-based 3D object classification”, *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 82-89, 2004

[122] P. Somol, P. Pudil, J. V. Kittler, “Fast Branch & Bound Algorithms for Optimal Feature Selection”, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(7), 900-912, 2004

[123] A. Jagannathan and E. L. Miller, “Unstructured 3D Point Cloud Matching within Graph-theoretic and Thermodynamic Frameworks”, *Proceedings of the IEEE Computer Society’s Conference on Computer Vision and Pattern Recognition (CVPR)*, 2005

[124] A. Jagannathan and E. L. Miller, “Shape-preserving Mesh Decimation within a Graph-theoretic Framework”, *Proceedings of the IEEE Signal Processing Society’s Asilomar Conference on Signals, Systems and Computers*, 2005, to appear

[125] A. Jagannathan and E. L. Miller, “Mesh Segmentation: From Triangulations to Parts”, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, accepted, under revision

[126] http://www.paraform.com/ppdl