AN ALGORITHMIC PERSPECTIVE ON PROBLEMS IN BIOLOGICAL IMAGING

BY
LOPAMUDRA MUKHERJEE

DISSERTATION

Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy
in Department of Computer Science and Engineering
University at Buffalo, The State University of New York, 2008

Buffalo, New York

AN ALGORITHMIC PERSPECTIVE ON PROBLEMS IN BIOLOGICAL IMAGING

Lopamudra Mukherjee
Department of Computer Science and Engineering
University at Buffalo, The State University of New York, 2008
Jinhui Xu, Advisor

The last decade has seen considerable advances in our understanding of the genome and computational methods have played a very crucial role in these developments. Decoding the human genome posed a significant computational challenge and is undoubtedly one of our major accomplishments; today, future research direction is focused on *understanding* the sequences at an unprecedented level of detail. This process is continuously generating a steady stream of interesting computational problems. In this dissertation, we concentrate on several such problems arising from the need to understand the structural and functional organization of the nuclear components and their effect on gene regulation and function.

The first problem we study is related to extracting the same or similar "foreground" from a set of image slices. The set of $2D$ slices correspond to images of the $3D$ chromosome acquired at different focal planes of the fluorescent microscope. But since the resolution in-plane (in the $2D$ slice) is higher than the resolution along the stack, a standard three dimensional segmentation is rather problematic. Our view of this problem is to perform figure-ground segmentation in the $2D$ planes individually with the additional condition that the foregrounds must be consistent w.r.t. each other. We design new based algorithms for this "cosegmentation" problem formulating it as a Markov Random Field (MRF) type energy function and report on encouraging results on a variety of images.

The second problem we tackle is the so-called Generalized Median Graph problem motivated from the requirement of building topological maps of chromosome organization in the nucleus. The problem belongs to a class of prototype building problems where the input class is a set of graphs; in other words, we want to build a 'model graph' for a set of graphs. Our main result is the first combinatorial algorithm for this problem which runs in polynomial time and yields near-optimal solutions in practice. We show that using our LP based algorithm, even in the worst case, a solution within a factor of two of the optimal solution can be obtained if the distances between the graphs are known correctly. We propose an additional algorithm based on a bi-level framework to obtain solutions arbitrarily close to the optimal but in non-polynomial time. We discuss experimental evaluations using a publicly available graph database and illustrate results on a set of biological images.

The third problem we address is an important partial matching problem of geometric objects. The input is in the form of sets of under-sampled slices (e.g., $2D$ contours) of one (or more) unknown $3D$ objects (e.g., $3D$ chromatin surfaces), possibly generated by slicing planes of arbitrary orientations, the question we are interested in is whether it is 'possible' that two under-sampled sets have been taken from the same object. Since the three dimensional structure of objects of interest (e.g., chromosomes) often cannot be inferred precisely in fluoroscence microscopy because of poor resolution in the third dimension, such techniques are important to establish correspondence between structures in a sequence of time-lapse images. We present efficient algorithms for addressing this question.

Next, we propose new approaches for analyzing time-lapse microscopic nuclear images. Our techniques address the problem of limited spatial and temporal resolution capacities of current microscopic imaging techniques which allow acquisition of images only in time-lase mode (which leads to significant frame-to-frame information loss). We present a suite of geometric approaches for solving the problem. Our techniques provide a comprehensive solution to (1) raw image simplification (2) segmentation them and (3) effective recovery of complicated motion and deformation as well as the change of intensity surfaces from pairs of images in a microscopic image sequence. These techniques are also readily applicable to other types of images for reconstructing motion and intensity surfaces of deformable objects.

Finally, we propose a set of novel techniques to determine, analyze, and interpret the mobility patterns of functional sites (Replication and Transcription) in the nucleus. This is motivated from recent studies that have shown a link between movement of the sites and the actively expressing components of DNA. Our algorithms provide for the first time the tools to interpret the seemingly stochastic motion patterns of the functional sites within the nucleus in terms of a set of tractable 'patterns' which can then be analyzed to understand their biological significance.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

MICROSCOPY is a powerful imaging tool in modern biological research. Significant rogress in optical technology has led to the design and development of better and more powerful image acquisition devices. Simultaneous strides have also been made in the development of techniques to analyze and understand the images better. Hand labeling and manual calculations have largely been replaced by sophisticated computational tools which address tasks including (but not limited to) simplification of images, measurements (distances and other quantitative features) and the identification of cells. Infact, computational methods have become intrinsic not only to imaging but all aspects of modern biological research. As the problems of relevance to the community become more complex and computationally demanding, advanced data analysis methods are needed to find efficient solutions. The imaging devices in routine use in current biological research generate a large volume of image data: this requires the development of powerful data analysis tools that can process such data and transform the *qualitative* images to meaningful *quantitative* information.

In this dissertation, we propose a set of algorithmic tools to facilitate the study of three important biological imaging problems: (a) Chromosome organization; (b) Dynamics of chromatin domains in the nucleus of living cells; and (c) Mobility properties of functional sites (such as replication and transcription sites). These are crucial steps in understanding the structural and functional organization of nuclear components and their effect on gene regulation and function. We discuss the biological significance of these problems in Section 1.2. In Section 1.3, we focus on formulating the biological questions mathematically, at the right level of abstraction. But before introducing the problems of interest, we review the important 'phases' of a typical biological image analysis system and see how each problem fits into a broader research goal.

1.1 Biological Image Processing/Analysis Systems

The biological image analysis pipeline typically starts from the acquisition of the images. Before we may obtain the final "results", a number of problems such as preprocessing, registration and segmentation must be addressed. We provide a brief description of some of these important steps below.

Image Acquisition. The main tool for image acquisition is microscopy. Microscopy can be divided into several broad categories such as optical, electron, scanning probe and infrared microscopy. Optical or light microscopy are especially popular and the device of choice for the problems and evaluations discussed in this dissertation. A detailed review of optical microscopic techniques can be found in http://micro.magnet.fsu.edu/primer/opticalmicroscopy.html. Among other light microscopic techniques, fluorescence microscopy is preferred because of its high specificity in the visualization of particles that fluoresce in the presence of excitatory light. The acquisition proceeds by tagging different molecules with fluorescent dyes. Then, a fluorescent microscope illuminates the molecules with high energy light. In response, the molecule emits a lower frequency light. The microscope separates the emitted light from the brighter excitation which then is captured by a CCD camera (full color or monochrome). Besides fluorescence microscopy, the other common forms of optical microscopy are Confocal and Phase Contrast Microscopy.

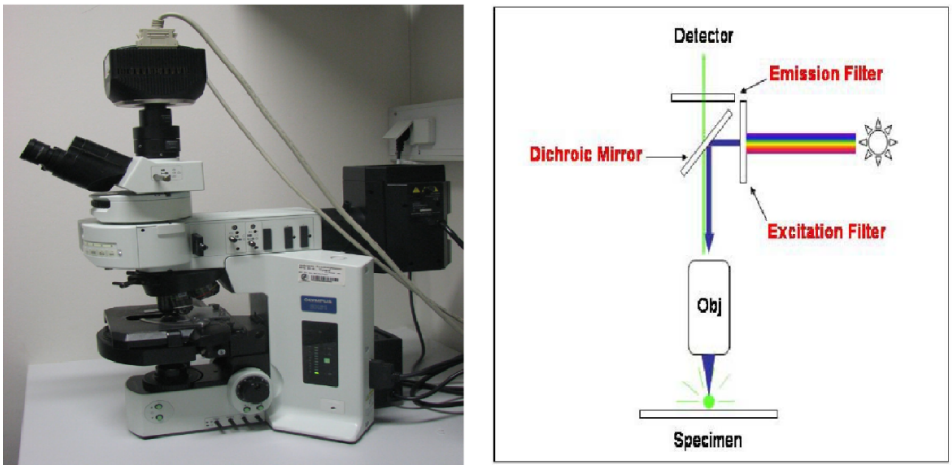


Figure 1.1: Fluorescent microscope (left) and schematic diagram of its filters (right); Source: Wikipedia

Preprocessing. Preprocessing immediately follows image acquisition and is a necessary prerequisite for any subsequent image analysis. This included procedures

such as noise and artifacts removal (or suppresion), signal-to-noise ratio enhancement and adjustment of brightness and/or contrast primarily for visualization purposes. Convolution/filtering operations are commonly used for noise reduction and denoising as well as to correct for illumination non-uniformities.

Visualization(3D) Images of the three dimensional microscopic structures in cells are typically acquired as a set (or stack) of $2D$ slices. In *live cell* imaging, the phenomena of interest is intrinsically temporal $3D$, but only $2D$ slices at discrete time points are available. The visualization process therefore requires extrapolation using tools from computer graphics such as surface or volume rendering. Volume rendering visualizes complex 3D structures as 2D projections without explicity defining the surface geometry. In contrast, surface rendering techniques computes the object surface, representing it as a mesh. The most common surface rendering technique is the Marching Cube algorithm [60], which creates a triangulation of the 3D surface. The 3D surface is visualized as an iso-surface, after applying thresholding to all images in the stack, using a same or different intensity thresholds.

Segmentation Segmentation refers to the tools and techniques used to partition an image into meaningful regions. In context of biological imaging, segmentation refers to the task of extracting the object(s) of interest from the background (i.e figure-ground segmentation). The simplest form of segmentation is threshold-based where the pixels are divided into foreground and background based on a certain intensity threshold. However, given the variability of appearance of functionally identical objects from one cell to the next and the non-uniform illumination properties within a cell, it is often difficult to find a threshold that works for similar images and yields the object of interest as a single meaningful region. If a priori knowledge about the shape of the objects are available, an active contour model approach using a shape prior can be used for better segmentation such as in [27]. Graph cuts based techniques which enforce smoothness properties of the resultant segmentation has been adopted [20; 107]. Despite segmentation of biological images being a widely studied problem, several challenges still remain. Given the large volume of images being produced by current imaging devices, the biggest challenge is to adapt or devise a segmentation technique to segment several related images simultaneously. We address one such problem in Chapter 2.

Registration The goal of image registration is to establish correspondence between images of the same object acquired from different views, at different times or in different coordinate systems. The need for registration occurs in several biological imaging applications such as finding correspondence of chromatin domains from time-

lapse microscopic images and registering images of the same object acquired under different imaging modalities (such as DAPI stained flourescence microscopic images and phase contrast images of the nucleus). The choice of the registration algorithm depends on the problem at hand. For example, in the case of registering images of nucleus (as mentioned above), the movements are mainly due to shift in the cover slip from one image modality to the next. Hence, an affine registration algorithm like ICP (Iterative Closest Point) is sufficient to register the images accurately. However, occasionally objects are subject to shape changes during the acquisition process: for example, chromatin domains (CD) change their shape and size as they move from one timepoint to next. In such cases, a deformable registration algorithm based on TPS (thin plate splines) can be applied. Notice that such registration tasks are often complicated due to the presence of large number of CDs or sites of genetic activity that need to be registered, but also several other competing influences such as movement of the cell as a whole and disappearence/re-emergence of sites from one time point to the next. We discuss these issues in detail in Chapter 5.

Tracking Tracking refers to the task of finding correspondence of an object across consecutive frames of an image sequence, and is common in several live microscopy imaging problems. The correspondence is usually preceded by registration to correct for the global movement of the cell and local transformations of individual objects (if applicable). Tracking algorithms vary based on the criterion used for determining correspondence and also the matching algorithms used. Several parameters such as local image features, nearest neigbor information, inter-object relationships and optical flow [49] have been widely used. Also, fuzzy-logic based tracking approaches [97] have been shown to yield reliable results. In Chapter 5, we discuss a new registration/tracking algorithm for analysis of movement of chromatin domains from time-lapse micrscopic image sequences.

Parameter Estimation Subsequent to the steps descibed above, parameters of biological interest such as velocity, acceleration, parameters of rotation and translation are often estimated in live cell imaging. Statistical tools are also used for evaluating the mean, standard deviation and corelation coefficients to evaluate the similarity of spatial distributions of proteins to understand their functional significance.

Novelty discovery Systematic study of complex interactions in biological systems often requires the design of sophisticated analytical tools drawn from a wide variety of areas like pattern recognition, machine learning, data-mining and vision. Some of the common applications include clustering gene expression patterns, tissue type classification, mining biomedical databases, automatic identification of cell

phases, to name a few. In this dissertation, we study a few such problems in Chapters 3 and 6.

1.2 Background and Motivation

The genetic information of all organisms is encoded in the cell nucleus. The cell nucleus is the core of a cell which separates the linearly coded genetic information (or chromosomal karyotype) from the surrounding cytoplasm. Inside the nucleus, DNA is highly packaged in a hierarchy including DNA double helix, nucleosomes, chromatin fiber, chromatin loop domains, higher order chromatin loop domains, and chromosome. The residual framework (after extracting the chromatin from the nucleus) of the cell nucleus is termed the nuclear matrix. The structural changes of the nucleus matrix are widely believed to have close correlation with genomic functions, such as DNA replications, gene transcription, and RNA splicing/transportation. While significant progress has been made in the past decade in understanding the molecular basis of the human genome, not much information is available about the structural and functional organization of the nuclear components and how they affect gene regulation and function. We have still not able to fully understand how the whole genome is coordinated. Gene library and sequencing analysis methods have shown that humans are not endowed with significantly more genes than other mammals, and have only about twice as many genes as the fruitfly. Thus, it is the coordination of overall genomic expression and not the sheer number of individual genes that makes the phenotypic difference among species.

To fully understand the coordination of genomic expression, we have to explore much more than *just* the simple linear sequence of the DNA. We need to understand how the linear array of genes is arranged along the chromosomes and how it is expressed in the three-dimensional context of the cell nucleus [8]. With the recent development of $3D$ microscopy and the availability of a number of gene probes and labeling approaches, it is now possible to visualize sites of genomic components. But this is still far away from revealing the mechanism of nucleus for orchestrating the activation and silencing of genes. Part of the reason is that there is enormous amount of DNA inside a nucleus and the complicated (unknown) movement of DNA makes it extremely difficult to determine how genes are turned on and off and what is their dynamics. Current microscopic imaging and labeling techniques can only provide very limited information for motion tracking. Nuclear images are often noisy where small

errors in acquisition propagate strongly into the data, and unlike most organ-scale imaging scenarios, few ground truth models (and expert inputs) are available. Thus, new computational techniques are need that can be coupled with the biological approaches to answer these questions accurately and effectively. The goal of this thesis is to address some of these problems motivated from our need to better understand the cell nucleus. To provide automatic tools for information integration and analysis, we present here a collection of novel mathematical models and algorithmic tools for studying three biological problems: (1) chromosome organization, (2) the dynamics of chromatin domains, and (3) the mobility properties of functional sites (such as replication/transcription sites (RS/TS)) in the living cell nucleus. In the next section, we briefly introduce the biological basis and computational aspects of each problem.

1.3 Overview of our Results

The first problem we study is how to segment two or more related images simultaneously, called the Cosegmentation problem and is applicable to the problem of segmenating chromosomes from 3D image stacks. The second problem we tackle is the Generalized Median Graph problem. This problem arises naturally out the need for building topological maps of chromosome organization in the nucleus. The next problem we study is the problem of matching 3D geometric objects which are represented by under sampled slices. This has applications in several biological and medical imaging problems which will be discussed in-depth later. The fourth problem relates to determining the dynamics of objects from time-lapse image sequences, with special focus on microscopic nuclear image sequences of living cells. Last but not the least, we present a suite of novel techniques to determine, analyze, and interpret the mobility patterns of functional sites (RS/TS) in nucleus from time-lapse image sequences.

1.3.1 Cosegmentation of Images using histogram matching

Images of chromosomes in the nucleus are acquired by labeling the chromosomes with nucleic acid sequences known to hybridize specifically with that particular chromosome, and then imaging it using a flourescent microscope. This process results in a series (or slices) of images of the 3D chromosome derived from different focal planes (called the Z-stack). However, inspite of using deconvolution techniques, fluorescence microscopy suffer from significant blurring artifacts (caused due to emission of light from structures which are out of focus), and this makes it difficult to understand the

structural properties of the chromosome clearly. To mitigate this problem, additional image processing techniques are used to further de-noise and segment the contours for further processing. One such segmentation approach is the threshold based approach, which chooses a single intensity threshold to apply to all images of the stack individually. Though threshold-based segmentation works nicely for individual images in the stack, the segmentation of chromosome contours when viewed across consecutive images are often somewhat discontinuous. A continuous and smooth segmentation across the Z-stack can be used to generate more accurate chromosome contours and hence a realistic estimate of its geometry and spatial proximity to other chromosomes which is a focus of the next section.

The problem of segmenting a 3D stack of images can be precisely formulated as a cosegmentation problem on consecutive images of the stack. Cosegmentaion refers to the task of segmenting the same object from a pair of images. The segmentation for each image can be cast using a partitioning/segmentation function that partitions the images into foreground and background such that the histograms of the segmented foreground regions (based on intensity, texture features etc.) is similar. Apart from being a reasonable requirement to achieve the desired objective, this constraint allows a representation in a simple algebraic form. Using Markov Random Field (MRF) energy terms for the simultaneous segmentation of the images together with histogram consistency requirements using the squared L_2 (rather than L_1) distance, after linearization and adjustments, yields an optimization model with interesting combinatorial properties. We investigate and discuss these properties and provide a linear programming based algorithm for the problem in Chapter 2.

1.3.2 Generalized Median Graphs

Large-scale geometry of chromosomes in cell nuclei influences many molecular biology processes. Of particular interest is its effect on genomic structure and its functional implications during development and cell differentiation. Variability in genomic structure in turn accounts for cell-type-specific gene expression and silencing patterns in multicellular organisms. However the direct impact of higher-order nuclear architecture on these patterns is not yet known. A promising direction in understanding genomic evolution and cell-type-specific variability points towards the studies of higher-order chromatin arrangements in numerous cell types from different species. Numerous research projects have attempted to map the large-scale organization and distribution of chromatin in various cell lines. A reliable 3D topological model would provide the necessary foundation for studying the effect of higher-order chromatin

distribution on nuclear functions. Unfortunately, a comprehensive three-dimensional arrangement of all 23 pairs of chromosome in a diploid somatic cell nucleus have been lacking so far. There is a need for such models for different cell types at various stages of the cell cycle and at various stages of terminal differentiation. In this dissertation, we develop techniques to build models of chromosome organization which are general enough to be applicable to all of the aforementioned scenarios.

If all information regarding chromosome organization in a particular cell is represented as a graph, the problem of building a topological map of chromosome organization naturally transforms to an instance of a generalized median graph problem which is as follows.

Problem 1 (Generalized Median Graphs) *Let L_V and L_E denote the set of node and edge labels of the graph, respectively. A labeled undirected graph G is then a four tuple, $G = (V, E, f_v, f_e)$, where V is the vertex set, E is the edge set, $f_v : V \rightarrow L_V$ is a function assigning labels or weights to the nodes, and $f_e : E \rightarrow L_E$ is a function assigning labels or weights to the edges. Let $G = (G_1, G_2, \ldots, G_n)$ be a collection of graphs with the following properties:*

- $\forall G_i = (V_i, E_i, f_v, f_e)$, $V_i \subset V$ *and* $E_i \subset E$.
- *No restriction on the uniqueness of the vertex labels of V_i, i.e., $f_v(u_i) = f_v(v_i); u_i, v_i \subset V_i$ is allowed.*
- *No restriction on the cardinality of the graphs in G, i.e., $|G_i| \neq |G_j|; G_i, G_j \in G$ is allowed.*

The generalized median graph for $G = \{G_1, G_2, \ldots, G_n\}$ is defined as

$$\bar{G} = \arg\min_{\hat{G}} \sum_{i=1}^{n} d(\hat{G}, G_i),$$

where $d(\cdot, \cdot)$ is an appropriately defined 'distance' function.

The median graph problem shows strong connections to several well known notorious NP-hard problems, which partially explains why it is difficult to design a polynomial time algorithm for it. For instance, a closely related problem is the subgraph isomorphism problem (also called graph matching problem; an APX-hard problem). For this problem, even though no polynomial time algorithm is previously known, there were several recent attempts on developing practically efficient algorithms to find suboptimal solutions for some special cases.

In Chapter 3, we propose a polynomial time algorithm for the median graph problem. Our proposed approach first introduces an edit grid which is the adjacency matrix of the to-be-determined median graph, and then tries to embed the n input graphs onto the edit grid by permuting them to the common graph (i.e., the unknown median). With the introduced edge weights, we can represent the total "distance" between the median graph and the n input graphs as a quadratic function by using a generalized edit distance function. Using an interesting linearization technique, we are also able to show that the median graph can be formulated as an integer program (IP), and can be solved in polynomial time through relaxation to obtain good sub-optimal solutions. We show that if the pairwise graph distance can be accurately computed, our approach also implies a 2-approximation for the median graph problem. The quality of our solution can also be further improved by using a bi-level framework. This bi-level algorithm first solve a linear program and then use the solution as input to a feasibility problem. By solving a sequence of the feasibility problems, we are able to obtain a solution for the median graph problem which is arbitrarily close to the optimal. Our algorithm is **the first algorithm** for this problem which runs in polynomial time and yields near optimal solutions. Furthermore, our algorithm can also be used as a graph matching technique. Experimental results are shown on a publicly available graph database and the application of median graphs to the problem of building topological maps of chromosome organization in the nucleus is discussed at length.

1.3.3 Fitting Polygonal Regions for Matching 3D Polyhedra

To determine the dynamics of chromatin domains (CD) in a living cell, a key problem is to determine, for each CD, its corresponding ones in next image. However, since the cell nucleus could have global movement, and each CD is only represented by a set of under-sampled $2D$ cross-sections (or contours) in microscopic images, the two sets of contours for the same CD (in two consecutive images) could be rather different even if there is no deformation. Thus to establish correspondence for a set (up to a few thousands) of CDs in nuclear images, a key problem is to develop efficient techniques for the following geometric *fitting problem*.

Given two sets P and Q of $2D$ polygonal regions with each set representing the cross sections (or slices) of an unknown $3D$ polyhedron, design an efficient algorithm to determine whether P and Q are generated from the same $3D$ polyhedron. In each of the two sets P and Q, a cross section c_i may include one or more simple polygons with possible holes in the interior of each polygon. The boundaries of these

polygons are the common intersections of the cross sections and the boundary of the $3D$ polyhedra. The set of cross-sections in one set are parallel to each other and evenly spaced in the region occupied by the union of the possibly different $3D$ objects.

Matching geometric objects is a fundamental problem in computational geometry with numerous applications in many different areas [3]. Due to its importance, this problem has been extensively studied [3]. A common approach for solving this problem is to compute the minimum "distance" between a pair of objects under certain transformations (e.g., translation, rotation, scaling, etc.), where the distance function could be either Hausdorff or Fréchet distance. In computer vision, medical imaging, and computational molecular biology, a related problem, called *alignment problem*, has been studied [3]. The alignment problem tries to recognize (or dock) a known object (or model) from an image scene by using model-based recognition techniques [3]. In this problem, the model is often known and some reference points as well as their corresponding points in the other image or object can be relatively easily determined. However, in our fitting problem, both polyhedra are unknown and the reference points is often difficult to obtain. Thus more efficient technique is needed.

Matching in $3D$ (using Hausdorff distance) often exhibits high complexity even under translation. For example, current best known algorithm for matching two set of points with size m and n takes $O((mn)^2((n+m)^{1+\epsilon}))$ time. Part of the reason is because the number events for matching is $O((nm)^3)$. For more complicated pattern (e.g., segments, curves) matching, the complexity is even higher.

Based on several interesting techniques, we have obtained an efficient solution [68] for this problem under translation. Our approach includes two phases. In the first phase, our algorithm identifies a set of possible locations whose size in general is much smaller than the number $O((mn)^3)$ of events for points or segments matching. In the second phase, our approach verifies each location found in the first phase by using an simple $1D$ or $2D$ arrangement. The key to our approach is an interesting reduction from the shape matching problem to a dynamic string matching problem. In the string matching problem, we are given a dynamically changing (i.e., supporting insertion and deletion) text and a set of fixed patterns (defined over positive integer between 1 and n). The goal is to design a data structure so that all matches (between the text and the patterns) can be efficiently reported when there is a change in the text. Using a data structure given in [87], we are able to report each match in polylogarithmic time.

1.3.4 Determining Dynamics of Chromatin Domains in Living Cells

Chromatin domains (CD) in a nucleus are subject to constant changes in shape, position and orientation. The determination of such dynamics is crucial for understanding nuclear organization and function, especially the overall genomic coordination of the cell nucleus. Current research on determining the dynamics of the nuclear architecture (e.g., spatial and temporal of replication and transcription sites, chromatin domains, and chromosome territories) is still in its early stage. The reasons for this are manifold. First of all, current techniques for imaging living cells (mainly optical microscope) can not yet provide the desired spatial resolution for exactly motion tracking. For example, a typical optical microscope generate a set of parallel 2-D cross sections at $0.5\mu m$ intervals. But DNA double helix is only about $2nm$ in diameter, nucleosomes 11 nm, and chromatin fiber 30 nm. Thus many details of the chromatin will not be visualized. Secondly, chromatin in nucleus often exhibits rather complicated and rapid motion. Current microscopy techniques can not yet provide the necessary time resolution to keep tracking of the shape changes of chromatin domains. Thirdly, available gene probing (such as fluorescence in situ hybridization or FISH) and labeling techniques (e.g., using antibodies to proteins) often introduce diffused signal and make it very difficult to segment the exact contours of the labeled domains.

Constrained by these difficulties, most of previous studies on the dynamics of chromatin domains use either statistic methods or manual calculation to measure the trends or overall movement of functional domains in cell nucleus[10; 11], and cannot yield accurate results. To provide more efficient and reliable automatic techniques, we develop a set of algorithms for solving the following three problems: (1) Determine global movement of a cell nucleus; (2) Find corresponding chromatin domains in two consecutive images; (3) Recover local movement and deformation of each individual chromatin domain.

1.3.5 Understanding the Mobility Properties of Functional Sites

Replication and Transcription sites (RS/TS) exhibit constant motion inside the nucleus. Recent research [8] observes a significantly faster movement of the sites encompassing the euchromatin DNA than those around the hetrochromatin DNA. Since euchromatin is known to be mainly composed of actively expressing genes whereas hete-

rochromatin consists largely of inactive genes, the observed motion behavior strongly suggests a deeper connection between the functional sites' mobility patterns and gene expression. To better understand this relationship, we propose to develop algorithmic tools for determining and understanding mobility patterns of functional sites.

To study the mobility properties, we use cells transfected with a PCNA-GFP fusion construct to generate images of functional sites. It has been demonstrated that PCNA associates at sites of replication throughout S-phase [86]. In addition, PCNA is known to dissociate from sites where replication has concluded and dynamically reassociate at sites of ongoing replication. This confers us the ability to study and track the functional sites such as DNA replication in living cells. The recombinant DNA construct fuses the genes PCNA and GFP produces a modified fluorescent PCNA-GFP fusion protein. The PCNA-GFP plasmid is transfected into human HeLa cells. Colonies expressing this fusion protein stably are selected and exponentially growing populations of such HeLa cells are observed with a temperature controlled chamber mounted on our Olympus IX-70 flourescence microscopy. This allows us to track a functional site in three dimensions and time. We optimize a time-lapse collection routine of 30 frames per minute for tracking the dynamics.

We are particularly interested in understanding the mobility properties of replication sites. We approach this goal with a two-fold objective. The first goal is to determine whether there is any region inside the nucleus that tends to have higher or lower degree of motion within a certain period of time. To answer this question, we propose a technique which uses high dimensional point set clustering to group sites based on their mobility properties. We then illustrate how we employ this idea to identify spatial patterns by splitting the nucleus into various almost-independent mobility zones. Our second goal is to identify sets of sites that seem to be moving together, as if linked by a slightly deformable chain (sub-structure identification). Since each such site is associated with certain chromosome, knowledge on such topology preserving structures could also lead to better understanding of chromosomes organization.

Chapter 2

Cosegmentation of Images using histogram matching

2.1 Problem Description and Previous Works

Cosegmentation refers to the simultaneous segmentation of the same object (or similar regions) from two (or more) images. It was recently proposed by Rother et al. [85] in the context of segmenting an object of interest from an image pair (e.g., an image of a person taken with different backdrops). The idea has since found applications in segmentation problems in videos [23]. The model also nicely captures important settings in several biological and medical imaging applications. In flourences microscopic imaging, we often encounter scenarios where a stack of 2D images (with neighborhood relationship bewteen consequtive slices) representing a 3D object need to be simultaneously segmented to yield a smooth contour of object. The cosegmentation approach is a natural formulation for such problems. Also, in several applications in medical imaging, two different scans of the same patient (usually on different days) are acquired during a treatment procedure. In order to jointly segment a similar region in both scans, typically, a registration process that puts them in the same coordinate system is used as a first step. This is computationally intensive (and typically non-convex); as a result, gradient descent or line-search heuristics are employed for optimization [81]. But if our interest is analyzing only a specific region present in both images, *cosegmentation* offers an attractive alternative. Here, cosegmentation is a tool to bypass registration, but one may imagine a setting at the other extreme – where a pair of images have little in common *except* the imaged (foreground) object. In such situations, image registration is not very meaningful. Rather, akin to human perception, we may want to automatically extract only the 'coherence' in the image

pair. As an example, consider Fig. 2.1, where approximately the same object appears in the two images. Notice that the background and the object's spatial position in the respective image(s) may be unrelated. We would like to extract the coherent regions from both images simultaneously.

Figure 2.1: The same object in different positions in two images with different backgrounds.

Segmentation of objects and regions in multiple images has typically been approached in a class-constrained fashion. That is, given a large set of images of the object (or an object class) of interest, how can we solve the task of segmenting or recognizing the object in a set of unannotated images? Of course, one option is to use a set of hand-segmented images or a manually specified model(s) – an approach employed in several papers, see [13; 29] and references therein. Such training data has also been successfully used for performing segmentation and recognition in parallel [105; 106], and for segmentation in a level-sets framework [84]. A number of ideas have been proposed for the unsupervised setting as well: [103] suggests using a database of images that have not been segmented using a generative probabilistic model, and [14] learns the figure-ground labeling by an iterative refinement process. In [85], the object of interest is segmented using just one additional image. The authors approach the problem by observing that similar regions (that we desire to segment) in a pair of images will have similar histograms, noting that such a measure has been successfully used for region matching [100]. A generative model is proposed – assuming a Gaussian model on the target histogram (of the to-be-segmented foreground) characterized by a parameter θ; they suggest maximizing the posterior probability that the foreground models in both images are the same. The formulation results in a challenging optimization problem, which requires an iterative minimization procedure. Some clever observations allow the solution of subproblems using graph-cuts, unfortunately, the approximation factor guarantees obtained for each iteration (subproblem) do not carry over to the original optimization. Recently, [96] proposed

constructing a Joint Image Graph for soft image segmentation and calculating point correspondences in the image pair. Because the underlying discrete problem is hard, they used a relaxation method followed by an iterative two step approach to find the solution.

In this chapter, we propose a linear programming framework for the problem. Inspired by [85], we seek to match the histograms[1] of the segmented regions. But rather than a generative model (with predefined distribution families) for the histograms, we focus on the algebraic representation, so we may impose the matching requirement as a constraint. Our successive model includes histogram constraints as additional appropriately regularized terms in the segmentation objective function. However, we consider the structure of the model using the squared L_2 distance (SSD) for measuring histogram similarity, instead of the L_1-norm. After linearization and adjustments, the constraint matrix of the resultant optimization model exhibits some interesting combinatorial properties – especially in terms of the determinants of its submatrices and what can be said about the LP solutions of the 'relaxation'. Specifically, unlike the (harder) L_1-norm problem, the constraint matrix of the proposed model can be shown to have a 2-modular structure, which implies that the LP solution of the relaxed model contains *only* multiples of "half-integral" (instead of arbitrary fractional) values. This allows the design of a simple rounding scheme that gives good solutions in practice and enables bounding the sub-optimality due to rounding under some conditions. We analyze these issues in Section 2.3.1 and Section 2.3.2. Finally, we present experimental results in Section 3.5 and concluding remarks in Section 2.5.

2.2 Histogram matching

Let the two input images be $I_i = [I_i(j)], i \in \{1, 2\}, j \in \{1, \cdots, n\}$, where each image has n pixels. An intensity histogram is a representation of the frequency of intensity values in an image. Let the histogram bins be given by sets $H_1, \cdots, H_K$, where H_k corresponds to the k-th bin. For each image I_i, a coefficient matrix $\mathbf{B}_i$ of size $n \times K$ is such that for pixel j and histogram bin H_k,

$$B_i(j,k) = \begin{cases} 1 & \text{if } I_i(j) \in H_k; \\ 0 & \text{otherwise.} \end{cases} \tag{2.1}$$

The entry $B_i(j,k)$ is 1 if pixel intensity $I_i(j)$ belongs to $H_k^{\{i\}}$ (i.e., the intensity of $I_i(j)$ is in bin k), where i refers to the first or the second image. We can see that

[1] We consider intensity histograms in our implementation, though the proposed model can be directly extended for other types of histograms (incorporating texture features, etc.) as well.

summing over the columns of $\mathbf{B}_i$ gives the histogram for each image as a vector. Now, consider $\mathbf{X}_1$ and $\mathbf{X}_2$ as a pair of $\{0,1\}$ assignment vectors for images I_1 and I_2, that specifies the assignment of pixels to foreground and background regions by a segmentation method. For $i \in \{1,2\}$, $X_i(j) = 1$ if $I_i(j)$ is classified as foreground and 0 otherwise. We want to cosegment (i.e., by assigning to foreground) two regions from the image pair with the requirement that the two histograms of the foreground pixels are similar. The histogram, $\mathbf{H}^{\{i\}}$ for image I_i, for the pixels assigned to the *foreground* is

$$H_k^{\{i\}} = \sum_{j=1}^{n} B_i(j,k) X_i(j) \quad \forall k \in \{1,2,\cdots,K\}, \quad \forall i \in \{1,2\}. \tag{2.2}$$

Since $X_i(j)$ is 0 if pixel j is a background pixel, we can simply focus on the histogram of the foreground pixels and penalize the variation. We note that an expression similar to (2.2) was discussed in a technical report accompanying [85] to obtain a supermodularity proof.

Modeling image segmentation problems as maximum a posteriori (MAP) estimation of Markov Random Fields with pairwise interactions has been successful [16; 46] in the computer vision community and gives good empirical results. It seems quite suitable for the present application. In such formulations, given an image I, each pixel (random variable) $p \in I$, can take one among a discrete set of intensity labels, $T = \{t_1, t_2, \cdots, t_d\}$. The pixel must (ideally) be assigned a label $f_p \in T$ similar to its original intensity, to incur a small *deviation (or data) penalty*, $D(p, f_p)$; simultaneously two adjacent (and similar) pixels, p, q, must be assigned similar labels to avoid a high *separation (or smoothness) penalty* $W(p,q) : p \sim q$ (where $\sim$ indicates adjacency). This gives the following objective function for arbitrary number of labels,

$$\min \sum_{j \sim l} W(j,l) Y(j,l) + \sum_{f_j \in T} \sum_{j=1}^{n} \hat{X}(j, f_j) D(j, f_j)) \quad j, l \in I, \tag{2.3}$$

where $Y(j,l) = 1$ indicates that j and l are assigned to different labels and $\hat{X}(\cdot,\cdot)$ gives the pixel-to-label assignments. Proceeding from (2.2) and allowing for cases where the histograms do not match *perfectly*, we can specify the following simple binary model where $X, Y \in \{0,1\}$ (for the two label case) using a user defined error tolerance, ϵ

$$\begin{aligned} \min \quad & \sum_{i=1}^{2} \sum_{j \sim l} W_i(j,l) Y_i(j,l) + \sum_{i=1}^{2} \sum_{j=1}^{n} X_i(j) D_i(j) \\ s.t. \quad & \left| \sum_{j=1}^{n} B_1(j,k) X_1(j) - \sum_{j=1}^{n} B_2(j,k) X_2(j) \right| \leq \epsilon \quad \forall k, \\ & |X_i(j) - X_i(l)| \leq Y_i(j,l) \quad \forall i, \quad \forall (j \sim l). \end{aligned}$$

2.3 Successive model

The previous model may be modified by including the histogram constraint as an additional (regularized) term in the objective function. We note that the difficulty of the resultant model depends critically on the choice of the norm in this additional penalty. Using the squared L_2 distance (rather than L_1) has significant advantages that are revealed once we analyze the structure of the model, as we will discuss shortly. First, we can see that using the squared L_2 distance, the objective is

$$\min \sum_{i=1}^{2}\sum_{j\sim l} W_i(j,l)Y_i(j,l)+\sum_{i=1}^{2}\sum_{j=1}^{n} X_i(j)D_i(j)+\lambda\sum_{k=1}^{K}\left(\sum_{j=1}^{n} B_1(j,k)X_1(j)-\sum_{j=1}^{n} B_2(j,k)X_2(j)\right)^2$$

where λ is the regularizer controlling the relative influence of the histogram difference in the objective. The squared term in the objective function can be linearized as follows. Consider two corresponding bins $H_k^{\{1\}}$ and $H_k^{\{2\}}$ in the two images. Let $|H_k^{\{1\}}| = n_{1k}$ and $|H_k^{\{2\}}| = n_{2k}$, and the number of foreground pixels after the segmentation in those bins is ν_k and $\hat{\nu}_k$ respectively. Then, the squared term estimates the sum of $(\nu_k - \hat{\nu}_k)^2$ over k, i.e., $\sum_k(\nu_k\nu_k - 2\nu_k\hat{\nu}_k + \hat{\nu}_k\hat{\nu}_k)$. We may use an auxiliary variable, $Z_i(j,l)$, such that $Z_i(j,l) = 1$ if both nodes j and l belong to bin $H_k^{\{i\}}$ and are also part of the foreground in image I_i, it is zero otherwise. Then, summing over all such $\mathbf{Z}_i$'s : $i \in \{1,2\}$, gives the terms $\nu_k\nu_k$ and $\hat{\nu}_k\hat{\nu}_k$. Similarly, we may use another auxiliary variable, $V(j,l)$, which is set to 1 if node j (and node l) belongs to bin $H_k^{\{1\}}$ (and bin $H_k^{\{2\}}$), and is part of the foreground in image I_1 (and image I_2). To specify the linearized representation, let us first introduce some notations. Let $\widehat{(j,l)}_i$ denote those pairs (j,l) that satisfy $B_i(j,k) = 1$ and $B_i(l,k) = 1$ for some bin $H_k^{\{i\}}$ and image i (i.e., intra-image links). Also, $\overline{(j,l)}$ denotes those pairs (j,l) that satisfy $B_1(j,k) = 1$ and $B_2(l,k) = 1$ for bins $H_k^{\{1\}}$ and $H_k^{\{2\}}$ (i.e., inter-image links).
The objective function can then be written as

$$\begin{aligned}
\min \quad & \sum_{i=1}^{2}\sum_{j\sim l} W_i(j,l)Y_i(j,l) + \sum_{i=1}^{2}\sum_{j=1}^{N} X_i(j)D_i(j) + \lambda\sum_{k=1}^{K}\left(\sum_{i=1}^{2}\sum_{\widehat{(j,l)}_i} Z_i(j,l) - 2\sum_{\overline{(j,l)}} V(j,l)\right) \quad (2.4)\\
s.t. \quad & X_i(j) - X_i(l) \le Y_i(j,l), \quad X_i(l) - X_i(j) \le Y_i(l,j) \quad \forall i,j,l,\\
& X_i(j) + X_i(l) \le Z_i(j,l) + 1 \quad \forall \widehat{(j,l)}_i,\\
& X_1(j) \ge V(j,l), \quad X_2(l) \ge V(j,l) \quad \forall \overline{(j,l)},\\
& \mathbf{X},\mathbf{Y},\mathbf{Z},\mathbf{V} \in \{0,1\}.
\end{aligned}$$

2.3.1 The constraint matrix in (2.4) and its properties

An alternative to solving the $\{0,1\}$ integer program (IP) in (2.4) directly (using branch-bound methods) is to use the relaxation approach – by relaxing the $\{0,1\}$ requirement to $[0,1]$. We must then develop a rounding scheme to obtain an integral solution from the fractional LP solution. In a general setting, nothing much can be said about the *real* values in the LP solution; as a result, rounding and analysis must be performed to bound the loss and obtain an approximation. However, in some *special cases*, it is well known that the situation is significantly better. For example, recall that if the model has only (monotone) constraints of the form in the first set of inequalities in (2.4) above, the constraint matrix is totally unimodular (i.e., the determinants of each of its square submatrices are in $\{0, \pm 1\}$). By the Hofman-Kruskal theorem [56], the optimal vertex solution of the linear program is *integral*. In cosegmentation, the constraints, $X_i(j) + X_i(l) \leq Z_i(j,l) + 1$, spoil the unimodularity structure (it is non-monotone), however. The question then is whether we can still make any assertions about the values in the solution. Fortunately, the model still retains a desirable structure, which can be inferred by using certain generalizations of unimodular matrices and the determinant property. What we will be able to claim as a result of this property is that the values in the optimal LP solution *cannot* be arbitrary *reals* in $[0,1]$. We first provide some definitions and then move to our main results.

Definition 1 (Nonseparable matrix) *A matrix in nonseparable if there do not exist partitions of the columns and rows to two (or more) subsets C_1, C_2 and R_1, R_2 such that all nonzero entries in each row and column appear only in the submatrices defined by the sets $R_1 \times C_1$ and $R_2 \times C_2$.*

Let the constraint matrix of (2.4) be denoted as $\mathbf{A}$. Below, we outline the key properties of the constraint matrix for the cosegmentation problem.

Lemma 1 $\mathbf{A}$ *is a nonseparable matrix.*

Proof: The entries in $\mathbf{X}$ refer to the pixels of an image. For two pixels that are adjacent w.r.t. the chosen neighborhood system $\mathcal{N}$ (e.g., four neighborhood), the columns of their corresponding $\mathbf{X}$ entries will be non-zero in at least one row (for the constraint where they appear together). Then, they must be in the same partition (either R_1 or R_2). Other 'neighbors' of these pixels must also be in the same partition. Because each pixel is reachable from another via a path in $\mathcal{N}$, the same logic applied repeatedly shows that the $\mathbf{X}$ columns (which correspond to pixels) must be in the

same partition. Each $\mathbf{Y}$ and $\mathbf{Z}$ variable appears in at least one constraint (non-zero entry in $\mathbf{A}$) together with $\mathbf{X}$, and so must belong in the same partition as $\mathbf{X}$. Each $\mathbf{V}$ variable appears in two constraints, with $X_1(\cdot)$, and $X_2(\cdot)$, and so must also be in the same partition. Therefore, $\mathbf{A}$ is non-separable. □

Theorem 1 *The determinant of all submatrices of* $\mathbf{A}$ *belongs to* $\{-2,-1,0,1,2\}$*, i.e., its absolute value is bounded by* 2*.*

Proof: We prove by induction. Since the entries in $\mathbf{A}$ are drawn from $\{-1,0,1\}$, the 1×1 matrix case is simple. Now assume it holds for any $m-1\times m-1$ submatrix and consider $m\times m$ submatrices. By construction, $\mathbf{A}$ (and any non-singular square submatrix) has at most three non-zero entries in a row. Let $\bar{\mathbf{A}}$ be a $m\times m$ non-singular submatrix of $\mathbf{A}$. To obtain the result, we must consider the following three cases.

Case 1: $\bar{\mathbf{A}}$ has one non-zero entry in any row/column.

Case 2: $\bar{\mathbf{A}}$ has two non-zero entries in every row and column.

Case 3: $\bar{\mathbf{A}}$ has three non-zero entries in any row.

It is easy to verify that the structure of $\mathbf{A}$ rules out other possibilities. For presentation purposes, we will start with Case 3 and consider Case 2 last.

Case 3: $\bar{\mathbf{A}}$ **has three non-zero entries in any row.** First, observe that the rows with three non-zero entries must come from the first three constraints in (2.4). In each such constraint, there is at least one variable, $Y_i(j,l)$, $Y_i(l,j)$, or $Z_i(j,l)$, that does not occur in *any other constraint*. Therefore, the corresponding columns of that variable must have only one non-zero entry, say at position (p,q). We may permute $\bar{\mathbf{A}}$ so that $(p.q)$ moves to location $(1,1)$. Let $M_{[u,v]}$ denote the matrix obtained by deleting row u and column v. The determinant of $\bar{\mathbf{A}}$ is expressed as $\bar{A}(1,1)\det(\bar{A}_{[1,1]})$, where $\bar{A}_{[1,1]}$ is $m-1\times m-1$. Now, $\bar{A}(1,1)\in\{\pm 1,0\}$ and $\det(\bar{A}_{[1,1]})\in\{\pm 2,\pm 1,0\}$, so the product of these terms is in $\{\pm 2,\pm 1,0\}$.

Case 1: $\bar{\mathbf{A}}$ **has** 1 **non-zero entries in any row.** The same argument used in case 3 applies here.

Case 2: $\bar{\mathbf{A}}$ **has** 2 **non-zero entries in every row and column.** This case was proved by Hochbaum et al. [47] (Lemma 6.1) and the same idea can be applied here. Wlog, we may assume that the two non-zero entries in row i of $\bar{\mathbf{A}}$ are in columns i and $(i+1)\text{mod } m$ (due to Lemma 1). Hence, $\det(\bar{\mathbf{A}})=\bar{A}(1,1)\det(\bar{A}_{[1,1]})-(-1)^m\bar{A}(m,1)\det(\bar{A}_{[m,1]})$. The submatrix determinants equal 1, since they are both triangular matrices (with non-zero diagonal elements). Thus, $\det(\bar{\mathbf{A}})\in\{\pm 2,\pm 1,0\}$. □

Corollary 1 *The model in (2.4) has super-optimal half integral solutions, i.e., each variable in the optimal LP solution is in* $\{0, \frac{1}{2}, 1\}$.

Corollary 1 is important – it shows that $\mathbf{A}$ has 2-modular structure with half-integral solutions [57]. This property leads to a two-approximation for a wide variety of NP-hard problems including vertex cover and many variations of 2-SAT [47]. We also utilize this property for some of our analysis in the subsequent sections. Finally, a comment on some consequences of half integral solutions. If the objective function has only positive terms, we can round all the $\frac{1}{2}$ variables up to 1, leave the integral variables unchanged, and still ensure that the value of the objective is within a factor of two of the optimal solution [47]. This is not applicable in our case due to the negative term in the objective function. However, we can still obtain approximations if the half integral solution satisfies some conditions, as we show in the next section.

2.3.2 Rounding and Approximation

The approximation depends critically on how the variables are rounded. If we 'fix' the $\{0, 1\}$ variables, and round $\frac{1}{2}$'s to 1, a good approximation may be obtained. However, such a solution may not be feasible w.r.t. the constraints in a worst case setting (although in practice, a simple rounding heuristic exploiting half-integral solutions may work). We discuss this issue next.

We refer to the block of $\mathbf{X}$ variables in the solution vector as $\mathbf{X}$ for convenience; $\mathbf{X}$ includes $\mathbf{X}_1$ and $\mathbf{X}_2$. The corresponding block in the optimal LP solution is given as X_1^*, and X_2^* (in the present context we will refer to X's as sets). Clearly, $X_i^* = X_i^{*|\{0\}} \cup X_i^{*|\{1\}} \cup X_i^{*|\{\frac{1}{2}\}}$, where $X_i^{*|\{0\}}$, $X_i^{*|\{1\}}$ and $X_i^{*|\{\frac{1}{2}\}}$ refers to the 0, 1 and $\frac{1}{2}$ entries in X^* respectively. Let $X_{i|H_k}^{*|\{\frac{1}{2}\}} \subseteq X_i^{*|\{\frac{1}{2}\}}$ refer to those entries that are in histogram bin k in image i (for presentation purposes, we assume H_k in image 1 corresponds to H_k in image 2, and so we drop the image superscript). A subset of the constraints in (2.4) relating $X_{1|H_k}^{*|\{\frac{1}{2}\}}$ and $X_{2|H_k}^{*|\{\frac{1}{2}\}}$ are

$$X_1^*(p) + X_1^*(q) \leq Z_1^*(p,q) + 1 \quad X_1^*(p), X_1^*(q) \in X_{1|H_k}^{*|\{\frac{1}{2}\}}, \tag{2.5}$$

$$X_1^*(q) \geq V^*(q,r) \quad X_1^*(q) \in X_{1|H_k}^{*|\{\frac{1}{2}\}}, \tag{2.6}$$

$$X_2^*(r) \geq V^*(q,r) \quad X_2^*(r) \in X_{2|H_k}^{*|\{\frac{1}{2}\}}, \tag{2.7}$$

$$X_1^*(p) \geq V^*(p,r) \quad X_1^*(p) \in X_{1|H_k}^{*|\{\frac{1}{2}\}}, \tag{2.8}$$

$$X_2^*(r) \geq V^*(p,r) \quad X_2^*(r) \in X_{2|H_k}^{*|\{\frac{1}{2}\}}. \tag{2.9}$$

First, consider the $\frac{1}{2}$-valued entries. Since $X_1^*(p)$, $X_1^*(q)$ and $X_2^*(r)$ are all $\frac{1}{2}$, and (2.4) is a minimization, $Z_1^*(p,q)$ is 0 and $V^*(q,r)$ is $\frac{1}{2}$ at optimality. Setting

$X_1^*(p), X_1^*(q)$ and $X_2^*(r)$ to 1 satisfies (2.6)-(2.9), regardless of the value of $V^*(\cdot,\cdot)$, but (2.5) is violated if $Z_1^*(p,q)$ is unchanged. On the other hand, if we round $X_1^*(q)$ (or $X_1^*(p)$) to 0, (2.5) is satisfied, but (2.6) or (2.8) is violated. Therefore, to ensure a feasible solution, we must round a few $\frac{1}{2}$ variables (V and X) to 0, and set a few Z variables (which were 0 in the optimal solution) to 1. In the general case, this might increase the objective function arbitrarily, making it difficult to obtain an approximation. However, if the optimal solution satisfies some conditions, we can still bound the gap.

For convenience, let $a_{ik} = |X_{i|H_k}^{*|\{1\}}|$, $b_{ik} = |X_{i|H_k}^{*|\{\frac{1}{2}\}}|$, and $c_{ik} = |X_{i|H_k}^{*|\{0\}}|$. Hence, if histogram bin H_{ik} has n_{ik} pixels, $n_{ik} = a_{ik} + b_{ik} + c_{ik}$. We 'fix' the integral X variables in the rounding process, and only the half-integral variables are adjusted to either 0 or 1. Also, since we will analyze the solution bin-wise, the subscript k is implicit and will be removed for notational convenience, i.e., a_i, b_i, c_i denote a_{ik}, b_{ik}, c_{ik} respectively. Suppose for H_{ik}, a rounding scheme sets $b_i^{(1)}$ entries to 1 and $b_i^{(0)}$ entries to 0, so $b_i = b_i^{(1)} + b_i^{(0)}$. In this section, our focus is to analyze the increase in the histogram mismatch penalty in the objective function value due to rounding. In the optimal solution, let this term evaluate to

$$O_{H_k}^* = \lambda(O_Z^* - 2O_V^*) = \lambda\left(\sum_{i=1}^{2}\sum_{\widehat{(j,l)}_i} Z_i^*(j,l) - 2\sum_{\overline{(j,l)}} V_{jl}^*\right) \tag{2.10}$$

We now represent (2.10) in terms of a_i, b_i and c_i.

$$O_Z^* = \sum_{i=1}^{2}\sum_{\widehat{(j,l)}_i \in H_k} Z_i^*(j,l) = \sum_{i=1}^{2}(a_i^2 + \frac{1}{2}a_i b_i) \tag{2.11}$$

$$O_V^* = \sum_{\overline{(j,l)} \in H_k} V_{jl}^* = a_1 a_2 + \frac{1}{2}b_1 b_2 + \frac{1}{2}a_1 b_2 + \frac{1}{2}a_2 b_1 \tag{2.12}$$

We know that each variable in the $\mathbf{Z}^*$ and $\mathbf{V}^*$ blocks corresponds to a pair of pixels i.e., $\mathbf{X}$'s. These pixels belong to the same image for $\mathbf{Z}^*$ and different images for $\mathbf{V}^*$. Now, let us analyze the values of $\mathbf{Z}^*$ variables in the optimal LP solution. Only two cases may arise if $\mathbf{Z}_i^*(\cdot,\cdot)$ is non-zero. It is 1 when both items in the pair are taken from $X_{i|H_k}^{*|\{1\}}$ (cardinality of this set is a_i^2). The second case is when it is set to $\frac{1}{2}$, this is when one item in the pair is taken from $X_{i|H_k}^{*|\{1\}}$ and the other from $X_{i|H_k}^{*|\{\frac{1}{2}\}}$ (cardinality of this set is $a_i b_i$). V^* is also set accordingly by looking at the distinct combinations. This analysis explains (2.11) and (2.12), so we may represent the histogram term in the optimal solution as a function of the number of 1- and $\frac{1}{2}$-valued variables.

We now enumerate the respective increase (or decrease) of these variables in the objective function value due to rounding in a case-by-case basis. Table 2.1 shows the cost-variation due to Z variables and V variables, see the caption for details. For instance, consider #1 in Table 2.1 (bottom). This refers to the case when one X variable in the pair is from $X^{*|\{1\}}_{i|H_k}$ and the other is from the subset of $X^{*|\{\frac{1}{2}\}}_{i|H_k}$ variables which are rounded to 1. All such Z^* variables are set to $\frac{1}{2}$ in the optimal LP solution, but must be rounded to 1 in the integral solution to satisfy the constraint. There are $a_1 b_1^{(1)} + a_2 b_2^{(1)}$ such variables, where each increases by $\frac{1}{2}$. Hence, the net increase is $\frac{1}{2}a_1 b_1^{(1)} + \frac{1}{2}a_2 b_2^{(1)}$. Similarly, consider #2 in Table 2.1 (top). This refers to the case when one X variable in the pair is from $X^{*|\{1\}}_{1|H_k}$ and the other is from the subset of $X^{*|\{\frac{1}{2}\}}_{2|H_k}$ variables which are rounded to 1. All such V^* variables are set to $\frac{1}{2}$ in the optimal LP solution, but must be rounded to 1 in the integral solution to ensure feasibility. This is a decrease in the overall value of the objective by a value $\frac{1}{2}a_1 b_2^{(1)}$, since there are $a_1 b_2^{(1)}$ such variables. All other cases can be similarly calculated.

Summing over the increase and decrease columns of both variables, gives the net increase in the objective function due to rounding. Let the increase in $O^*_{H_k}$ due to rounding be $\rho = \rho_Z + \rho_V$, where ρ_Z (and ρ_V) is the increase due to Z (and V). These can be expressed as

$$\rho_Z = \left(b_1^{(1)}\right)^2 + \frac{1}{2}a_1 b_1^{(1)} - \frac{1}{2}a_1 b_1^{(0)} + \left(b_2^{(1)}\right)^2 + \frac{1}{2}a_2 b_2^{(1)} - \frac{1}{2}a_2 b_2^{(0)} \tag{2.13}$$

$$\rho_V = \frac{1}{2}\left(b_1^{(0)}b_2^{(0)} + b_1^{(1)}b_2^{(0)} + b_2^{(1)}b_1^{(0)} + a_1 b_2^{(0)} + a_2 b_1^{(0)} - b_1^{(1)}b_2^{(1)} - a_1 b_2^{(1)} - a_2 b_2^{(1)}\right) \tag{2.14}$$

By standard manipulations and using $b_i = b_i^{(0)} + b_i^{(1)}$, ρ_V can be simplified as follows.

$$\rho_V = \frac{1}{2}(b_1 b_2 + a_1 b_2 + a_2 b_1 - 2b_1^{(1)}b_2^{(1)} - 2a_1 b_2^{(1)} - 2a_2 b_2^{(1)}) \tag{2.15}$$

We now go back to the optimal solution and express it in terms of a and b. Plugging in the values of (2.11) and (2.12) in (2.10), we get

$$\begin{aligned} O^*_Z - 2O^*_V &= \sum_{i=1}^{2}(a_i^2 + \frac{1}{2}a_i b_i) - 2a_1 a_2 - b_1 b_2 - a_1 b_2 - a_2 b_1 \\ &= (a_1 - a_2)^2 + \frac{1}{2}a_1 b_1 + \frac{1}{2}a_2 b_2 - (b_1 b_2 + a_1 b_2 + a_2 b_1) \end{aligned} \tag{2.16}$$

To analyze the loss due to rounding, we must analyze the conditions under which we can establish a relationship between a lower bound for (2.16), and an upper bound

#	Pairs	Increase	Decrease
1	a_1 with a_2	–	–
2	a_1 with $b_2^{(1)}$	–	$\frac{1}{2}a_1 b_2^{(1)}$
3	a_1 with $b_2^{(0)}$	$\frac{1}{2}a_1 b_2^{(0)}$	–
4	a_2 with $b_1^{(1)}$	–	$\frac{1}{2}a_2 b_1^{(1)}$
5	a_2 with $b_1^{(0)}$	$\frac{1}{2}a_2 b_1^{(0)}$	–
6	a_1 with c_2	–	–
7	a_2 with c_1	–	–
8	$b_1^{(1)}$ with $b_2^{(1)}$	–	$\frac{1}{2}b_1^{(1)} b_2^{(1)}$
9	$b_1^{(1)}$ with $b_2^{(0)}$	$\frac{1}{2}b_1^{(1)} b_2^{(0)}$	–
10	$b_1^{(0)}$ with $b_2^{(1)}$	$\frac{1}{2}b_1^{(0)} b_2^{(1)}$	–
11	$b_1^{(0)}$ with $b_2^{(0)}$	$\frac{1}{2}b_1^{(0)} b_2^{(0)}$	–
12	c_1 with c_2	–	–

#	Pairs for $i = 1, 2$	Increase	Decrease
1	a_i with $b_i^{(1)}$	$\frac{1}{2}a_1 b_1^{(1)} + \frac{1}{2}a_2 b_2^{(1)}$	–
2	a_i with $b_i^{(0)}$	–	$\frac{1}{2}a_1 b_1^{(0)} + \frac{1}{2}a_2 b_2^{(0)}$
3	$b_i^{(1)}$ with $b_i^{(1)}$	$\left(b_1^{(1)}\right)^2 + \left(b_2^{(1)}\right)^2$	–
4	$b_i^{(0)}$ with $b_i^{(0)}$	–	–
5	$b_i^{(1)}$ with $b_i^{(0)}$	–	–
6	c_i with c_i	–	–

Table 2.1: Case by case accounting the increases and decreases in objective function value due to V (top table) and Z (bottom table). The left-most columns in the left and right tables enumerate the specific pairs, e.g., "a_1 with a_2" refers to the case where one item in the pair is picked from $X_{1|H_k}^{*|\{1\}}$ and the other from $X_{2|H_k}^{*|\{1\}}$.

on $\rho_Z + 2\rho_V$ given in (2.13) and (2.14). First, we look at (2.16) assuming $a_1 \geq$

$\alpha a_2,\ a_i \geq \alpha b_i,\ b_1 \geq b_2$.

$$
\begin{aligned}
O_Z^* - 2O_V^* &= (a_1 - a_2)^2 + \frac{1}{2}a_1b_1 + \frac{1}{2}a_2b_2 - (b_1b_2 + a_1b_2 + a_2b_1) && (2.17)\\
&\geq \left(a_1 - \frac{a_1}{\alpha}\right)^2 + \frac{1}{2}a_1b_1 + \frac{1}{2}a_2b_2 - (b_1b_2 + a_1b_2 + a_2b_1) && (2.18)\\
&\geq \left(a_1 - \frac{a_1}{\alpha}\right)^2 + \frac{1}{2}\alpha b_1^2 + \frac{1}{2}\alpha b_2^2 - b_1^2 - a_1\frac{a_2}{\alpha} - a_2\frac{a_1}{\alpha} && (2.19)\\
&= a_1^2\left(1 - \frac{1}{\alpha}\right)^2 + \left(\frac{\alpha}{2} - 1\right)b_1^2 + \frac{\alpha}{2}b_2^2 - \frac{2}{\alpha}a_1a_2 && (2.20)\\
&\geq a_1^2\left(1 - \frac{1}{\alpha}\right)^2 + \left(\frac{\alpha}{2} - 1\right)b_1^2 + \frac{\alpha}{2}b_2^2 - \frac{2}{\alpha^2}a_1^2 && (2.21)\\
&= a_1^2\left((1 - \frac{1}{\alpha})^2 - \frac{2}{\alpha^2}\right) + \left(\frac{\alpha}{2} - 1\right)b_1^2 + \frac{\alpha}{2}b_2^2 && (2.22)
\end{aligned}
$$

Setting $\alpha > 1 + \sqrt{2}$, ensures a positive lower bound for the optimal solution of the objective function. Notice, that it is difficult to yeid an approximation for values of α lower than $1 + \sqrt{2}$, because the value of the optimal solution may possibly be negative. We now establish an upper bound from the increase using the same values

$$
\begin{aligned}
\rho_Z + 2\rho_V &= (b_1^{(1)})^2 + (b_2^{(1)})^2 + \frac{1}{2}a_1(b_1^{(1)} - b_1^{(0)}) + \frac{1}{2}a_2(b_2^{(1)} - b_2^{(0)}) + a_1(b_2^{(0)} - b_2^{(1)}) + a_2(b_1^{(0)} - b_1^{(1)}) + b\\
&\leq b_1^2 + b_2^2 + \frac{1}{2}a_1b_1 + \frac{1}{2}a_2b_2 + a_1b_2 + a_2b_1 + b_1^2\\
&\leq 2b_1^2 + b_2^2 + \frac{3}{2}a_1^2(\frac{\alpha+1}{\alpha^2})
\end{aligned}
$$

If we set $c = \max(\frac{3(\alpha+1)}{2(\alpha^2-2\alpha-1)}, \frac{4}{\alpha-2})$, then it follows that $\rho_Z + 2\rho_V \leq c.(O_Z^* - 2O_V^*)$. The value of c gets progressively better as the value of α increases. Setting $\alpha = 3$, c is equal to 4. For this approximation to hold overall, each bin must satisfy one of the following conditions.

- Either the bin has no $\frac{1}{2}$ X variables, in this case, $b_i = 0$. Here there is no increase in the objective function after rounding, or
- For $i \in \{1, 2\}$, $a_1 \geq \alpha a_2,\ a_i \geq \alpha b_i,\ b_1 \geq b_2$ keeping the histogram bin index fixed. This also requires $\alpha > 1 + \sqrt{2}$.

For $i \in \{1, 2\}$, we can prove the following result (proof provided in the supplement):

Lemma 2 *If $a_1 \geq \alpha a_2,\ a_i \geq \alpha b_i,\ b_1 \geq b_2$, then $\rho_Z + 2\rho_V \leq C \cdot (O_Z^* - 2O_V^*)$ where $C = \max(\frac{3(\alpha+1)}{2(\alpha^2-2\alpha-1)}, \frac{4}{\alpha-2})$. For $\alpha = 3$, $\rho_Z + 2\rho_V \leq 4 \cdot (O_Z^* - 2O_V^*)$.*

In Lemma 2, the increase is bounded by a multiple of the lower bound $(O^*_Z - 2O^*_V)$ if the conditions are satisfied. Notice that C decreases with an increase in α. What remains to be addressed is (1) to specify what the rounding scheme $\mathcal{R}$ is, and (2) the loss due to rounding for the MRF terms in the objective in (2.4). First, consider $\mathcal{R}$: how to round $X_i^{*|\{\frac{1}{2}\}}$ to $\{0,1\}$. To do this, we solve the original MRF problem in (2.4) (but without the histogram constraints) on both images using max-flow/min-cut. Let γ_{MRF} be the value of such a solution and $\gamma^{(\frac{1}{2})}_{MRF}$ be the part of γ_{MRF} corresponding only to variables in $X_i^{*|\{\frac{1}{2}\}}$. Clearly, $\gamma^{(\frac{1}{2})}_{MRF} \leq \gamma_{MRF}$. We round $X_i^{*|\{\frac{1}{2}\}}$ variables based on their assignment in the MRF solution. Let $O^*_{X,Y} = \sum_{i=1}^{2} \sum_{j \sim l} W_i(j,l) Y_i^*(j,l) + \sum_{i=1}^{2} \sum_{j=1}^{n} X_i^*(j) D_i(j)$ be the optimal LP solution. Again, $O^*_{X,Y}$ can be written as $O^*_{X,Y} = O^*_{X,Y}(0,1) + O^*_{X,Y}(\frac{1}{2})$. Since γ_{MRF} is the smallest possible solution for the original (unconstrained) MRF, $\gamma^{(\frac{1}{2})}_{MRF} \leq \gamma_{MRF} \leq O^*_{X,Y}$. The solution after rounding is $O_{X,Y} = O_{X,Y}(0,1) + \gamma^{(\frac{1}{2})}_{MRF} \leq O^*_{X,Y} + O^*_{X,Y} = 2O^*_{X,Y}$. We can now state the following result:

Theorem 2 *If $a_1 \geq \alpha a_2$, $a_i \geq \alpha b_i$, $b_1 \geq b_2$ where $\alpha = 3$, then $O_{X,Y,Z,V} \leq 5 \cdot O^*_{X,Y,Z,V}$.*

2.4 Experimental results

We performed evaluations of our cosegmentation algorithm on several stereo image pairs as well as as imges of chormosomes in the nucleus obtained as 3D stacks using flourescent microscope. The image sizes were 128×128, and we used the RBF kernel to calculate the deviation $D(\cdot)$ and separation penalties $W(\cdot,\cdot)$ in (2.4). In the current implementation, histogram consistency was enforced using RGB intensities and gradients only, but the histogram specification can be easily extended to use texture features, for example.

2.4.1 Experiments on Stereo Images - Comparisons with Graph Cuts

In this section, we performed evaluations of our cosegmentation algorithm on several image pairs used in [85] (see http://research.microsoft.com/∼carrot/software.htm) together with additional image pairs that we collected from miscellaneous sources. We cover (a) qualitative results of segmentation, (b) error rate, i.e., percentage of misclassified pixels (using hand-segmented images), and (c) empirical calculation of the loss in optimality due to rounding. We also illustrate results from a graph-cuts

based MRF segmentation method (GC) applied on the images independently and evalute based on criterias (a) and (b) mentioned before.

In Figure 2.4.1 we show cosegmentation results for a set of image pairs with a similar object in the foreground but different backgrounds. In the first row (Stone), the stone in image-2 is larger (which gives dissimilar foreground histograms), but the algorithm successfully segments the object in both images. Notice that our method compares well to Rother et al., on this image pair (see [85]). For the next image pair (Banana), our algorithm is able to recognize the object in both images but shows a significant improvement in image-2 compared to GC with the same parameter settings. For the third image pair (Woman), our algorithm segments roughly the same foreground as GC for image-1. For image-2, several regions in the background with similar intensities as the foreground are segmented by GC. Using the histogram constraints eliminates these regions from the segmentation giving a cleaner and more accurate segmentation. In the fourth (Horse) and fifth (Lasso) image pairs, GC does not perform well on image-1, but yields similar results as our method on image-2. Note that the object (foreground) in Lasso image-1 is difficult to discern from the background making it a challenging image to segment. However, using the histogram constraints, we are able to obtain a reasonable final segmentation.

Instance	λ	Coseg error rate	approximation	GC error rate
Stone	1.0	1.56%	1.04	3.57%
Banana	0.01	3.02%	1.02	7.75%
Woman	0.01	2.14%	1.06	$> 10\%$
Horse	0.01	4.80%	1.02	5.70%
Lasso	0.01	2.87%	1.03	7.9%

Table 2.2: Misclassification errors (percentage of pixels misclassified) in the segmentation and empirical approximation estimates of the cosegmentation solution.

The specific value of λ for each image pair was determined empirically by iterating over four possible values – starting from $\lambda = 0.001$ and increasing it by an order of magnitude until $\lambda = 1.0$. The specific value of λ for each image pair is listed in Table 2.2. We found that if $\lambda < 0.0001$, the results are similar to those obtained by graph-cuts segmentation. Table 2.2 also shows the error rate (i.e., the average pixel misclassification error for each image pair). These were generated by comparing the segmentation with hand segmented images. In general, the misclassification error rate was $1 - 5\%$, which is an improvement over GC as shown in Fig. 2.4.1 and Table

2.2. The empirical estimate of the approximation error also shows that the integral solution is within a factor of 1.1 of the lower bound (LP solution).

2.4.2 Experiments on Additional Stereo Images

In this section, we provide some additional cosegmentation results in Fig. 2.4.2. The mean of misclassification error rate for the image-pairs shown in Fig. 2.4.2 was (a) Bus: 7.45%, (b) Knut: 5.74%, (c) Woman: 2.70%, (d) Coke can: 2.34%.

2.4.3 Cosegmentation on Biological Image Stacks

Fluorescence microscopy has provided scientists with tools to image and analyze microscopic structures as three dimensional objects. This process typically results in a series (or slices) of images of the 3D object (such as chromosomes or chromatin domains) derived from different focal planes (called the Z-stack). However, inspite of using deconvolution techniques, fluorescence microscopy suffer from significant blurring artifacts (caused due to emmision of light from structures which are out of focus), which makes it difficult to understand the structural properties of the microscopic objects clearly. To mitigate this problem, additional image processing techniques are used to further de-noise and segment contours of the subcellular structures for further processing. One such segmentation approach is the threshold based approach, which chooses a single intensity threshold to apply to all images of the stack individually. Though threshold-based segmentation works nicely for individual images in the stack, the segmentation of object contours when viewed across consecutive images are often somewhat discontinuous (note that most segmentation algorithms are applicable on a single image and does not provide a natural way to improve segmentation by sharing information between related images such as consecutive slices in this case). A continuous and smooth segmentation across the Z-stack can be used to generate more accurate object contours and hence a realistic visualization of the subcellular structures.

In this section, we show the cosegmetation approach can be effectively used to accurately segment 3D stack of images obtained from fluorescent microscopy. For our purpose, we look at images of chromosomes in the nucleus, acquired using a process called 'chromosome painting'. In this process, collections of nucleic acid sequences known to hybridize with a specific chromosome are assembled. Sequences specific for every chromosome are then converted to probes labeled with a fluorescent dye. The cell is then hybridized with a chromosome's paint probe, which results in specific

labeling of that chromosome. Fluorescent signals are acquired in 3D together with phase contrast images of cells in transmission light. A few images of such an image stack is shown in Figure 2.4.3 (top row).

Notice that unlike the stereo images in the previous sections, not only the foreground but also the backgrounds are similar from one image to the next. Hence, the intensity histograms of both the foreground and backgrounds may be similar, which complicates the process of cosegementation. Therefore, instead of using just the intensities, we also incorporate Gabor texture features [89] and distance based features into the histogram as well. We compute gabor texture features of each image as per the parameters specified in [89] and then further subdivide the intensity histogram bins, by clustering the pixels based on their gabor features. The intensity-texture bins formed can be further subdivided, by clustering pixels based on their distances to each other in the same image and in the consecutive images.[2] As the final step, the user is asked to indicate a pixel which can be definitely classified as background. We identify the bin to which this pixel belongs and all bins whose mean intensity is lower than the mean intensity of this bin. Such bins are then 'shifted' in one of the images so that no longer correspond to similar bins in the other images.

Figure 2.4.3 shows the slices $5-7$ of a stack containing 10 slices. These correspond to approximately the middle of the Z-stack. The result of segmentation are shown in Figure 2.4.3 (bottom row). Cosegmentation can be performed with any number of slices as long as the histogram constraints are Z-stack enforced between consecutive slices. Time taken by the algorithm for two images at a time is approximately 30 seconds.

2.5 Conclusions

We propose a linear programming based algorithm for the cosegmentation problem. The model uses an objective function with MRF terms together with a penalty that depends on the sum of squared differences of the foreground regions' histograms. We prove some interesting properties of the constraint matrix of cosegmentation LP, specifically that the optimal LP solution will only have values in $\{0, \frac{1}{2}, 1\}$. Half integrality leads to a simple rounding strategy that gives good segmentations in practice and also allows an approximation analysis under some conditions. We believe these to be useful contributions for this problem. A direction of future research will be to

[2]This step is possible, since the relative positions of the chromosome does not vary much from one slice to the next.

investigate if such properties can lead to purely combinatorial algorithms for cosegmentation.

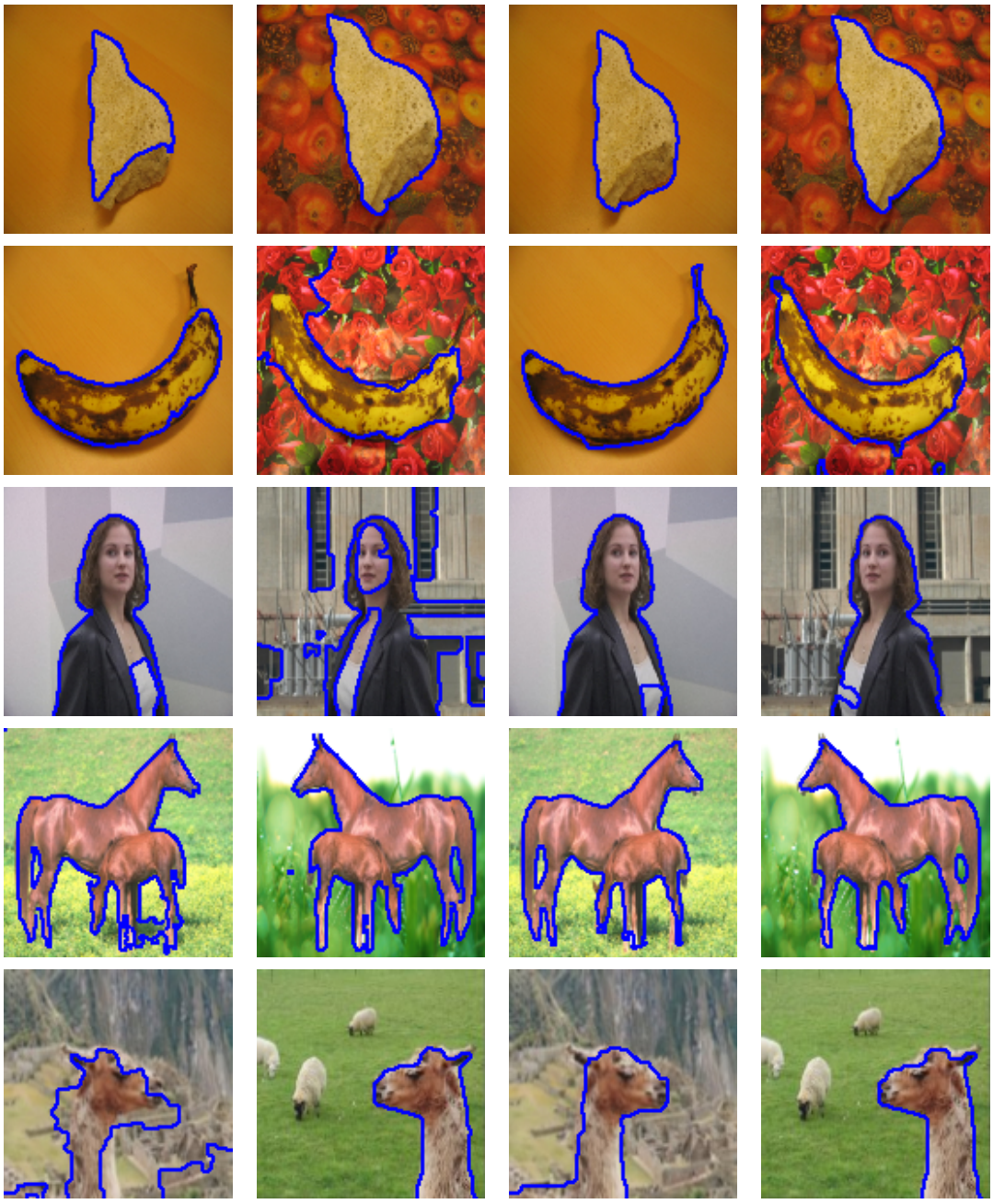
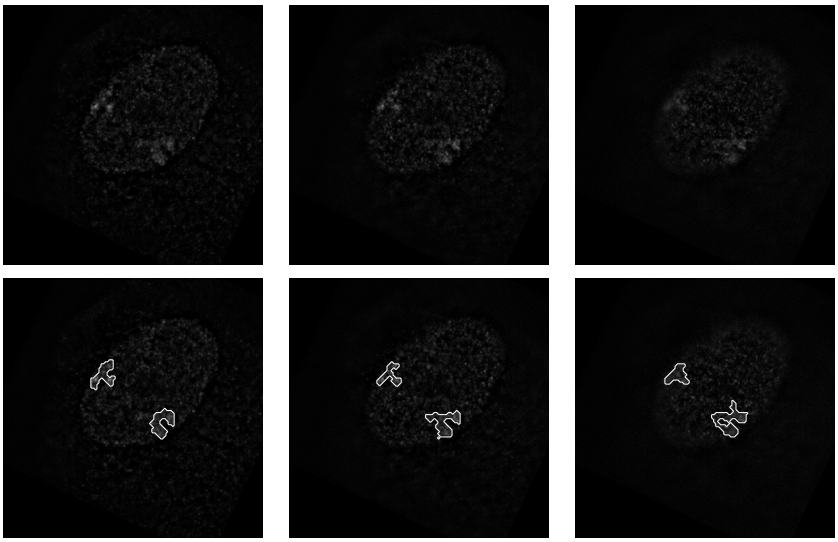


Figure 2.2: The images in columns (a) and (b) are solutions from independent graph-cuts based segmentations on both images, the pair of images in columns (c) and (d) are solutions from our cosegmentation algorithm applied on the images simultaneously. The segmentation is shown in blue.

Figure 2.3: The original images are shown in columns (a) and (b), the pair of images in columns (c) and (d) are solutions from our cosegmentation algorithm applied on the images simultaneously. The segmentation boundary is given in blue. The Bus, Knut, and Coke can image pairs were obtained from Flickr. The Woman image pair was taken from http://grail.cs.washington.edu/projects/digital-matting/video-matting/



(a) (b) (c)

Figure 2.4: Application of cosegmentation of chormosome image stacks

Chapter 3

Generalized Median Graphs

3.1 Problem Description and Previous Works

Graphs are ubiquitous in many application areas where information regarding object-to-object relationships needs to be encoded. There is an abundance of graph theoretic algorithms being employed for many fundamental problems in computer vision, data mining, and artificial intelligence. Efficient algorithms for a variety of problems such as segmentation [91], spatial data-mining [74], clustering [18], belief representation [90] and shape matching [93] are based on tools such as spectral methods, graph cuts, bipartite matching and minimum spanning trees. More recently, a number of works have also focused on developing scalable methods for mining, clustering, classification and search using graph-based representations [28; 104]. Some novel applications also include mining frequent substructures in protein data, biomedical text-mining, and pattern recognition tasks in biomedical imaging.

In this chapter, we concentrate on a modeling problem – to build an exemplar from a set of graphs. This is motivated from an important biological imaging problem which we will introduce shortly. To give an intuition, consider a sequence of graph representations of a dynamic 'scene'. Each graph in the set is different as a result of either noise, distortion, or movement of the 'actors' in the scene. We are then faced with the task of building *a single* composite model describing the scene in the best possible way. Problems of this form are broadly known as *Prototype Learning*, and require learning a model of a class given several of its members. The Generalized Median Graph problem asks following question: given a set of graphs, what is a *median* graph (not necessarily from the input set) that is a good representative (or prototype) for the set?

It seems natural that a good model is one that is close to each item (or repre-

sentation) in the set. In other words, it minimizes the sum of 'distances' to the set elements. A model that even broadly fits this definition directly serves two useful purposes. First, answers to higher level questions regarding the set no longer need evaluation of each individual set item. A good model is sufficient to address such questions and is attractive in terms of efficiency. Second, individual items in the set may have outliers that are not necessarily standard indicators of the characteristics of the set. Here, a good model addresses the need for accuracy and reliability. The generalized median graph problem is, thus, a natural formalization for many problems arising in structural pattern recognition, computer vision, bioinformatics, data mining and a number of other application areas.

In this chapter, we study a new approach for generating a *median* from a set of input graphs using a linear programming framework. We start by formulating the problem in Section 3.1.1 and then review some of the previous works for this problem as well as a closely related problem of graph isomorphism (in Section 3.1.2). Next, we describe how the graph isomorphism cost model can be generalized for attributed graphs (in Section 3.2.1). Later, we describe how this cost model can be incorporated into our optimization model (in Section 3.2.2, Section 3.2.3) that yields the generalized median graph (in Section 3.4). Finally, in Section 3.5 we show extensive experimental results and application of median graphs to two interesting problems – the first from biological image analysis and the second arising in drug design tasks.

3.1.1 Problem Description

Let a labeled undirected graph G be given as $G = (V, E, f_v, f_e)$, where

- V is the vertex set, E is the edge set
- L_V is the set of node labels, L_E is the set of edge labels,
- $f_v : V \rightarrow L_V$ is a mapping from nodes to labels or weights, and
- $f_e : E \rightarrow L_E$ is a mapping from edges to labels or weights.

Let $\mathbb{G} = (G_1, G_2, \ldots, G_n)$ be a collection of graphs (in arbitrary orientation) with the following properties:

1. $\forall G_i = (V_i, E_i, f_v, f_e)$, $V_i \subseteq V$ and $E_i \subseteq E$.
2. No restriction on the uniqueness of the vertex labels of V_i, i.e., $f_v(u_i) = f_v(v_i)$, $u_i, v_i \in V_i$ is permissible (and likely).

3. No restriction on the cardinality of the graphs in $\mathbb{G}$, i.e., $|G_i| \neq |G_j|$, $G_i, G_j \in \mathbb{G}$ is permissible (and likely).

The median graph for $\mathbb{G} = \{G_1, \ldots, G_n\}$ must minimize the sum of distances as follows.

$$\bar{G} = \arg\min_{\hat{G}} \sum_{i=1}^{n} d(\hat{G}, G_i) \quad G_i \in \mathbb{G}, \tag{3.1}$$

where $d(\cdot, \cdot)$ is an appropriately defined 'distance' function. In the simplest case, $d(\cdot, \cdot)$ can be the cost of the fewest edit operations required to 'convert' one graph to the other. Alternative definitions of distance may reflect a *similarity measure* between a pair of graphs, as we will see shortly. When $\bar{G} \in \mathbb{G}$, the median graph is the set median; if we waive this requirement, we get the generalized median graph problem.

3.1.2 Previous works

The idea of median graphs were recently introduced by Jiang, Münger, and Bunke [52]. They proposed an approach for generating the median graph from a set of graphs by simultaneously finding (1) the generalized median graph and (2) the optimal mapping between the to-be-computed median and each graph in the input set. The actual optimization, however, was genetic search based. A subsequent paper by Hlaoui and Wang [45] focused on building the median graph through a two step process. First, they found a set of likely nodes for the median. This was done using a clustering algorithm on the node sets of all input graphs. The input graphs were matched (by node edits alone) to the node set of the median at step i, and this process was iterated to determine a good choice for the median. In the second stage, they assigned labels to edges based on the matching results from the first stage. Their method relied on certain hypotheses where local choices that improved the objective function were picked greedily. For the special case when the graphs are certain types of trees, some nice results are known [79].

Observe that to be able to build a median for a set of graphs, it is necessary to be able to quantitatively evaluate the similarities between a subset of graphs. When we consider only two graphs, we have the graph matching or graph isomorphism problem[1]. Unlike generalized median graphs which is still relatively new, the graph isomorphism problem, has been extensively studied by the optimization, mathematics,

[1] a graph isomorphism from a graph G to a graph G' is a bijective mapping from the nodes of G to the nodes of G' that preserves all labels and the structure of the edges. A subgraph isomorphism applies from subgraphs of G to subgraphs of G'

and computer science communities. The earliest papers on this topic are due to Corneil and Gotlieb [31] in 1970 and Ullmann [98] in 1976. The status of the problem is interesting – no polynomial time algorithms are known, at the same time, it is also not known to be **NP**-hard. However, a number of approaches exist for solving various special cases of this problem, for example see [12]. We will avoid an exhaustive discussion (we refer to [95] for details) but discuss only on a subset of algorithms that have used mathematical programming frameworks to do graph matching.

The linear programming (LP, for short) formulation by Almohamad and Duffuaa in [2] uses adjacency matrices for weighted graph matching. This approach nicely exploits the relationship between permutation matrices[2] and graph isomorphism as follows.

$$\min \|A_{G_0} - PA_{G_1}P^T\|, \tag{3.2}$$

where A_{G_0} and A_{G_1} denote the adjacency matrices of the two edge weighted graphs and P denotes a permutation matrix. Hence, the problem reduces to finding one to one correspondences between the vertex sets of G_1 and G_2 such the graphs become 'close' to each other. As (3.2) shows, this is equivalent to finding a permutation matrix, P that minimizes the difference of one graph (say, A_{G_0}) with the permuted version of the other (say, $P(A_{G_1})$). One can immediately see that this problem is closely related to the Quadratic Assignment Problem (QAP).

The algorithm in [2], while being the first of its kind, has a few limitations. First, this algorithm cannot be directly applied to the isomorphism problem on more general pairs of graphs where the number of vertices in two graphs are different. Also, it assumes that the graphs are not vertex labeled. Extending the algorithm to the more general case of *vertex labeled* graphs with *weighted* edges is challenging. To see this, consider the factors contributing to the edit cost in general graphs. Normally, 'edits' are performed on nodes and edges and can be broadly classified as insertion and deletion of nodes ($f_v(u)$), edges ($f_e(e)$) and substitution of nodes ($d_v(f_v(u_1), f_v(u_2))$) and edges ($d_e(f_e(e_1), f_e(e_2))$). The problem of generalized graph isomorphism becomes rather ill-posed for the case where the costs (for nodes *and* edges) are not defined in the same metric space. To motivate this argument, let us consider an illustrative example. In Fig. 3.1, we seek to match graphs G_0 and G_1 in a minimal edit cost sense. Assuming vertex substitution has unit cost (eg., Hamming distance) and edge replacement costs are in L_1 or L_2 space, the matching will favor mapping $\triangle abc$ in

[2]A permutation matrix is obtained by permuting the rows of an $n \times n$ identity matrix according to a permutation of the numbers 1 to n. Each row/column has precisely single 1 and every permutation maps to a unique permutation matrix.

G_0 to $\triangle def$ in G_1 instead of $\triangle abc$. Clearly, the matching is a trade-off between competing influences.

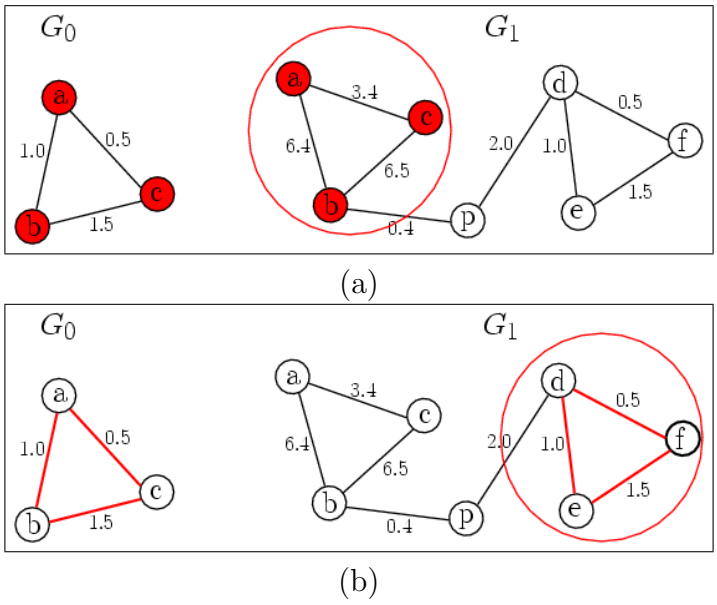


Figure 3.1: Matching with labels and weighted edges.

For the special case of graphs with labeled vertices and unweighted edges, Justice and Hero [53] very recently proposed a modification of the algorithm in [2] to define the Graph Edit Distance in terms of an Integer Linear Programming formulation. This technique allows G_0 and G_1 to have unequal number of labeled vertices as follows. An edit grid is constructed, given as a complete graph, $G_\Omega = (V_\Omega, E_\Omega)$ where $|V_\Omega| = N \leq |V_1| + |V_2|$. The vertex labels belong to an alphabet, Σ. First, the nodes (and edges) on the edit grid are initialized to ϕ (and 0). Then, the initial graph G_0 is placed on this edit grid. The ϕ-labeled vertices of the edit grid take the labels of the vertices of G_0 that are aligned with them. All edges in G_Ω (labeled 0) are converted to 1 if they denote a real edge in G_0. An edit operation on this *standard placement* of G_0 is then defined as either changing the label of a vertex from a character in Σ to ϕ denoting deletion or from ϕ to a character in Σ denoting insertion. Edit operations on edges are denoted likewise using $0 \rightarrow 1$ or $1 \rightarrow 0$ transitions. This amounts to permuting the standard placement of G_0 to minimize the following objective.

$$\min \sum_{i=1}^{N} \sum_{j=1}^{N} d(f_v(A_0^i), f_v(A_1^j)) P^{ij} + \frac{1}{2} \|A_0 - P A_1 P^T\|, \tag{3.3}$$

where A_0 and A_1 are the adjacency matrices of the standard placements of G_0 and G_1 on the edit grid and A_0^i (and A_1^j) is the ith (and jth) vertex of A_0 (and A_1) and $d \in \{0, 1\}$ is the cost function for the vertices.

While vertex and edge edit costs are in the same space (binary), (3.3) does not adequately capture the notion of *similarity distance*. To see this, consider two to-be-matched input graphs with vertices having large degree (≥ 3), as shown in Fig. 3.2 (nodes c and x). If vertices with mismatched labels from the two graphs are aligned, one naturally pays a cost of 1 for a single vertex pair mismatch, see $x \rightarrow c$ in Fig. 3.2 (second column). This is clearly small compared to the cost incurred when vertices having a large difference in degree are matched to each other, see $c \rightarrow c$ in Fig. 3.2 (first column). A natural tendency of the algorithm, therefore, would be to match vertices with the *same degree*, instead of vertices with the *same label*. In fact, in pathological cases labels may be completely marginalized in favor of same-degree-vertex alignments, especially if the vertices have high degree. While this is in accordance with the edit cost definitions, but in most applications, the semantic meaning of a vertex in a graph is as important as its structural relationship with other nodes. For example, consider matching a 'C' (carbon) to a 'H' (hydrogen) simply because they share bonds with the same number (degree) of atoms.

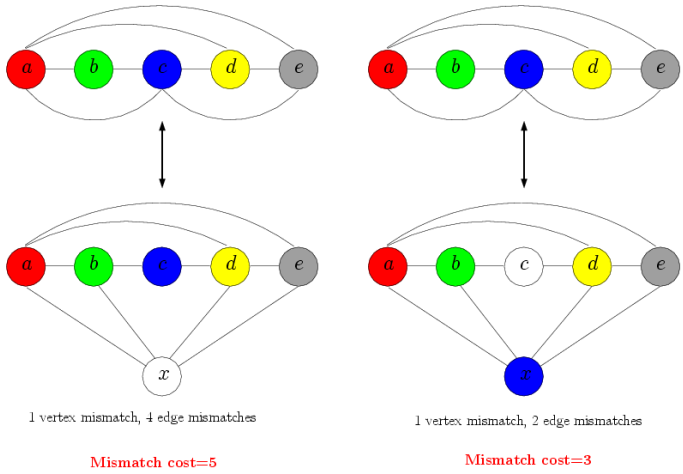


Figure 3.2: Two possible matchings where characters in the nodes are the *labels* and the node colors denote the alignment determined by the algorithm. The cost for vertex *and* edge mismatch is 1. In the left figure, the matching seems reasonable noticing that the vertices with the same labels are well aligned. The mismatch cost is 5; the right matching ignores the vertex labels and matches vertices with the same degree to get a lower mismatch cost.

The above discussion suggests that using edit distance as a cost function in many applications yields a biased weighted matching where a degree mismatch has a higher penalty. A generalized distance function may be more suitable that considers the cost of replacing vertices to reflect the vertex labels and the associated edge information

(structure) concurrently. We will discuss these issues in the next section.

3.2 Main Ideas

3.2.1 A suitable cost function

To address the ill-posedness problem outlined in the previous section, we adopt the following strategy – we will match the edges explicitly, however, since edges are identified by their end vertices, this process will also match vertices implicitly. Assume two to-be-matched vertex labeled graphs, $G_0 = (V_0, E_0, f_v, f_e)$ and $G_1 = (V_1, E_1, f_v, f_e)$. The dissimilarity between the vertex labels of these two graphs is encoded in a matrix S, $S(i,j) = 0$ if $i = j$ and $S(i,j) = 1$ otherwise. The presence or absence of a link between vertices is indicated by their edge weights, f_e where $f_e \in \{0,1\}$. Therefore every edge $e = (u,v)$ has three components, the two end vertices and the link f_e. Consider a match between edge $e_i = (a,b) \in G_0$ and $e_j = (c,d) \in G_1$ i.e., $a \rightarrow c, b \rightarrow d$. The cost that must be paid for such an 'edge' matching is given by

$$cost(e_i, e_j) = S(a,c) + S(b,d) + \|f_e(e_i) - f_e(e_j)\|, \tag{3.4}$$

where the $cost(e_i, e_j) = 0$ if both end vertices of the edge pair match perfectly. Otherwise, it reflects the sum of the costs of

- misaligned vertices (based on the dissimilarity of aligned vertex labels) and
- difference of weights of the edges, which is 0 unless it is matched to a null (i.e., absent) edge.

We must now define *what* needs to be optimized. Suppose a vertex $u_1 \in G_0$ is aligned with $u_2 \in G_1$ such that $S(u_1, u_2) > 0$. Each edge incident on u_1 pays a cost of $S(u_1, u_2)$ due to a misaligned u_1. The total cost of one vertex misalignment pair $(u_1 \leftrightarrow u_2)$ is then

$$D(u_1, u_2) = [\deg(u_1) + \deg(u_2)] \cdot S(u_1, u_2). \tag{3.5}$$

Notice that all edges associated with u_1 and u_2 have paid only part of their cost. The vertex misalignment cost, if any, for the other end vertices of the incident edges of u_1 and u_2, is considered a the time of evaluation of the other pairs of aligned vertices. Considering all edges, the cost function that models this is

$$\texttt{min} \sum_{i=1}^{N} \sum_{j=1}^{N} D(i,j) P^{ij} + \frac{1}{2} \|A_0 - P A_1 P^T\|. \tag{3.6}$$

The second term in (3.6) evaluates the cost of matching a 'real' edge to a null edge (see third term in (3.4)). However, (3.6) as a measure of graph similarity distance is still not entirely accurate. The calculation is perfect when an edge matches with a null edge; however, when an edge matches to another real edge, the dissimilarity, $S(u_1, u_2)$, is counted twice (for $u_1 \in G_0$ *and* for $u_2 \in G_1$). This overestimation must be deducted to reflect the actual cost. We will discuss this shortly.

Before we proceed any further, we will first generalize the notion of graph similarity distance (in (3.6)) in the context of median graphs and then address the overestimation issues in the generalized setup. If there are only two graphs, the cost calculated is the similarity of one graph to the *permuted version* of the other. In our formulation, all input graphs are embedded (placed) in an edit grid, G_Ω, $|G_\Omega| \geq N$. A permutation applied on an input graph is simply a change of placement from one position to the other. If all the input graphs are isomorphic, then it should be possible to find the same placement for all graphs, which is also the generalized median of the input set. The distance of the input graphs to the median in this case is zero. Extending this logic to the general case where input graphs are not necessarily isomorphic, our objective is to find a set of permutations – one for each graph such that the resultant set of placements are the same (or as close as possible). It can be verified that the *mean* of all such close placements will yield the generalized median for the set of input graphs. Because which permutation must be applied to a certain graph is not known in advance, we must simultaneously permute all pairs of graphs, and calculate the cost incurred. In the next section, we will introduce the integer program that models this intuition. We will then address the overestimation in (3.6). Throughout the remainder of this chapter, upper case letters (A) and upper case letters with a single subscript (A_0) will denote matrices; upper case letters with two subscripts or two superscripts will refer to individual matrix entries (A_{ij}, A_0^{ij}, and $A(i,j)$).

3.2.2 Model I: Cost when both graphs are permuted

The edit grid, G_Ω, provides a common 'reference' frame in which the the sum of variations (cost) between the permuted graphs can be defined and computed precisely. Let us first consider the cost definition when two graphs are simultaneously permuted onto G_Ω and then extend the definition for multiple graphs. If graphs, G_0 and G_1, are permuted by P_0 and P_1 respectively, (3.6) is represented as follows.

$$\sum_{i=1}^{N}\sum_{k=1}^{N}\sum_{j=1}^{N} D(i,j)P_0^{ik}P_1^{kj} + \frac{1}{2}\|P_0A_0P_0^T - P_1^TA_1P_1\|, \tag{3.7}$$

where $P_0^{ik} = 1$ indicates that $v_i \in G_0$ is permuted to position k on the edit grid. A triplet, (i, j, k), such that $P_0^{ik} P_1^{kj} = 1$ implies that $v_i \in G_0$ is permuted to the position k, and position k on the edit grid maps to $v_j \in G_1$ and so we must incur a cost, $D(i, j)$. The main difficulty in (3.7) involves the product term of two variable matrices, P_0 and P_1. Our linearization involves an additional variable $X \in \Re^{N \times N \times N}$, X_{ijk} is 1 if $P_0^{ik} P_1^{kj} = 1$ and 0 otherwise. This naturally yields

$$P_0^{ik} + P_1^{kj} = 2 \implies X_{ijk} = 1, \tag{3.8}$$

where " $\implies$ " denotes *implies that*. (3.8) can be converted to regular linear constraints as

$$P_0^{ik} + P_1^{kj} \geq 2X_{ijk}, \quad P_0^{ik} + P_1^{kj} - 1 \leq X_{ijk}. \tag{3.9}$$

Observe that when $P_0^{ik} + P_1^{kj} = 2$, X_{ijk} must be 1 to simultaneously satisfy both constraints in (3.9); if $P_0^{ik} + P_1^{jk} \leq 1$, X_{ijk} must be 0. (3.8) and (3.9) are hence equivalent. Incorporating these into (3.7) gives

$$\sum_{i=1}^{N} \sum_{j=1}^{N} D(i, j) (\sum_{k=1}^{N} X_{ijk}) + \frac{1}{2} \| P_0 A_0 P_0^T - P_1^T A_1 P_1 \|. \tag{3.10}$$

We now revisit the overestimation issues – to calculate the overestimation in (3.10), we first need to determine the edge-to-edge mappings between G_0 and G_1. This is given by the "1" entries in the dot product of $P_0 A_0 P_0^T$ and $P_1^T A_1 P_1$. The overestimation, say $C \in \Re^{N \times N}$, is associated with (i, j) positions with 1s and must be deducted. For presentation purposes, let us define the following notations

$$\gamma = (P_0 A_0 P_0^T + P_1^T A_1 P_1), \tag{3.11}$$

$$\bar{\gamma} = (P_0 A_0 P_0^T - P_1^T A_1 P_1), \tag{3.12}$$

$$\phi(S, X)_i = \sum_{i_1=1}^{N} \sum_{j_1=1}^{N} S(i_1, j_1) X_{i_1 j_1 i}, \tag{3.13}$$

Then,

$$C_{ij} = \begin{cases} \phi(S, X)_i + \phi(S, X)_j & \text{if } \gamma_{ij} = 2; \\ 0 & \text{if } \gamma_{ij} \leq 1 \end{cases} \tag{3.14}$$

Notice that for an edge, e_{ij}, when the sum in (3.14) is 2, i.e., when the the dot product of $P_0 A_0 P_0^T$ and $P_1^T A_1 P_1$ at position (i, j) is 1, C_{ij} is non-zero (i.e., equal to the overcharged cost of matching the end vertices of e_{ij}). Here, i and j denote the

indices of vertices of the individual graphs *after* permutation, these can be mapped back to the original indices of those vertices in G_0 and G_1 using X. Let M be the maximum mismatch cost of an edge (computed offline, see (3.4)). The conditional constraints in (3.14) can then be easily expressed as

$$\begin{aligned} \phi(S,X)_i + \phi(S,X)_j - C_{ij} \leq (2-\gamma_{ij})M, \\ C \leq MP_0A_0P_0^T, \quad C \leq MP_1^TA_1P_1. \end{aligned} \tag{3.15}$$

Here, the constraints (denoted without subscripts) apply for all matrix elements. Therefore, the final objective function that takes care of the overestimation is

$$\min \sum_{i=1}^{N}\sum_{j=1}^{N} D(i,j)(\sum_{k=1}^{N} X_{ijk}) + \frac{1}{2}\|\bar{\gamma}\| - \underbrace{\frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} C_{ij}}_{\text{overestimation}}, \tag{3.16}$$

where $\bar{\gamma}$ is as defined in (3.12) and the constraints are

$$P_0e = e; P_0^Te = e; P_1e = e; P_1^Te = e, \tag{3.17}$$

$$P_0^{ik} + P_1^{kj} \geq 2X_{ijk}, \quad P_0^{ik} + P_1^{kj} - 1 \leq X_{ijk}, \tag{3.18}$$

$$\phi(S,X)_i + \phi(S,X)_j - C_{ij} \leq (2-\gamma_{ij})M, \tag{3.19}$$

$$C \leq MP_0A_0P_0^T, \quad C \leq MP_1^TA_1P_1, \tag{3.20}$$

$$P_0, P_1, X \in \{0,1\}; C \in [0,M]. \tag{3.21}$$

When only one graph is permuted, we have a simpler model with only $O(n^2)$ variables and constraints. We discuss this in the next section.

3.2.3 Model II: Cost function when only one graph is permuted

In this section, we consider the simplified case of Section 2.2, i.e., when only two graphs are given. Then, we can keep the target graph fixed (which serves as an edit grid) and only the source graph is permuted; we no longer need to linearize a product term. This is also the same as finding the isomorphic mapping of one graph with respect to the other and can be used for matching two graphs. The number of variables and constraints required are only $O(n^2)$. For convenience, let

$$\psi_i(P) = \sum_{l=1}^{N} S(l,i)P^{il} \tag{3.22}$$

we thus have the following simplified Integer Program,

$$\min \quad \sum_{i=1}^{N}\sum_{j=1}^{N} D(i,j)P^{ij} + \frac{1}{2}\underbrace{(S+T)}_{\text{slack variables}} - \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} C_{ij} \tag{3.23}$$

$$\begin{aligned} s.t. \quad & Pe = e; P^T e = e, \\ & A_1 P - P A_0 + S - T = 0, \\ & \psi_i(P) + \psi_j(P) - C_{ij} \leq (2 - (PA_0 + A_1 P)_{ij})M, \\ & C \leq M A_1 P; C \leq M P A_0, \\ & S, T, P, X \in \{0,1\}; C \in [0, M]. \end{aligned} \tag{3.24}$$

The relaxation of this Integer Program can be solved optimally in polynomial time. We can then round the solution to obtain integral solution values. We discuss the rounding in Section 3.4.1.

3.3 Linearization and Relaxation

In this section, we describe our relaxation technique. We illustrate the ideas using the familiar objective function in (3.2) in Section 3.1.2. The strategy applies readily to (3.16)-(3.21). We observe that the objective, $\sum_{ij}(A_{G_1} - PA_{G_2}P^T)_{ij}$, is equivalent to

$$\sum_{l}(\texttt{vec}(A_{G_1}) - \texttt{vec}(A_{G_2})(P \otimes P))_l, \tag{3.25}$$

where $\otimes$ indicates the Kronecker product and $\texttt{vec}(\cdot)$ vectorizes a matrix by stacking its columns. Consider a matrix variable Q where $Q = P \otimes P$, the nonlinearity in the second term can be addressed[3] as follows.

$$\sum_{l}(\texttt{vec}(A_{G_1}) - \texttt{vec}(A_{G_2})Q)_l, \tag{3.26}$$

where $Q \in \Pi_{n^2}$ and Π_d denotes permutation matrices of size $d \times d$. Q has some additional interesting properties; specifically, the (i,j)th *block* of Q, denoted as $Q_{[ij]}$ corresponds to element P^{ij}. Below, we use $Q(i,j)$ to denote the (i,j)th *entry* of Q

[3]The relaxation in [2] works perfectly when only one graph is permuted, (3.2) is used only as an example.

and S, T are slack variables. The resultant LP is

$$\begin{aligned}
\min \quad & \sum(S+T) && (3.27)\\
s.t. \quad & \texttt{vec}(A_{G_1}) - \texttt{vec}(A_{G_2})Q + S - T = 0 \\
& \sum_{l=1}^{n} Q(k+(i-1)n, (j-1)n+l) = P^{ij},\ \forall i,j,k \in [1,n], \\
& \sum_{i=1}^{n} Q_{[ij]} = P, \sum_{j=1}^{n} Q_{[ij]} = P,\ \forall i,j \in [1,n], \\
& Pe = e; e^T P = e^T; P, Q, S, T \geq 0. && (3.28)
\end{aligned}$$

The relaxation of (3.27)-(3.28) is polynomial time solvable.

3.4 Generalized Median Graphs

A natural extension of (3.16)-(3.21) yields a model for computing the generalized median graph for a set. Let the 'distance' between two graphs as defined in (3.16) be given as $d_p(G_0, G_1, P_0, P_1)$. Notice that while $d_p(\cdot)$ simply computes the cost given four parameters (no unknowns), (3.16) tries to optimize the distance where the permutations are unknown. Using the notation above, our objective is to permute all graphs onto the edit grid and can be expressed as

$$\texttt{min} \sum_{x=1}^{K}\sum_{y=1}^{K} d_p(G_x, G_y, P_x, P_y), \tag{3.29}$$

where K is the number of graphs in the set. The 0-1 solution can be obtained by rounding the fractional entries of the K permutation matrices (discussed in detail in Section 3.4.1) and obtaining the median graph as a mean of the permuted input graphs. This yields a polynomial time algorithm for computing the generalized median graph, no bounded running time algorithms were known before.

Our experimental results using this model are discussed in Section 3.5. In general, this approach can be used for reasonably sized graphs (no assumptions on the structure) and yields good results in practice. Since the number of variables in the model are $O(N^4K)$, for large graphs some application specific heuristics may need to be employed. In the following sections, employing the ideas above and additional observations we discuss a hybrid approach that yields a factor two approximation ratio if the graph distances are accurately known. Finally, we propose another bi-level algorithm that finds a solution arbitrarily close to the optimal (though running time is non-polynomial in worst case).

We employ the concept of generalized median for a set of objects in metric spaces [51] as a first step. Let the distance (discussed above) between two graphs be given as $d(G_0, G_1) = d_p(G_0, G_1, I, P_1)$. We first introduce the following result.

Lemma 3 (Metric) *For any three graphs G_p, G_q and G_r, $d(G_p, G_q) \leq d(G_p, G_r) + d(G_q, G_r)$ i.e., $d(\cdot, \cdot)$ is metric.*

Let $G = (G_0, G_1, \ldots, G_K)$ be a collection of graphs with $|V_i| = |V_j|, \forall i, j$. Dummy nodes can be introduced if $|V_i| \neq |V_j|$, so $N = \max(|V_1|, \ldots, |V_n|)$. Let G^* be the optimal median of this set. By definition, G^* satisfies

$$G^* = \arg\min_{\bar{G}} \sum_{i=1}^{K} d(\bar{G}, G_i)$$

We will avoid computing G^* directly. Instead, we employ the lower bounding ideas in [51] to model the distance of every graph, G_i to the generalized median by an unknown variable, x_i. Therefore, $x_i = d(G^*, G_i)$. Then, the problem can be modeled as follows.

$$\begin{aligned} \min \quad & \sum_{i=1}^{K} x_k && (3.30) \\ s.t. \quad & x_k + d(G_k, G_l) \geq x_l, \forall k, l, \\ & x_l + d(G_k, G_l) \geq x_k, \forall k, l, \\ & x_k + x_l \geq d(G_k, G_l), \forall k, l, \\ & x_i \geq 0, \forall i \in [1, K]. && (3.31) \end{aligned}$$

(3.30)-(3.31) can be solved optimally in polynomial time.

Property 1 *(from [51]) The sum of the entries of $\mathbf{x} = (x_1, \ldots, x_K)^T$ denotes a lower bound on the distance of all graphs to the optimal generalized median.*

Lemma 4 *There must be a graph $G_a \in G$ that satisfies*

$$\sum_{i=1}^{K} d(G_k, G_a) \leq 2 \sum_{i=1}^{K} d(G_k, G^*),$$

where G^* is the optimal median graph for G.

Proof: Let $\mathbf{x}^* = (x_1^*, \ldots, x_K^*)^T$ be an optimal solution for (3.30)-(3.31). Consider a graph $G_a \in G$ s.t. $a = \{j : x_j^* \leq x_i^*, \forall i \in [1, K]\}$. From (3.31), we know $\forall G_i, d(G_i, G_a) \leq x_i^* + x_a^*$. Since $x_a \leq x_i^*$, $d(G_i, G_a) \leq 2x_i^*$. Therefore $\sum_{i=1}^{K} d(G_i, G_a) \leq 2 \sum_{i=1}^{K} x_i^*$. Applying Property 1, the lemma follows. □

Theorem 3 *The graph $G_a \in G$, $a = \{j : x_j^* \leq x_i^*, \forall i \in [1, K]\}$, is a 2-approximation for the generalized median graph problem if $d(\cdot, \cdot)$ is known correctly.*

If $d(\cdot, \cdot)$ is known correctly, the factor two approximation (for the Hybrid algorithm) holds by comparing the solutions of (3.30)-(3.31) and (3.16)-(3.21) and outputting the better solution.

3.4.1 Rounding

We would like to convert the fractional solution of P into a 0-1 permutation matrix which is 'close' to P. This is done as follows. We construct a bipartite graph, $G = (V, E)$ with subgraphs G_1 and G_2, where the indices of rows in P denote vertices of G_1 and indices of columns in P are given by the vertices of G_2. Since P is of size $n \times n$, $|V| = 2n$. The edge set, E, has exactly $\frac{n^2}{2}$ members because we join every vertex in G_1 to every vertex in G_2. The weight of an edge, $w(e)$, between $v_i \in G_1$ and $v_j \in G_2$ is equal to the entry $P(i, j)$. The optimal solution to the bipartite matching problem can be obtained efficiently. It is easy to prove that this yields the closest permutation matrix to P. The above strategy can also be directly adopted for the generalized median graph case with multiple graphs.

3.4.2 Alternate bi-level algorithm (A sketch)

In this section, we briefly outline our alternate bi-level algorithm to obtain even better solutions. Let $U = \sum_{i=1}^{K} d(G_k, G_a)$. We know that the optimal solution lies in $[\sum_i x_i^*, U]$ and any feasible solution in this range is a factor two approximation. We adopt a binary search in $[\sum_i x_i^*, U]$ starting at a value $c = \sum_i x_i^*$ and solving a feasibility problem (FP) at each step. The objective function of the FP is max 0 subject to the constraints that the sum of distances of $G_i \in G$ to an unknown graph $\hat{G}$ is upper bounded by c. Here the adjacency matrices of $\hat{G}$ and P_i are variables. The constraints are defined by (3.16)-(3.21). If this returns a feasible solution, we are done (i.e., $\hat{G} = G^*$). Otherwise, we continue the binary search. The FP can be solved using branch-and-bound type methods (available in most commercial solvers) but may have non-polynomial running time in the worst case.

3.5 Experimental Results

We present our experimental evaluations in three parts. In the first subsection, we evaluate the generalized cost model on a publicly available graph database for isomor-

phism testings. We follow this with evaluations of median graph determination given a collection of graphs. In the next subsection, we discuss experiments in context of a median graph problem in biological image analysis. Finally, we discuss an application of median graphs to drug design (pharmacophores).

The numerical results are based on an implementation of the algorithm in C++ using CPLEX as the linear program solver.

3.5.1 Database Evaluations

From the *Graph Database* [39] (see `http://amalfi.dis.unina.it/graph/`), we selected about 900 labeled (both vertices and edges have labels) graphs (files with the prefix *r005*). All combinations of the parameters, *size* $= \{20, 25, 40\}$, and *isomorphism percentage(mcs)* $= \{90\%, 70\%, 50\%\}$, were considered. For every unique combination, all isomorphism pairs were evaluated and the mean of the errors analyzed. We considered the vertex attributes alone which were normalized to a smaller integer numeric scale. The dissimilarity matrix S of vertex labels was created as follows. Let G_0 and G_1 be two input graphs and let d_{G_l,L_i} denote the degree of vertex v_i with label L_i in graph G_l. Then the dissimilarity between vertices $v_i \in G_0$ and $v_j \in G_1$ is given as

$$S_{ij} = \begin{cases} 0 & \text{if } L_i = L_j \\ \max(\|L_i - L_j\|, d_{G_0,L_j} + d_{G_1,L_i}) & \text{if } L_i \neq L_j \end{cases}$$

The second case above appropriately penalizes matching of vertices with different labels (in general, S depends on the application).

The average percentage of correct mappings for each of nine combinations is shown in Fig. 3.3. When the graphs are close to isomorphic (e.g., $mcs90$) albeit with arbitrary orientations and embedding, our algorithm does quite well ($\sim 90\%$) in finding the permutations that matches similar labels together while maximally preserving the graph structure at the same time. In the extreme case ($mcs50$), when only 50% of the graphs are isomorphic, our algorithm still performs well and achieves about 83% accuracy on an average.

We now discuss the experimental evaluations of our generalized median graph algorithm. The idea is to generate a base graph and then create a new set of graphs from the base by randomly adding noise. The median determined can then be evaluated against the base graph. The base graph was generated as in [39] and served as the truth (known median). For the set of input graphs, error was added (vertices/edges added and deleted) in the base graph. The probability of error in an edge (and error

Size	Percentage of Isomorphism		
	$mcs90$	$mcs70$	$mcs50$
20	93.62%	87.18%	87.26%
25	92.32%	82.96%	84.67%
40	86.48%	81.87%	81.25%

Figure 3.3: Table of Isomorphism evaluations on Graph Database.

magnitude) connecting two vertices of a graph was considered independent of the vertices. The computed generalized median graph was compared to the base graph, the mean of the errors (w.r.t. similarity distance) normalized by the size of the base graph was then analyzed. Here, $N = 6$, $K \in [4, 10]$ and 30% error was introduced (in Figs. 3.4(a), 3.4(c)) in the vertices and edges. In Figs. 3.4(b) and 3.4(d), the percentage of error introduced was varied. In Fig. 3.4(a), we evaluate the performance as a function of the number of graphs in the input set. The y-axis in the plot shows the normalized similarity distances of every graph to the base graph (under best permutation) calculated as above. Observe that the average characterizes the amount of 'noise' in the input set when the best possible permutations are applied on the graphs. We can see that the algorithm performs well and consistently finds a generalized median that is close to the base graph. The deterioration in performance with a larger set of graphs (around 10) is only marginal. We can see the same trend when the errors introduced in the set are increased to up to 50% in Figure 3.4(b). In Figures 3.4(c) and 3.4(d), we repeat the same evaluations as above; however, we compare the performance of our algorithm with the lower bound. This gives us a reference to evaluate the similarity distance of the computed median to the best (not necessarily existent) solution. In both plots, we can see that the solutions are very close to the optimal.

3.5.2 Applications in Biological Image Analysis

Biological background Chromosomes within the human cell nucleus are known

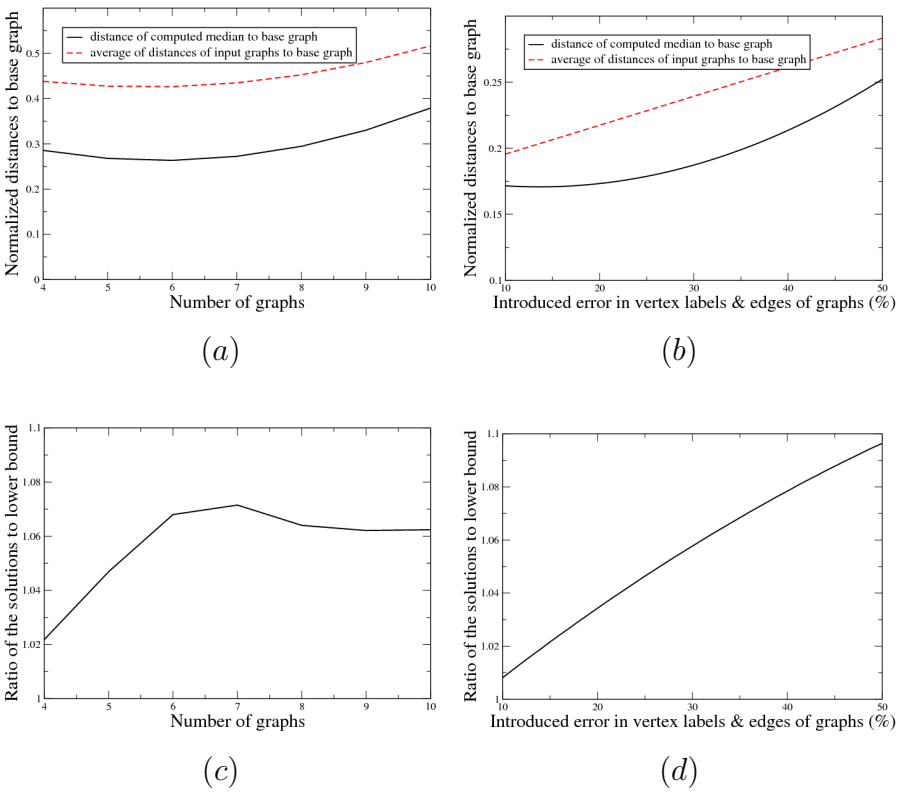


Figure 3.4: Average of the distances of input set graphs and the distance of the computed median to the base graph w.r.t. the number of graphs (with 30% error) in (a) and introduced error in (b). Ratios of the median to the lower bound in (26)-(27) w.r.t the number of graphs in (c) and introduced error in (d).

to occupy discrete territories and evidence from recent literature suggests that these might be functionally relevant because the position of these territories in nuclear space is 'organized' [32]. This organization shows a statistical correlation with gene density and the size of chromosomes [44]. In humans, larger chromosomes (numbers $1-10$) are arranged on the periphery. Secondly, chromosomes may have non-random neighbors [72]. Biologists suspect that such patterns are related to genomic evolution and cell-type-specific variability. Based on this, several researchers [17; 71] have stressed on the need of mapping large-scale organization and distribution of chromatin in various cell lines. The standard approach relies on a sequence of experiments on a large number of cells and then manually investigating the chromosome to chromosome relationships sequentially using the acquired images (see Fig. 3.5, one for every cell). A reliable topological 'model' can provide the necessary foundation for studying the effect of higher-order chromatin distribution on nuclear functions. Unfortunately, an arrangement map for all 23 chromosome pairs in a diploid human cell nucleus is unavailable.

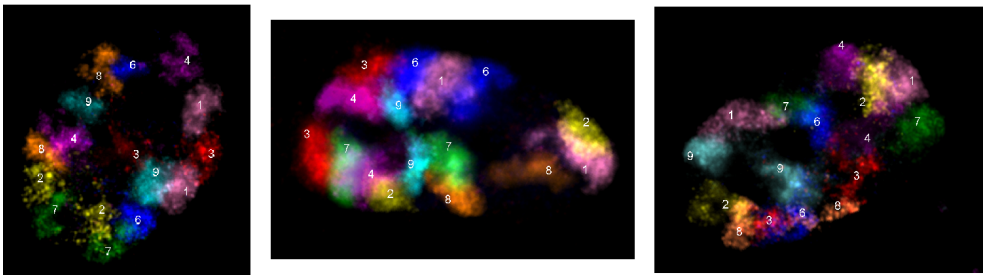


Figure 3.5: Three nuclear images (out of a set of thirty-seven) with eight chromosome paints.

Chromosomal organization maps. Topological map derivation of chromosomes transforms naturally to the generalized median graph problem. Each individual chromosome (denoted as numbers in Fig. 3.5) is represented as a labeled vertex in a graph. Due to microscopic acquisition limitations, at most 8 pairs of chromosomes can be 'labeled' per cell; hence, our graph has 16 vertices with at most 2 vertices with the same vertex label (i.e., chromosome numbering). Spatial proximity (adjacency) between chromosomes is expressed as an edge between vertices. We focus on 8 pairs of the larger chromosomes (numbers $1, 2, 3, 4, 6, 7, 8, 9$) from the *human lung fibroblast cell line*. The acquisition process was repeated for 37 cells overall. The raw images were segmented to create masks for individual chromosome pairs and individual graph representations derived (see Fig. 3.6).

To evaluate the goodness of our final 'prototype' for this cell line we divide the cells into a training set (19 cells) and a test set (18 cells). We report on the following observations.

Optimality. Figure 3.6(g) shows the median graph obtained from the training set. The objective function value was 1.43 times the lowest possible cost for computing the median for this set of graphs (lower bound by (3.30)-(3.31)). This ratio for the {training, test, entire} set medians were {1.58,1.6,1.63} indicating that the computed generalized median is a better representative for the set than any of the component graphs. Note: Exhaustive enumeration analysis suggests that a solution much closer to the lower bound is unlikely.

Substructure frequency. We generated groups of chromosomes which occur frequently as a chain or a sub-group (e.g., numbers (b, c, d)) by graph mining [1; 104] and other available algorithms [67]. We then compared these to the results obtained by the median graph model for this cell line. Figure 3.6(h) shows groups (colored edges) with an occurrence frequency of $\geq 70\%$. Fig. 3.6(h) shows that almost all such sub-graphs align *perfectly* with the computed median graph indicating that the

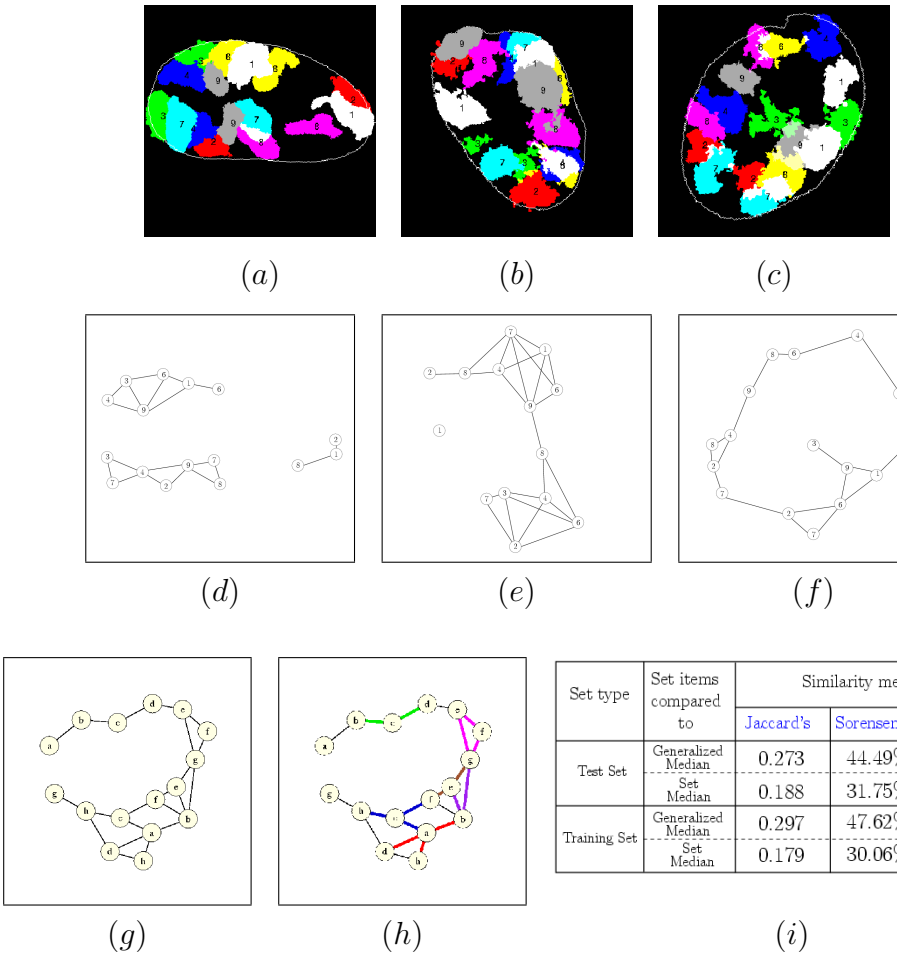
model incorporates the essential information from the constituent graphs.

Structure Prediction. The basic question we wanted to answer was – given a new cell, can we predict the non-random part of its chromosomal organization? If yes, then the model serves its purpose and can be used to interpret biologically relevant information. To do this we calculated Jaccard's, Sorensen's, and Mountford's similarity of the median (from training set) with each cell in the training/test set (considering attributed edges as set members). The results in Fig. 3.6(i) show that the similarity is only slightly worse for the test set wrt the training set. Considering that Sorensen's is a percentage similarity measure, given a new cell, we can predict close to 50% of its chromosome organization. We will omit further discussion on the biological significance of these results.

3.5.3 Application to Drug Design: Pharmacophores

Background. Drug design involves finding drugs molecules that bind with a target molecule in the human body. The target molecule is biologically related to one or more disease conditions; the process of the drug molecule *binding* to the target inhibits the metabolic pathways specific to certain diseases. Therefore, one important aspect of drug design is to identify the target molecule. However, in many scenarios, the chemical composition of the target molecule is not known precisely. To address this problem, researchers typically try a number of drug molecules with varying chemical structures to account for the variations in the target molecule. An additional problem stems from the fact that all molecules that bind to a specific target site (called active molecules) may not be suitable for use in drug design (e.g., toxicity and instability). Therefore, given a number of molecules that bind to a target molecule, the task is to determine their structural similarities, and hence design a 'model structure' called *pharmacophore* such that one or more active molecules share structural similarity with the pharmacophore. In other words, a pharmacophore is a three-dimensional map of biological properties common to a large fraction of molecules that exhibit a particular activity. Hence, they serve as a *design template* in drug design.

Existing techniques for pharmacophore discovery rely on geometric point hashing (when the structures are described as sets of points), inductive logic programming (ILP) [38; 66] or graph mining [63]. Graph mining algorithms can identify commonly occurring substructures in a set of molecules, but inferring the structure of the 'template' molecule by stitching together individual substructures has to rely on expert knowledge. ILP, on the other hand is more general but computationally intensive – to learn full clausal theories (or rules) based on given background (or axioms), a



(a) (b) (c)

(d) (e) (f)

(g) (h)

Set type	Set items compared to	Similarity measures		
		Jaccard's	Sorensen's	Mountford's
Test Set	Generalized Median	0.273	44.49%	3.23
	Set Median	0.188	31.75%	2.29
Training Set	Generalized Median	0.297	47.62%	3.7
	Set Median	0.179	30.06%	2.3

(i)

Figure 3.6: $2D$ sections of chromosome organization in 3 cell-images in (a)-(c) and their graphical representation in (d)-(f); a generalized median graph for a set of 19 cells in (g); all patterns (except one) with frequency of $\geq 70\%$ across 19 cells (by graph-mining) were present in the generalized median graph and shown as colored in (h); average of similarity measures of items in the test/training sets wrt to the generalized median and set median of those sets in (i).

Prolog-type interpreter is used to carry out theorem proving. The design of practical and efficient theorem provers is still an active area of research, see [66].

Median Graphs in pharmacophore design. We observe that median graph formulation provides a natural framework not only for embedding the frequently occurring substructures, but can also be useful in designing the largest possible pharmacophore for a set of active compounds. To illustrate these concepts, we used a set of 12 active compounds from the class of Pyrimidine Nucleosides selected from AIDS Antiviral Screen Database located at National Cancer Institute website at `http://nci.cambridgesoft.com/` (a few are shown in Fig. 3.7). The size of molecules varied from 22 to 45, a single atom was present at most 22 times in a

molecule. Given the large size of the compounds, we used a variation of the algorithm proposed earlier to speed up computation. Instead of assuming that the orientation of the median graph is unknown, we selected one of the input graphs as the base orientation. We then permuted all other input graphs to match with the first graph. The permuted orientation of all graphs was then averaged to generate the median graph. We repeated the process for each input graph – generating a new graph (and corresponding cost) at each iteration. The best graph based on the cost value (see (3.7)) was chosen.

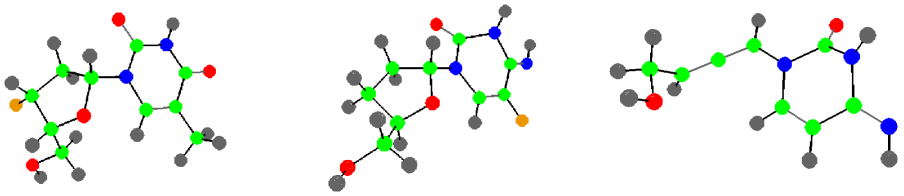


Figure 3.7: Three among 12 chemical structures in the Pyrimidine Nucleosides input set. The structures (left to right) correspond to Thymidine, 3'-deoxy-3'-fluoro- (8CI 9CI), 2',3'-Dideoxy-5-fluorocytidine, and N-[(Dimethylamino)methylene]-5-fluoro-2',3'-dideoxycytidine. Different colors correspond to different atoms as follows: gray for H, green for C, red for O, blue for N, and brownish-yellow for F atoms.

The generalized median graph is shown in Figure 3.8. The objective value for the computed generalized median was slightly better ($\sim 2\%$) than the set median. To evaluate how the frequently occurring substructures fit onto the median graph, we found all frequent subgraphs in the input set (frequency of $\geq 95\%$) by using standard graph mining software [1; 104]. Some of these are shown in Figure 3.9. Note that the graph mining software simply returns edge-groups that appear with high frequency; constructing a 'model' that fits all the frequently occurring substructures is not an easy task. Nonetheless, such frequent substructure information serves as a good verification tool. Since the generalized median graph for the set had already been computed, , it was easy to examine if the frequently occurring edge-groups were present in our model. We can see in Fig. 3.9, that all frequent substructures were also found in the computed median graph.

3.6 Conclusions

This chapter applies mathematical programming to design an optimization framework for the generalized median graph problem in context of two real world biomedical problems – in (a) biological image analysis and (2) structural pattern recognition

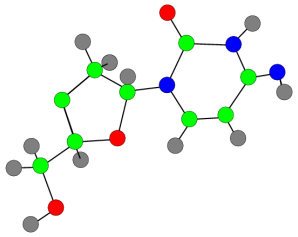


Figure 3.8: The computed median graph for the input set of structures in Pyrimidine Nucleosides.

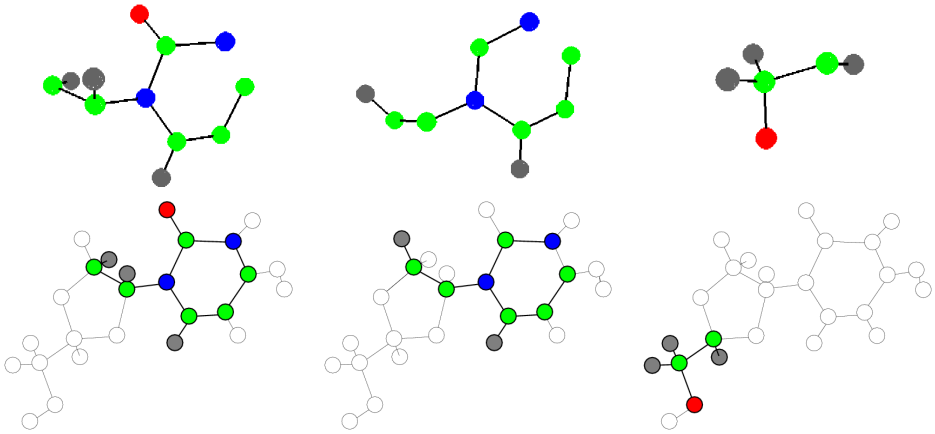


Figure 3.9: Three frequently occurring substructures in the Pyrimidine Nucleosides set (top row) and their placement in the computed median (bottom row).

in drug design. In addition to these problems, we believe that the ideas will find general applicability in building exemplars from representations such as attributed skeletal graphs and aspect graphs. The extension of the similarity metric is also of independent interest.

Chapter 4

Fitting Polygonal Regions for Matching 3D Polyhedra

4.1 Problem Description and Previous Works

Matching geometric objects is a fundamental problem in computational geometry with applications in computer vision [7; 88], computer graphics [54; 99], biology [21; 25], medical imaging [58; 78], and a wide variety of other diverse areas [76]. The problem has been extensively studied and a number of interesting results have been obtained. Unfortunately, there is still no single algorithm that addresses all shape matching problems. Part of the reason is that the problem is not very well defined (with respect to a universally accepted minimization criteria) − the efficiency of an algorithm in a particular context generally relies on the evaluation of its corresponding "distance metric" As can be expected, the semantic meaning of a distance metric varies from one application to the next. For this reason, there is no 'best' algorithm which yields the optimal answer for all shape matching problems. For specific applications, once a suitable algorithm is chosen, its distance metric needs to be 'tuned' to suit the application's specific needs. In general, this is not an easy step. For example, consider a situation where the input shape is under-samples of the original object. General algorithms will often go wrong here − given only a few undersamples of an object, it is difficult to predict its unknown $3D$ features. Reconstruction based on such undersamples is inherently flawed, usually as a function of the amount of information lost due to the undersampling. Given a query undersampled shape, and a pool of likely matches, it will be useful to have a simple "No" or "Maybe" answer that will eliminate a good part of possible candidates for a match. Once the culling process is done, a combination of other techniques could be used in conjunction

with specialized knowledge to determine likely matches. Our algorithm is designed to address this selective elimination step. Next, we will discuss more specific motivations from such diverse areas as cellular biology and medical imaging.

One challenging problem in cellular biology is to determine the dynamics of chromosome territories and chromosomal foci in cell nucleus of living cells through a sequence of microscopy images [69], that is, to determine the behavior of chromosomes and its constituent DNA. The microscopy images of a cell nucleus consist of a set of parallel $2D$ cross sections. However, due to a number of physical limitations (e.g., the resolution of microscopy and current labeling techniques for living cells), the set of cross sections are only under-samples of the very complicated nucleus structures. In a typical living cell, there may be hundreds (or even thousands) of labeled nuclear structures (e.g., replication sites (RS) or transcription sites (TS) [10; 11]). To determine their dynamics, a key problem is to establish the correspondence among those labeled structures in two consecutive image sets. Since each labeled nucleus structure could move independent of the other nuclear structures, the two sets of $2D$ images could be significantly different. Thus, a challenging problem is to determine the correspondence among the labeled nuclear structures in the two input images.

In computer vision literature, a related problem, called *alignment problem*, has been studied [6; 26; 82]. The alignment problem tries to recognize (or dock) a known object (or model) from an image scene by using model-based recognition techniques [75]. In this problem, the model is often known and some reference points as well as their corresponding points in the other image or object can be relatively easily determined. However, in our fitting problem, both polyhedra(s) are unknown and the reference points are often difficult to obtain. Thus, a more general technique is needed to address this problem.

We notice another closely related problem arising out of managing and querying huge volumes of data in PACS (picture archiving and communications systems) and imaging informatics. In Medical Imaging, content based image retrieval techniques (CBIR) are used to assist radiologists (or clinicians) in querying large databases for retrieval of $3D$ images with similar 'contents' as a query image[4; 102]. Similarity is first established by scanning the database for image tags similar to the query image. Once that is done, for each image with similar tags, a distance function [19] is computed. Notice that if the first step of tag scan returns a large number of images as potential matches, evaluating the distance function for this set can be extremely time consuming. If the volume datasets in the image database are segmented in advance, and then sampled and represented in far fewer number of slices, the query time could

be significantly expedited, at least by a few orders of magnitude. For example, if the number of images returned by the tag scan process is large, a partial matching algorithm could be employed on the query image and its potential matches. Once the prospective number of matches is further reduced, the distance functions or any other such measures could be evaluated.

To overcome the aforementioned difficulties in applications, we consider in this chapter the following problem. Given two sets P and Q of $2D$ polygonal regions with each set representing the cross sections (or slices) of an unknown $3D$ polyhedron (see Figures 4.1 and 4.2), design an efficient algorithm to determine whether P and Q are generated from the same $3D$ polyhedron. In each of the two sets P and Q, a cross section c_i may include one or more simple polygons, possibly with holes in the interior of each polygon. The boundaries of these polygons are the common intersections of the cross sections and the boundary of the $3D$ polyhedra, and the set of cross-sections in one set are parallel to each other.

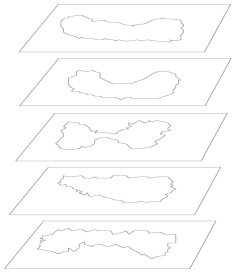


Figure 4.1: An unknown 3D object is represented by a set of horizontal slices, P.

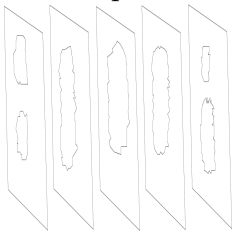


Figure 4.2: An unknown 3D object is represented by a set of vertical slices, Q.

Our fitting problem can be viewed as a special shape matching problem. Unlike the normal shape matching problem in which one determines for each point in one object a corresponding "closest" point in the other, our fitting problem needs to check whether there exists a transformation so that each polygon in P can "fit" with the corresponding polygons in Q. That is, we only need to determine the corresponding points for a subset of points in P and Q. It is a little surprising that up until now the term partial matching has referred only to answering subset questions of the form, "whether a query sample (shape) Q is a part (subset) of another sample (shape) A?". Questions such as, "could Q and A be undersampled representations of

an unknown object B?" still remain unanswered. In this chapter, we present efficient algorithms for answering these questions. Our algorithm is based on interesting geometric techniques and enables answering these queries either as plausible or a certain negative.

4.2 Preliminaries

Before going into the details of our algorithm, we will first discuss a few preliminaries.

Let $P = \{p_1, p_2, \ldots, p_m\}$ and $Q = \{q_1, q_2, \ldots, q_n\}$ be the two sets of cross sections of two (or possibly the same) unknown polyhedra, O_P and O_Q. Each $p_i \in P$ (and $q_j \in Q$) is the i^{th} (and j^{th}) slice in the form of a polygon or polygonal region representing the contours of O_P and O_Q. Let P_i and Q_j be the two supporting planes of p_i and q_j respectively. If P_i is not parallel to Q_j, the common intersection of P_i and Q_j (i.e., $P_i \bigcap Q_j$) forms a straight line l_{ij} which may intersect p_i and q_j and generate two sets of disjoint segments, S_{ij}^i and S_{ij}^j, on p_i and q_j respectively.

We say p_i fits (or ϵ-fits) q_j if there exists a transformation T such that under T, S_{ij}^i exactly matches (or ϵ-matches) S_{ij}^j, i.e., there exists a one-to-one mapping f between S_{ij}^i and S_{ij}^j such that the Hausdorff distance $\delta_H(s, s')$ between $s \in S_{ij}^i$ and $s' = f(s) \in S_{ij}^j$ is 0 (or $\leq \epsilon$). Similarly, we say P fits (or ϵ-fits) Q if there exists a single transformation T which makes each cross section $p_i \in P$ fit (or ϵ-fit) every slice $q_j \in Q$. Thus, to determine whether P and Q are generated from the same polyhedron, we only need to determine whether there exists a transformation T such that under T, P fits Q. It is interesting to notice that if P and Q are only under-samples of the represented $3D$ polyhedra(s), it is possible that P and Q could fit each other even if they are generated from different objects. On the other hand, if P and Q cannot fit each other, we can conclude with certainty that they are generated from different objects. This argument forms the basis of the decision which the algorithm makes on a given input.

Notice that according to the above definition, if both S_{ij}^i and S_{ij}^j are empty, p_i fits q_j. Thus, if for every pair of p_i and q_j, their corresponding S_{ij}^i and S_{ij}^j are empty, P fits Q. We call such a fitting a *non-intersecting fitting*. For example, if the two sets of cross sections interweave each other, no intersection occurs in the interior of either O_P or O_Q. As a result, the corresponding S_{ij}^i and S_{ij}^j will be empty, an example of a non-intersecting fitting. Allowing such non-intersecting fittings could, however, cause a potential problem. Consider two arbitrary sets of cross sections P and Q generated from two different polyhedra. Since we can easily translate them so that no polygon

in P intersects any polygon in Q, P and Q will thus incorrectly 'fit' each other. To address this issue, we associate with P and Q a box B − this box should be large enough to contain all the polygons in P and Q. When determining whether P fits Q, we consider only those transformations which make all polygons of P and Q lie inside B.

4.3 Fitting Algorithms under Translation and Rotation

In this section, we present the main ideas behind our algorithms for fitting P and Q under translation and rotation. We assume that the cross-sections in P are not parallel to those in Q (for parallel cross sections, they will trivially fit each other). To determine whether P and Q can fit (dock) each other, we fix P and move Q to find the best possible translation. We denote the translation of Q by a vector of $t \in R^3$ as $Q + t$. It should be pointed out that we make no assumptions about P and Q having equal number of cross sections. Clearly, if $|P|$ and $|Q|$ vary substantially, each cross section in Q may have significant freedom to move around P.

4.3.1 Observations

In our fitting problem, each $p_i \in P$ (or $q_i \in Q$) could be of arbitrary shape. For the sake of simplicity, let us first assume that each polygon is convex (in general, our algorithm works for any simple polygonal contour; it can also deal with multiple connected components). As mentioned before, for any two given sets of slices, P and Q, they provide only a partial shape information about the object from which they are acquired. Therefore, for a decision problem of whether these two slice sets are from the same object, a 'Yes' answer indicates plausibility. On the other hand, when our algorithm cannot fit the slices of P and Q, it decides a 'No' answer with certainty. It will be helpful to illustrate this concept with the help of the following example.

Suppose P contains 10 slices of an object. Each slice is a square with sides of length, 10 cm, aligned vertically. The gap (resolution or sampling) between consecutive slices, i and $i + 1$ in P is 1 cm. Let us assume that Q has 4 slices with the same gap between the slices (1 cm). Each slice of Q is a square of side length 20 cm. Since the first and the last slices are the extreme slices of P and may define the boundary of the object, any slice of Q cannot fit into P without exceeding the boundary. So P and Q cannot fit each other and are not slices of the same $3D$ object.

As another example, assume each slice of Q contains a square of side 5 cm, with everything else (from the above example) remaining unchanged (see Figure 4.3). It is clear due to the previous example that P and Q cannot fit under translation alone. But before determining a 'No Fit', we also need to consider if the slices of Q rotated in some fashion could fit the slices of P. Notice that the rotation of the slices is constrained by the cutting (slicing) planes. If P and Q were indeed from the same object, there must exist for each Q square, a corresponding P square, such that the transformed image of Q under some rotation, intersects the P square in two (or more) pair of edges. The transformed image of Q can be cut on the outside, in the middle or tangentially by the slice of P. However for this particular case, the length of side of P is greater than the largest line segment inside Q which is its diagonal (of length $5\sqrt{2}$). Therefore, there cannot exist any translation or rotation allowing Q to dock with P, indicating they cannot belong to the same object.

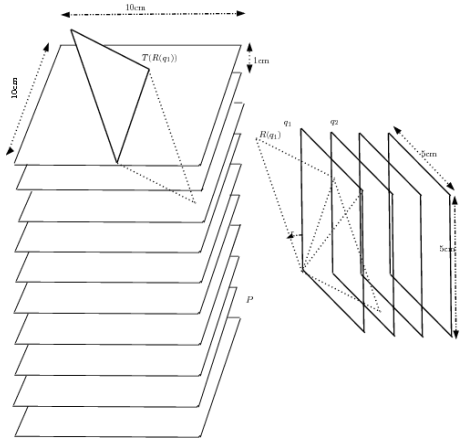


Figure 4.3: Translating and rotating the slices of Q in an effort to dock it with the slices of P.

There is one more case that needs due consideration. Recall that the slices are undersamples of a $3D$ object, the orientation and interval (gap between the slices) is arbitrary. In some cases, the slices of Q may lie in the gap between the P slices. For example, when the object is a pipe and the slope of the cutting planes for P and Q are $\pi/4$ and $3\pi/4$ respectively. The object is sliced in an interleaved sequence of the cutting planes P and Q (alternately) without the slices actually intersecting (see Figure 4.4). It is apparent that even though there is no intersection between the P and Q slices, the slices are in fact generated from the same object. Obviously, we cannot impose intersection between the P and Q slices as one of the necessary conditions for fitting. Every slice q in Q, can either touch some slice p in P or can be located between the sampling gap of two P slices. We would also need to consider

the relative positioning of the slices with respect to one another.

Apart from the several important issues illustrated above, there are a number of other factors which need to be considered. For instance, the Q slices may have many possible fittings with the slices in P. A matching algorithm has to consider all possible fittings for each cross section and determine the fittings that are shared by all cross sections in Q. It is obvious that a straightforward implementation could be rather inefficient. Also, the $3D$ polyhedra to be matched could be very complicated. In a possible fitting for q_j, some polygons may have non-intersecting fittings and the others may have intersecting fittings. An efficient fitting algorithm should consider both these kinds of fittings.

We also notice that although the fitting problem under translation is somewhat similar to the segment matching problem using Hausdorff distance, it seems quite difficult to extend those techniques to the fitting problem. Part of the reason is that the fitting problem only matches a small set of points on the boundary of those polygons, and the set of to-be-matched points is not known in advance. Such an uncertainty introduces additional difficulty.

In the following sections, we first discuss our main ideas to solve the problem efficiently and then present our algorithm.

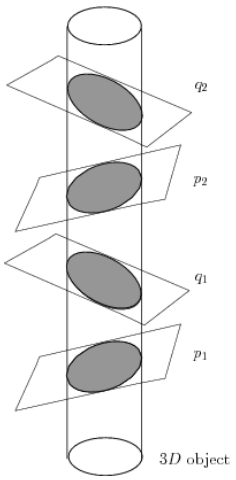


Figure 4.4: Illustration of every slice from one set fitting 'between' the sampling gap of two slices from the other set, resulting in no intersection.

4.3.2 Main Idea

To facilitate the presentation, let $P = \{p_1, p_2, \ldots, p_m\}$, $Q = \{q_1, q_2, \ldots, q_n\}$ denote the two sets of slices, where p_i, q_i is the polygonal representation of the i^{th} cross sectional slice. We say that a slice, q_i, can fit into P if there exists a tuple of translation vector

and rotation matrix, (t, r), which makes the transformed slice, q'_i , belong to either of the following two cases:
Case 1: $q_j + t$ intersects with a subset P_{q_j+t} of P and fits each cross section in P_{q_j+t}.
Case 2: $q_j + t$ does not intersect any cross section in P, and lies in the open strip delimited by a pair of consecutive cross sections in P.

Let $TR_{P,q}$ be the set of (t, r) under which q can be fitted into P, and $TR_{P,Q} = \bigcap TR_{P,q}$, then if $TR_{P,Q}$ is not empty, the algorithm will return a 'Yes' answer.

We will first consider how to fit one piece of slice q_i into P under translation by the way of Case 1, and then continue and consider Case 2 and rotation.

Assume the rotation matrix r is know in advance for P and Q. The goal of the algorithm is thus to find a translation vector v that makes Q fit into P. It's natural to find the translation vector set $T(P, q_i)$ for each simple polygon in Q, then check if there are any common vectors between them. Since the polygon is just a set of edges, we first investigate the line segment case as follows.

Let $T(l, x) = \{v \in R^3 \mid l \bigcap (x + v) \neq \emptyset\}$ be the set of translation vectors which translates l to intersect x. Given two line segments l_1 and l_2, $T(l_1, l_2) = \{v \in R^3 \mid (l_2 + v) \bigcap l_1 \neq \emptyset\}$ is the parallelogram with vertices formed by the difference of endpoints of l_1 and l_2 in the translation space. For a simple polygon $p_j \in P$, each edge s_k of $q_i \in Q$ generates a parallelogram in the translation space with every edge of p_j. The parallelograms generated by the edge s_k with adjacent edges of p_j share one side. These parallelogram patches can be 'glued' together to form a cylinder-like surface (see Figure 4.5) $T(p_j, s_k)$ in the translation space. It is easy to think of $\bigcup_{s_k \in q_i} T(p_j, s_k)$ as the set of translation vectors such that q_i fits into p_j per Case 1. Unfortunately, $\bigcup_{s_k \in q_i} T(p_j, s_k)$ may contain several false vectors.

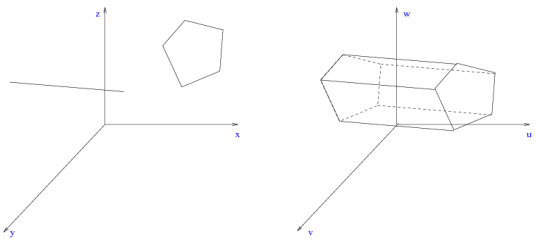


Figure 4.5: Translation space for one segment $T(p, s_i)$ where p is a slice from P and s_i is a segment of a slice $q \in Q$.

Notice that in order for q_i to sucessfully fit into p_j, we require one (or more) pair of edges in q_i to intersect with one (or more) pair of edges of p_j. But $\bigcup_{s_k \in q_i} T(p_j, s_k)$ may contain translation vectors such that the transformed contour of q_i has only one edge intersecting with some edge of p_j. In order to exclude such false vectors, we need

to correctly identify the pair of edge from q_i which intersects with p_j. One simple approach is to try every combination of two edges from q_i. But such an approach would be computationally inefficient. To get around this problem, we present a much faster algorithm based on the planesweep technique, discussed next.

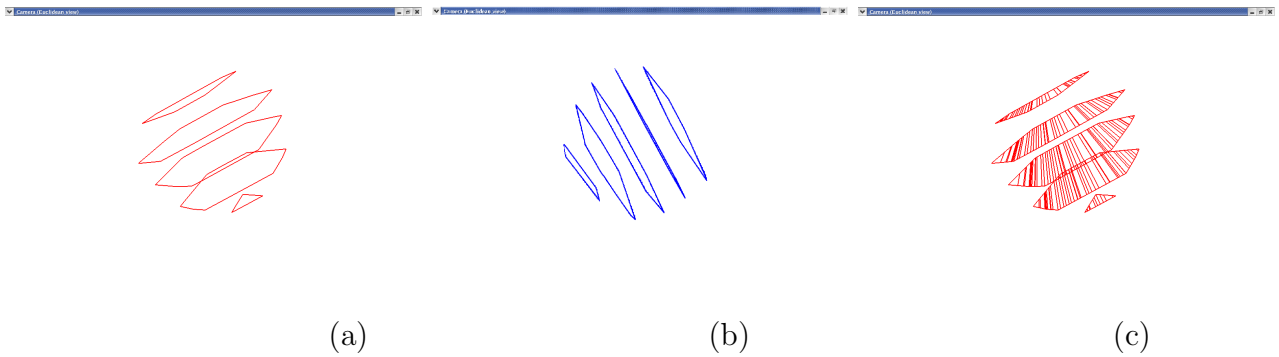
4.3.3 Fitting Algorithm using Planesweep

To find the set of translation vectors V, which can fit every Q slice into P, we first compute the set $V_i = \bigcup_{p_j \in P} T(p_j, q_i)$ (as described in the previous section) which fits one single slice q_i of Q, into P, V being the common intersection of all V_i. However, computing V_i even for one Q slice is not easy, and all slices of P needed to be considered together, as a whole. Our solution is to partition P into layers by adding cutting planes which are parallel to Q at each vertex of the P slices. This approach has one obvious advantage. Instead of P being represented as a set of complex slices, we can now operate with pairs of ordered line segments. This makes the process of finding each of the translation surfaces much simpler. The other advantage is that this partition enables us to fit q_i into the layered part of P as a whole. At each layer, the algorithm will return the pairs of segments cut by the sweep plane for each P slice. It can be implemented in the following way.

We consider a sweeping plane C moving continuously along the norm direction of Q. As C sweeps down (or up), it intersects with a number of slices of P. When C intersects with a slice of P, say p_j, it cuts it at one (or more) pair of edges (we call such edges, 'cut-edges'). At any time point, the information about all current cut-edge pairs is maintained in the sweep-plane-status data structure associated with the sweep plane. The sweep-plane-status, however, may change each time C passes a vertex of p_j. To keep the sweep-plane-status updated, an account keeping procedure is performed whenever an event (vertex) occurs. At this time, edges that have been passed are removed from the sweep-plane-status. Newly encountered edges are made current cut-edge pairs and inserted into the data structure. The set of cut-edges between two consecutive event points forms a layer of P.

We consider the fitting problem in a layer by layer fashion. At each layer, for each pair of cut-edges consisting of s_k and s_l, surfaces $T(q_i, s_k)$ and $T(q_i, s_l)$ are computed. $T(q_i, s_k) \bigcap T(q_i, s_l)$ is the set of vectors which transform q_i to touch s_k and s_l simultaneously. Let U be the union of all 'cut-edge' pairs in the current layer. $\bigcap_{s_k, s_l \in U}(T(q_i, s_k) \bigcap T(q_i, s_l))$ is the set of vectors which can fit q_i into the current layer of P.

To efficiently find all fitting positions at each layer, it is helpful not to consider



(a) (b) (c)

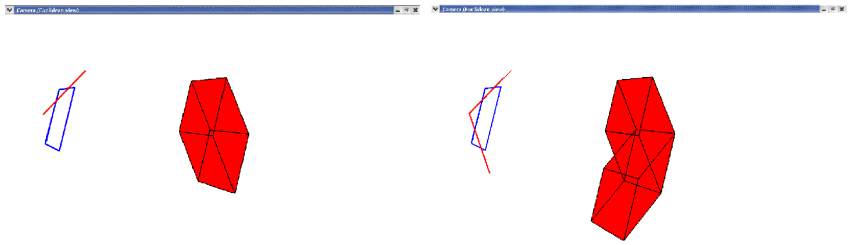
Figure 4.6: (a) First Slice Set P; (b) Second Slice Set Q; (c) "Trapezoidal Decomposition" of P along the direction parallel to Q

q_i as a whole, since q_i could have a rather complicated shape which makes it hard to find all possible translation vectors during the sweep procedure. A better way is to decompose q_i into trapezoids along the direction parallel to P with the height of each traoezoid less than the gap d of two consecutive slice of P.

For each trapezoid t of q_i, if v is a feasible translation vector, it will translate t to t' such that t' fits P by the way of either Case 1 or Case 2. We will compute the translation set incrementally by simulating the sweeping process of q_i across one layer of P. There are two types of significant change that need to be considered, one is when the trapezoid t starts to cross one supporting plane of P slices and the other is when t leaves the supporting plane. The event point for each pair of t and p can be computed in constant time. At any time, for each trapezoid t, it is associated with some P slice in either Case 1 or Case 2. We use this information to compute any possible translation vectors set for t; then compute the common intersection of translation vectors set for all the trapezoids. The common intersection is the set of vectors which can fit the whole q_i into the layered P between two adjacent event points. This computation is relatively easy, since for each t, the translation set is either a segment or a polygon with at most 8 sides or cylinder like object in the translation space.

4.3.4 Improved Algorithm

We now discuss a few important observations that help us design an improved algorithm for our problem. Recall from the previous sections that we had partitioned P into layers along the normal direction of Q. Further, we had decomposed the Q slices into smaller trapezoids to facilitate the fitting procedure, and then applied our algorithm to obtain the set of translation vectors which when applied to one slice set will



(a) (b)

Figure 4.7: Translation Region (in color) formed by translating (a) first line segment s_1 along a polygon q_i and (b) both s_1 and s_2 along q_i

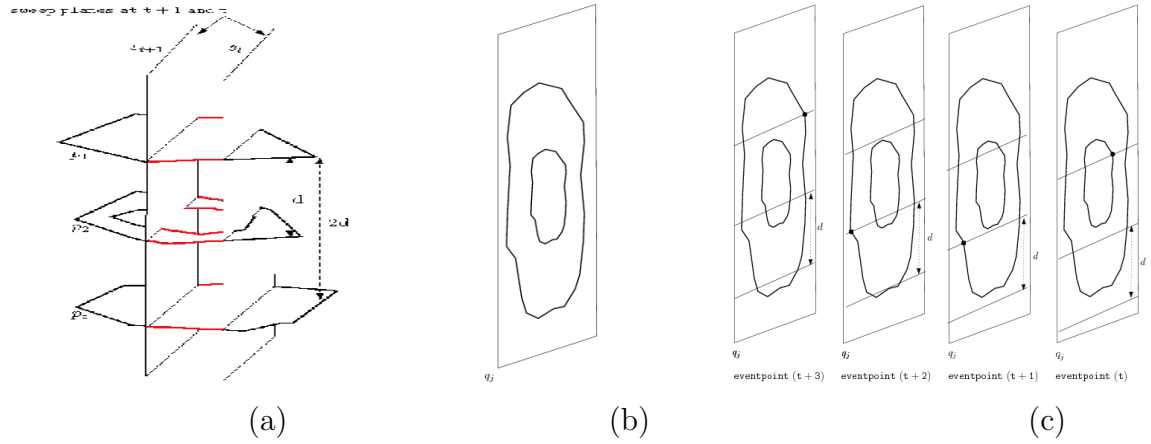
'dock' it with the other. We notice, however, that when P and Q have been decomposed appropriately, there is an easier way to determine whether a particular Q slice is a likely candidate to fit into a layered part of P without the overhead associated with the computational cost of finding the intersections of the parallelograms in the translation space. Though such an approach will still not yield a definitive positive answer with respect to a fitting, it can be used to significantly narrow the pool of likely Q slices that can fit into P between two particular positions of the sweep plane, and as a result speed up the running time of the algorithm substantially.

The basis of this argument is as follows. Assume $L^{t,t+1} = \{L_{p_1}^{t,t+1}, L_{p_2}^{t,t+1}, ..., L_{p_m}^{t,t+1}\}$ is the set of all line segments in a layer of P (between two consecutive event points t and $t+1$) (see Figure 4.8(a) for an illustration). Here, $L_{p_j}^{t,t+1}$ denotes the set of line segments of slice p_j ($\in P$). Assume an arbitrary 'guessed' alignment function which docks a slice q_i ($\in Q$) between t and $t+1$. Notice that for q_i to 'dock' at this position it is compulsory that it has exactly $|L_{p_1}^{t,t+1}|$ line segments in its sub-part (say, m) that intersects with p_1. Otherwise, the alignment *cannot* dock q_i at this position. Also, each of the $|L_{p_1}^{t,t+1}|$ segments in the sub-part m of q_i will intersect with a unique line segment in $L_{p_1}^{t,t+1}$. More importantly, this guessed alignment function will ensure that this invariant is preserved for subparts $m+1, m+2, ...$ of q_i, each subpart (in order) corresponding to one slice of P after p_1 considered in order, say $p_2, p_3, ...$ and so on. This indicates that to find this "guessed" alignment function, we first need to enumerate all possible positions for fitting q_i into a layered part of P.

We sweep each polygonal slice $q_i \in Q$ with a set of m parallel lines, similar to a line-sweep procedure[34]. The direction of the sweep is along the norm direction of P and the distance between two subsequent lines is d (d is the separation of the P slices). We illustrate this in Figure 4.8 (c). Similar to the classical line-sweep, in our

procedure, whenever any of the m sweep lines encounters a vertex, it denotes an event point. At each such event point e_k, we count the number of crossings (intersections) of each sweep-line with q_i and create a string S_k of size m. This is done by concatenating the number of intersections of each sweep-line taken in the order of $(1, 2, ..., m)$. The number of event points encountered during this event-procedure is no more than m times of the number of vertices in the polygonal slice q_j. The set of strings created is denoted as $S = (S_1, S_2, ..., S_k)$.

For any slice q_i to fit into a layer of P lying between positions t and $t + 1$, the string generated by concatenating the sequence of numbers $|L_{p_1}^{t,t+1}|, |L_{p_2}^{t,t+1}|, ..|L_{p_m}^{t,t+1}|$ should match with at least one string in S (see above). This reduces the problem to a classical $2D$ string matching problem, i.e., the problem of finding occurrence(s) of a pattern string within a set of strings. Several string matching algorithms[15; 55] determine in linear time whether one string is a subset of another. This significantly reduces the number of positions where the slices of P and Q can possibly fit and is adopted as a preprocessing phase before determining the parallelograms in translation space yielding a speed-up in the running time of our algorithm.



(a) (b) (c)

Figure 4.8: (a) A layer of P slices, where the set of line segments $L_{p_1}^{t,t+1}$, $L_{p_2}^{t,t+1}$, $L_{p_3}^{t,t+1}$ for p_1, p_2, p_3 *resp.* are shown in red. (b) an individual slice q_i of Q is shown (c) Sweeping the supporting plane of q_i with a set of m sweep-lines that are d distance apart. The event points along the sweep process are encountered when either of the m lines meets a vertex of q_i. Such event points are indicated as red disks.

4.3.5 Approximate Fitting

In the algorithms described in the previous sections slice-sets from P and Q need to fit exactly to be declared a match. For instance, one natural constraint was that each of

the $|L_{p_1}^{t,t+1}|$ segments in the sub-part m of q_i will intersect with a unique line segment in $L_{p_1}^{t,t+1}$ (see Section 4.3.4). We observe, however, that in real world applications, such exact matches cannot be expected. Consider for example the slices of human organs generated using imaging modalities such as magnetic resonance (MR) or computer tomography (CT). Contours of such slices represented as certain geometric objects can only be generated after segmentation. Occasionally, segmentation techniques introduce errors. As a result, strong constraints such as the ones mentioned above may be violated yielding a mismatch. To get around this problem, we propose a relatively minor modification to our previous algorithm. The new algorithm is as follows.

Consider two individual slices $p \in P$ and $q \in Q$. We relax our algorithm to allow p and q to be declared a match as long as q fits into a predefined $\epsilon-$neighbourhood surrounding the contour of p. As a first step, we define a $\epsilon-$neighbourhood of p. Let p_{out} and p_{in} be the outer and inner boundary of the neighbourhood surrounding p (this can be computed by finding the Minkowski Sum[34] of a disk with p). Then, we determine the set of translation vectors $T(p_{out}, q)$ which translates q to fit inside or on the boundary of p_{out}. Similarly, the set of translation vectors that translates q to fit on or inside the boundary of p_{in} is denoted by $T(p_{in}, q)$. The set of translation vectors that fits q within the annuli of p_{in} and p_{out} is computed by $T(p_{out}, q) - T(p_{in}, q)$. The remainder of the algorithm remains similar to the one mentioned in the previous section.

4.3.6 Fitting with Rotation

We also extend our algorithm to deal with cases where the relative orientation of two sets of slices is unknown, or only a rough estimate of the orientation is known. The natural way to solve this problem is to create 'regions' in rotation space (as in case of translation space) that will constitute the set of rotation vectors that fit the slice sets into each other. The difficulty is that in rotational space, such regions are often complex polyhedral surfaces. Determining the bounding surfaces of such polyhedras is computationally difficult and impractical for our purposes. To avoid this problem, we perform a sampling of the rotational space. We place a grid in the parametric space of the three rotational variables (say, the three Euler angles θ, ϕ, and ψ). Since an approximate first estimation of orientation is known, the resolution of the sampling and the bounds of the search space can be decided appropriately. Grid-points are placed in the θ-ϕ-ψ parametric space. The algorithm 'visits' each grid-point and queries the fitting algorithm with the current grid-point as an input

parameter. If the algorithm does not return a 'maybe match' answer at any of the grid points, the two input slice-sets are declared a mismatch. Otherwise, they are considered a probable match and the coordinate values of the last-visited grid-point are outputted.

4.4 Results

The algorithm was implemented in C++ with support from several libraries such as CGAL[37] and LEDA[64] for geometric computing and data structures, GMP[41] for precise arithmetic computations, and COIN[61] for computing the intersection of the parallelograms in translation space (see Section 4.3.2) using its linear programming functionality. The evaluation of the performance issues of the technique proceeded as follows. We chose 100 $3D$ models (in `.off` format) from the Princeton Shape Benchmark[92], a publicly available database of models collected from the internet (see Appendix). Sets of slicing planes of 5 arbitrary orientations were then randomly selected for each model, i. First, for each fixed orientation, a value d was randomly chosen (between 5 and 10) to indicate the number of slices of the model desired for that orientation. Then, the corresponding set of d equally spaced parallel slicing planes was used to determine slices of the $3D$ model, i for that slicing orientation. This process was repeated for each orientation for the model and then for all chosen 100 models yielding a set of 500 slice-sets of "unknown" $3D$ objects from arbitrary orientations. For each evaluation run, two slice-sets were randomly chosen from the 500 slice-sets and used as input to our algorithm. Overall, this evaluation step was repeated 2500 times and the results saved separately. Subsequently, in the analysis phase, the saved output was compared with the original model information associated with each slice set. Clearly, if the two input slice-sets are from the same $3D$ model, the algorithm should return a "maybe" answer. If the two input slice-sets are from different $3D$ models, the algorithm could return a either "maybe" or a "no" answer (suggesting that the two slice-sets *cannot* be from the same $3D$ object). The ratio of the "maybe" answers to the "no" answers for the second case above would indicate the accuracy (or the confidence) of the algorithm. We observe that our algorithm performs pretty well in terms of these parameters. The results are presented in Table 4.4.

In the 2500 evaluation runs performed, 476 runs had slice-sets chosen from the same object ($3D$ model). As can be expected, the empirical performance of the algorithm is in agreement with the theoretical discussion in Section 4.3. It identifies

the slice-sets from the same object perfectly (100%). The other 2024 runs of the algorithm had slice-sets from different objects as inputs. The algorithm correctly identified the inputs as *not* belonging to the same object in about 97.5% of the cases. It gave a "maybe" answer in about 50 cases (2.5%). A good percentage of these 50 cases could be runs where the representative $3D$ models were somewhat similar and the resolution of slicing was poor (d was small). As a result, some dissimilar characteristic of the $3D$ models may not have been captured in its slices resulting in a maybe answer. We believe that it may be desirable to have a slightly larger value of d (better slicing resolution) in situations where the to-be-compared unknown $3D$ models are expected to be quite similar in shape.

The evaluations were performed on a $2.6GHz$ machine with $1GB$ of RAM running GNU/Linux. The running time of each evaluation run was 0.48 seconds on an average over 2500 runs.

Type of Slices	% of times positive answer	% of times negative answer
From same object	100%	0%
From differnt object	2.5%	97.5%

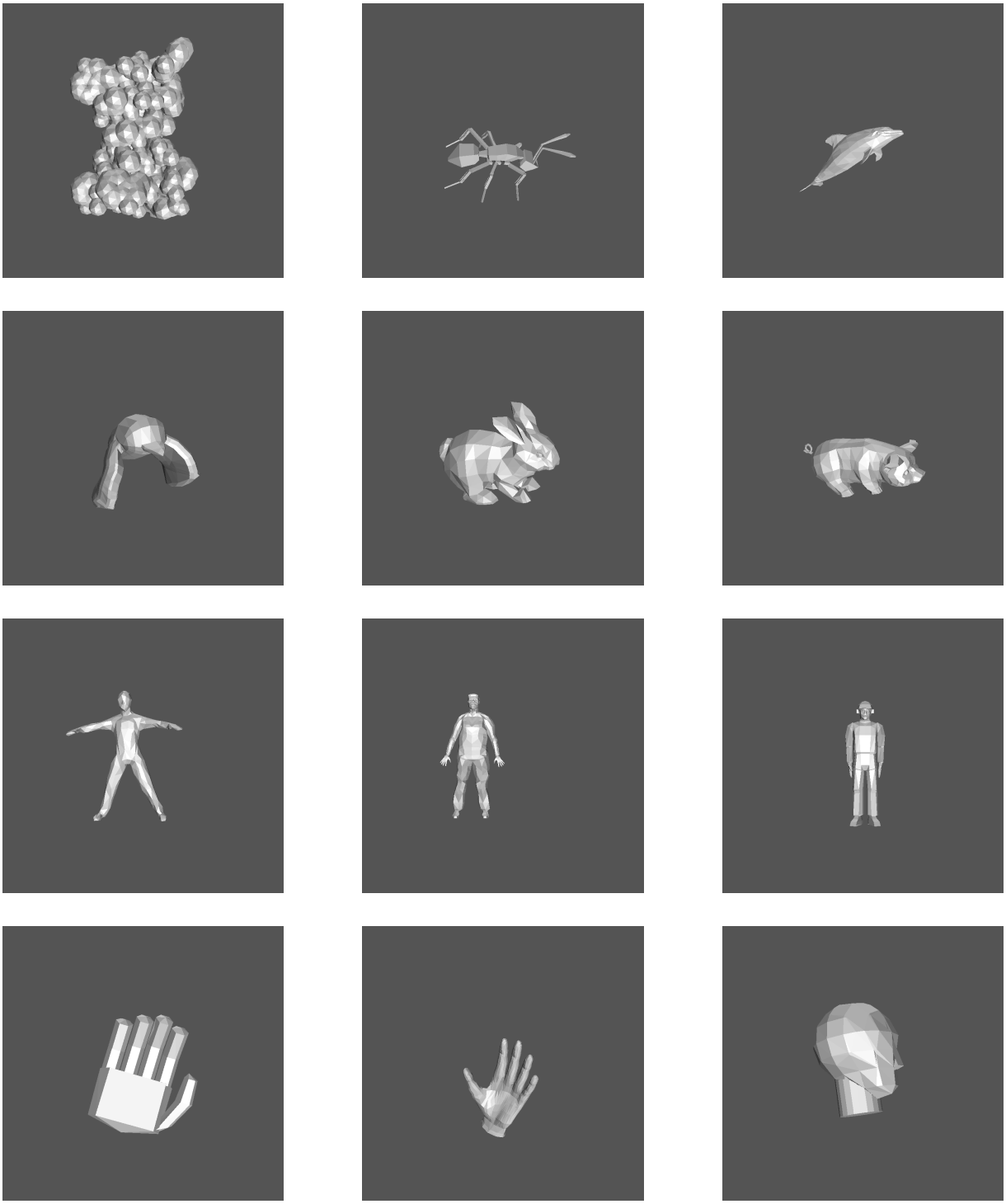
4.5 Conclusions

In this chapter, we have extended the conventional definition of partial matching to include problem instances where one needs to determine whether two arbitary input sets of polygonal slices of an unknown $3D$ object(s) correspond to the same object. The techniques proposed here enable answering these questions as a *maybe* (plausible) or a *certain negative*. It should be noted that the algorithm does not assume a very high sampling resolution in the input polygonal slices (here, resolution means distance between two consecutive polgonal slices). In fact, in degenerate cases the resolution may be quite poor and as a result an important physical feature of the unknown object may have been lost in its slice representation. Given two such under-sampled slice sets as inputs where some key differentiating physical feature may not have been captured, it seems that determining a *plausible* or a *certain negative* answer to whether they represent the same $3D$ object is the best one can do. Inspired from well accepted ideas in computational geometry, we propose efficient algorithms for addressing such questions. Apart from the theoretical results presented, we discuss a number of interesting applications of the proposed approach in such diverse areas as

molecular imaging and PACS/imaging informatics. Evaluation of an implementation of the technique on a subset of models taken from a widely used $3D$ shape benchmark have been presented and running-time of the program and other related issues discused. The evaluation results are in agreement with our motivating observations - the technique correctly determines matches when the input slices are from the same $3D$ object (100%) and yields a high degree of accuracy when the input slices are taken from disparate objects ($\geq 97\%$). Our algorithm is extremely easy to implement and we believe that apart from its immediate motivating application in molecular imaging (where $3D$ resolution of the microscopes is limited), the ideas presented will be helpful in a number of other disciplines.

4.6 A few examples of the 3D models used

The following are some of the models from the Princeton Shape Benchmark [92] which were used for evaluation of our algorithm. The results of the evaluations are presented in Section 4.4.



Chapter 5

Efficient techniques for analyzing time-lapse microscopic image sequences of living cells

5.1 Problem Description and Previous Works

Understanding the functional dynamics of the mammalian cell nucleus remains the final frontier of modern nuclear biology. To this end, recent developments in microscopy, imaging and labeling have led to a significantly improved view of the cell nucleus. This technology has enabled the observation of replication and transcription sites and their organization into higher order units or "zones". It is believed that this dynamic interplay and "re-zoning" may form the structural basis for elaborate global coordination of replicational and transcriptional programs within the mammalian nucleus [10; 11]. However, inspite of significant progress in microscopic imaging technologies in recent years, the computational task of analyzing the nuclear images to derive semantic information is still plagued with numerous challenges. The key impediment is the presence of an enormous amount of chromatin foci in the nucleus which are small in size, and can be hidden by diffused signals or quasi-foci (formed by the overlapping of some neighboring foci), making segmentation and other computational tasks extremely challenging. Moreover, these chromatin domains exhibit rather complicated movement including non-uniform translation, rotation and deformation. This problem is further compounded by the limited spatial and temporal resolution capacity of current microscopic techniques which introduce under-sampling or "gap" between acquired images. Under these circumstances, it is extremely difficult to do real-time visualization of such nuclear activities. Biological analysis of such data is

generally the result of long sessions involving manual calculations and hypothesis (based on experience), making it human error prone and very tedious. To regenerate smooth motion and deformation of the genomic structures, we need an effective way to first segment the chromatin foci and then apply motion tracking and intensity regeneration techniques to fill in the "gaps" accurately. Such regenerated information, if available, can also be used as a reference for improving the low contrast images in an image sequence. These problems motivate the design of a technology that allow biologists to do real time visualization of nuclear activities within the scope of existing imaging modalities.

The problem of segmenting genomic structures from microscopic nuclear images is particularly difficult because not only are the objects under consideration extremely small, there is very little meta-knowledge about the system and the image acquisition and quality depends heavily on experimental conditions, slight alteration of which can produce substantial difference in the acquired images. As a result, commonly used segmentation techniques fail to produce desirable results when applied to such images. Recently, researchers in the field of image processing are increasingly using image simplification as means to facilitate segmentation[65]. Simplified images contain less details but preserve the relevant information of the original image. Therefore segmentation routines work faster and more accurately on such images. However, to our best knowledge, no previous result has been reported on the application of image simplification in segmenting genomic structures from microscopic images of living cells.

The problem of recovering motion and deformation of moving objects, given only a few snapshots of the objects during their motion, has been a challenging problem in various fields such as computer vision and pattern recognition, and robotics. Most of the previous works have focused on recovering the rotation and translation of a single object in 3-D space from consequtive 2-D images, assuming that no deformation occurs in the period between two consecutive images [40; 62]. More recently, deformable models of objects have been adopted for simultaneous estimation of shape and motion of one or more 2-D or 3-D objects [33; 48; 59]. These approaches provide interesting insights into possible interpolation strategies and work well for the specific objects, images or assumptions they were designed for. Unfortunately, these techniques do not extend well to microscopic images for a number of reasons. They were primarily designed for large sized objects where sufficient information about the mechanics and physics behind the motion is available or can be assumed. For genomic structures, however, where no such data is available, it becomes necessary to exploit the limited

information hidden in the images in an effort to recreate an accurate motion of the objects under consideration.

In this chapter, we present a suite of geometric techniques addressing the problems mentioned above. First, we propose a novel approach for simplifying the intensity surfaces of the images, which improves the quality of the segmentation. We then suggest techniques to determine the shape and motion of genomic structures for any number of intermediate (or unavailable) time points in a 2D image sequence. We further extend our idea by not only regenerating the contours but also the continuous changes of intensity surfaces from one image to another. This allows us to generate as many "missing" in-between image frames as desired given as few as two time separated image frames. When a number of nuclear images acquired in time-lapse mode are available, our techniques allow an online recreation of a "movie" sequence, enabling a real-time visualization of nuclear processes with very little manual interaction. This provides a far less tedious biological interpretation of the imaging data.

The remiander of the chapter is organised as follows. First we discuss at length the different methodologies adopted and the intutions behind them. In the first subsections under this part, we overview our methods for simplification and segmentation of images. The next two subsections are devoted to discussion on motion tracking and intensity surface recovery approaches, applied on the set of objects obtained from these images. Finally, we illustrate our techniques by presenting results on microscopic images of the cell nucleus.

5.2 Method

5.2.1 Simplification and Segmentation

We begin with a sequence of 2D microscopic images of the same set of genomic structures (representing chromatin domains or chromatin foci) taken at different time points. Microscopic images are often blurred in portions by clouds of diffused signals which complicate the process of segmentation and intensity recovery. Therefore, our first goal is to simplify the images to make segmentation easier. The next sections describes a novel approach for simplification and segmentation.

Simplification

Image simplification is a fundamental problem in image processing and several other applied areas such as computer vision and robotics. Simplified images not only have a mathematical representation, but also provide a ground work for speeding up a

number of computation-intensive processes such as segmentation, matching, and registration.

The most commonly used approach (called *rectangular partition*) in image simplification is to partition the images into axis-aligned rectangles (or squares in quadtree partition) such that the intensities of all pixels in each rectangle are roughly the same [77]. However, this approach works best for images which have large sized objects or background with similar intensities. When the images contain many small objects or exhibit dramatic change in intensities, such approaches often encounter difficulty in reducing the image size. Since microscopic images of a cell nucleus can contain thousands of small foci, rectangular partition do not yield satisfactory results on these images. Also, existing segmentation algorithms (such as threshold-based, edge-detection-based, and region-base algorithms), when applied directly on such images, often fail to detect or generate the exact boundaries for all foci contained in the image (See Fig 5.5).

In this chapter, we propose a novel technique for simplifying microscopic images and then apply segmentation algorithms on the simplified images. The method is based on analysis of a recently observed interesting phenomenon which shows that the image profile around each (replication or transcription) focus follows certain *exponential distribution* (see Figure 6.1 (a) and 6.1 (b)).

Based on this observation, our main idea for simplification is to determine a set of *approximation functions* to approximate the intensity surface of the image. More specifically, we solve the following problem: Given a set $P = \{p_1, p_2, \cdots, p_n\}$ of pixels with each p_i associated with an intensity value $I_i > 0$, find a small set $F = \{f_1(\cdot), f_2(\cdot), \cdots, f_k(\cdot)\}$ of 2-variable functions with each $f_i(\cdot)$ defined by a generalized normal distribution surface such that the intensity I_i of each pixel p_i is closely approximated by the maximum value of the k functions at p_i, i.e., $|I_i - \max_{j=1}^{k} f_j(p_i)| \leq \epsilon$ (or $|I_i - \max_{j=1}^{k} f_j(p_i)| \leq \epsilon I_i$) for some small constant $\epsilon > 0$. Each function $f_i(\cdot)$ follows a generalized 2-D normal distribution

$$f_i(X) = \sigma e^{-(X-\mu)^T Q (X-\mu)},$$

where Q is a positive definite matrix (i.e., the intersection of $f_i(\cdot)$ with a plane $z = c$ is an arbitrarily oriented ellipse).

The algorithm we propose is as follows.

1. Compute a set P of local intensity maxima in image A.

2. For each $p \in P$, conduct a radial sweep and find a set of t approximate iso-intensity (error tolerance ϵ) contours denoted as C_p

3. For each $c \in C_p$, compute the smallest enclosing ellipse e_c for all points on c. The peak p, ellipse e_c, and the average intensity value of c uniquely determine a normal distribution function f.

4. For each f computed by step 3, put a window w on A centered at the peak of f and extract a sample submatrix m, compare how well f matches m, and choose the best matched function f.

5. Repeat step 2, 3 and 4 for all pixels in P and record the set F of normal distributions obtained.

6. For each pixel $q \in A$, find the best normal distribution in F to approximate it. If F cannot well approximate q, generate a new normal distribution for q and insert it into F.

The running time of the above algorithm depends crucially on the time taken to execute step 3. In this step, we apply an algorithm similar to Welzl's randomized algorithm for computing the smallest enclosing ellipse of an m-point set in 2D space [36]. The main idea of Welzl's algorithm is to randomly choose a point q from the m-point set Q, and recursively compute the smallest enclosing ellipse e for the remaining $m-1$ points. If q lies inside or on the boundary of e, e is the smallest enclosing ellipse for all n points, otherwise q must be on the boundary of the optimal enclosing ellipse. This algorithms runs in expected $O(m)$ time. The details of the algorithm is as follows.

EXTENDEDWELZL(Q, R)

1. If $Q = \phi$ or $|R|$=5 then compute the smallest ellipse e with a set R of points on the boundary of e and return e.

2. Otherwise, choose a point $q \in Q$ uniformly at random. Then recursively compute the smallest enclosing ellipse e with inputs $Q \setminus \{q\}$ and R. That is, e:=EXTENDEDWELZL($Q \setminus \{q\}$, R).

3. If $q \in e$ then return e.

4. Otherwise, recursively compute the smallest enclosing ellipse e with inputs $Q \setminus \{q\}$ and $R \cup \{q\}$. That is, e:=EXTENDEDWELZL($Q \setminus \{q\}$, $R \cup \{q\}$). Return e.

The algorithm for simplification can be easily implemented and runs in $O(n \times k)$ time, where n is the total number of pixels and k is the number of normal distributions. Since k is in general much smaller than n, the algorithm runs in near linear time.

Segmentation

After simplifying the raw cell nuclear images, the set of approximation functions can be used for speeding up the segmentation process. By combining the results of our approximation algorithm with the merging segmentation method [94], we are able to accurately segment DNA foci from nuclear images and avoid the influence of diffused signals by using relative thresholds. The algorithm is as follows.

MERGESEG(T_1, T_2, T_3)

1. After approximating the image, compute a set R of regions, which is the union of all connected regions in the domains of all normal distribution functions.

2. For each $r \in R$, if r contains a local maxima with intensity greater than T_1 then put r into the candidate foci set CR. Otherwise put it into non-candidate foci set NCR.

3. For each $r_i \in CR$, recursively merge it with its adjacent non-candidate region $r_j \in NCR$. If $\frac{W}{min(l_i, l_j)} \leq T_2$, where W is the number of weak edges on the common boundary, l_i and l_j are the perimeter of r_i and r_j respectively.

4. For each $r_i \in CR$, recursively merge it with its adjacent non-candidate region $r_j \in NCR$. If $\frac{W}{l} \leq T_3$, where l is the length of the common boundary.

5. Repeat step 3 and 4 until no more merging occurs, then CR is the set of all segmented foci.

Contour Simplification

The segmentation process returns the contours of all foci in the image. These contours can be represented as a set of polygons denoting the boundary of the genomic structures. However, due to unavoidable data noise, polygonal representation of contours (generated by segmentation algorithms) often consists of huge number of points (a point for every pixel in the boundary). This large set of points not only hides the real boundary of the genomic structures but also consumes a large amount of computation time when further processed. To avoid this problem, we use a contour simplification technique which smoothes out the rugged contours by reducing the number of points but preserves the underlying shape of the objects. It is important to note here that we assume that there is no interdependence between the shapes of the polygonal contours (otherwise the problem becomes NP-complete problem [35] and cannot be solved in polynomial time unless P=NP).

Our simplification algorithm adds an edge between every two vertices of the input polygon, whenever every intermediate vertex between them, taken in order, is at a distance lesser than a given threshold (or error tolerance ϵ) from the edge joining them. It then employs a shortest path algorithm (e.g., Dijkstra's algorithm [30]) for finding a path from the starting vertex to itself with a constraint that the path should contain at least one edge. The vertices on such a path are the vertices of the output polygon. The running time of the algorithm in worst case is $O(n^2)$ where n is the number of points on the boundary of the un-simplified polygon . However, in most cases, it runs in near linear time. This process yields the set of contours of both the initial and final images modeled as polygons.

5.2.2 Motion Tracking

The following section describes the various phases of motion tracking. In the first phase, we analyze our approach for determining correspondences between polygonal sets, followed by discussion on techniques for determining rigid and non-rigid transformations between the genomic structures under consideration.

Determining the correspondence between polygon sets

Once the genomic structures have been represented as simplified polygonal contours, the next step is to determine the correspondence between polygonal objects in the sets corresponding to the initial and final time points. It is necessary to mention here that biologists believe that the overall movement exhibited by chromatin domains are a result of two kinds of influence - one that is attributed to the movement of the whole cell nucleus and the other due to their own independent movement (which is somewhat restricted) [80]. This leads us to believe that if we correct for the movement of the cell nucleus, chromatin domains in consequtive time points will be within small neighborhoods of each other. However finding the global movements affecting the chromatin domains is not an easy step. There are several techniques like Nearest Neighbor and Iterative Closet Point Transform (ICPT) which are used to determine transformations that best allign two point sets. However such techniques cannot be directly applied when we consider polygonal sets instead of point sets. To overcome this difficulty, we propose a correspondence technique which takes into account not only their relative positions but also their shapes while finding the best possible matching.

For this purpose, we impose some reasonable restrictions on the movements of

the polygons which are in accordance with the observations of biologists. The first assumption is that a polygonal object is more likely to move to a position in its vicinity than to a position which is far off. We also assume that there exists a global transformation which can move almost every polygon in one set to a small neighborhood of at least one polygon in the other set. The basis of this assumption is to account for the movement of the cell nucleus containing the chromatin domains. It is important to note that we do not require that all polygons are affected by the same transformation. On the contrary, each polygonal object may have an unique local transformation, which we will discuss in the next section. However we do assume the existence of a transformation that generalizes our first assumption for a single polygon to a set of polygons.

This step is useful because once such a transformation is determined, local neighborhoods can be examined for proximity. For example, given polygon sets P and Q if a transformed polygon $p_i \in P$ overlaps with $q_j \in Q$ (or a neighborhood of q_j), it is an indication that q_j may be a possible correspondence for p_i. The notion of neighborhood can be thought of as a square of radius R (or any bounded region) centered roughly in the middle of each $q_j \in Q$. Clearly, when the neighborhood R is large, a global transformation satisfying the criteria above can be easily found, hence a large R is always feasible. To obtain better accuracy, we employ a binary search procedure on R, to determine the smallest *feasible* R, R_{opt}.

The problem of establishing correspondence between polygon sets is an optimization problem which can be formulated as follows.

Input: Two sets of polygons.

Decision problem Does there exist a transformation which when applied to all polygons in one set will bring each of them within a close *neighborhood* R of at least one polygon in the other set ?

Optimization problem If such a transformation exists, then find the optimal transformation which minimizes the neighborhood. Otherwise find the one which bring the most number of polygons to their neighborhoods.

Our algorithm for the decision version of the problem is as follows.

Given: Two sets of polygons P and Q with n and m polygons respectively, their centers, and minimum (R_{min}) and maximum value (R_{max}) of the square determining the neighborhood.

1. Start from $R = R_{min}$, draw squares of diagonal d around the centers of each polygons in P.

2. Make a guess that a polygon Q_j is mapped to a polygon P_k. Let T be the set of

transformations that brings center of Q_j on the boundary of the square around center of P_k.

3. Check whether T when applied to centers of all other polygons in Q, intersects at least one of the squares formed around the centers of P.

4. If not, repeat *Step 1* to *Step 3* by changing the R in a binary search manner till the current value of R differs from a previous one by a small margin, or R exceeds R_{max}.

5. If no transformation is found, return `false`.

The decision version of the problem takes $O(n \times m)$ time. This is because R_{min} and R_{max} are constants and it takes constant time to determine a value of R that results in intersection, provided the guess is correct or to terminate if the guess is wrong. By an exhaustive or random search method, the optimization problem will also take polynomial time ($O(n \times m)^2$) to determine the mapping of polygons that results in an optimal transformation. Rotation can also be introduced by an iterative method on the rotation angle.

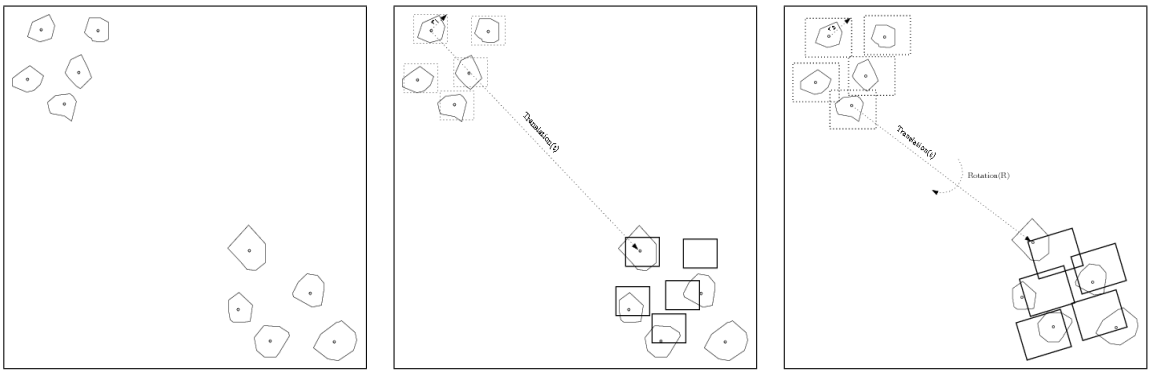
The above procedure yields a reasonably small value of R_{opt}. It also gives for each $p_i \in P$ (or $q_i \in Q$) , one or more probable candidates for correspondence in Q (or P) based on R_{min} and the overlap criteria. We follow up this procedure with another refinement process by modelling the problem as the well-known Maximum Bipartite Graph Matching[30] problem. In our case, the vertex sets are corresponding polygons of P and Q and the edge weights are the distances between individual polygons in these two subsets.

5.2.3 Determining local transformations in case of one-to-one mapping between polygons

This section reviews the techniques for determining the local transformations that resulted in one polygon being transformed into a corresponding polygon. Such transformations are categorized into two parts: *Rigid body transformations*, consisting of scaling, translation, and rotation, and *Deformable transformations*. We deal with each such transformation separately in the following sections.

Scaling

When the size of one of the two considered polygons is much larger than other, a transformation that possibly resulted in this phenomenon is scaling. We employ a simple



(a) (b) (c)

Figure 5.1: (a) Two sets of polygons for establishing correspondence (b) A transformation not establishing correspondence for every polygon. (c) A transformation establishing correspondence for each polygon.

routine to enlarge or decompress one polygon to match the other in size and keep a note of the scaling factor which is used later on while combining transformations.

Translation and Rotation

In this section, we discuss the problem of determining a rigid motion T, comprised of rotation and translation, which when applied to one polygon, best matches it with the other. To determine the quality of the matching, we use Hausdorff Distance as the matching metric. Hausdorff distance denoted by δ_H, is a commonly used distance metric for measuring similarity of shapes and is defined as follows

$$\delta_H = \max(\delta_H(A, B), \delta_H(B, A)) \text{ where } A \text{ and } B \text{ are arbitrary point sets.}$$

The advantage of using Hausdorff distance is that it can be computed in linear time [5] when polygons are convex and in $O(nlogn)$(n being the total number of vertices in the polygons) when the polygons are simple but not convex[43]. In this chapter we adopt Hausdorff distance as the metric but introduce a variation of the techniques used in [43].

We use the concept of *Minkowski Sum* to define a region of maximum deviation around one of the polygons. Minkowski sum for two polygons A and B is defined as $A \oplus B = (x + y | x \in A, y \in B)$. The notion of Minkowski Sum is similar to the morphological operation of *dilation* commonly used in image processing applications. In this case, given two polygons P and Q, we find the Minkowski sum of a square R of size r with P, which denotes an inner and outer boundary around P enclosing the

region of deviation S. The image $T(Q)$ of Q, under an appropriate transformation T, should always lie in S. In essence, we set a limit on the value of Hausdorff distance, denoted by r and then determine a transformation T such that the Hausdorff distance between $T(Q)$ and P falls within that limit. After rotating Q by an appropriate angle, finding the adequate translation is realized by determining for each edge e of Q a set of translations, under which e will always lie within S and then finding the intersection of all such sets in translation space. When no such transformation exists i.e., if there is no intersection, we modify the initial value of r and repeat the above procedure till a transformation is obtained for the minimum possible value of r . When the maximum value of r and the maximum degree of rotation is fixed, the complexity of the above procedure is $O(n \times m)$.

Shape Deformation

We propose a method called *shape-deformation* to account for non-rigid shape changes in polygons. Our method relies on the assumtion that, shape changes affect the peripheral region more than the central region. Apart from that, the main idea behind this routine is that the global non-rigid deformations can be broken down into independent non-rigid transformations occurring on small sections of the periphery. The method works as follows.

Input for this method are the polygons P and $T(Q)$ for which the Hausdorff distance is minimized. Shape-deformation procedure first computes one or a few central points in the common intersection of P and $T(Q)$ so that all points in $P \cup T(Q)$ can be guarded (i.e., visible to) by at least one of the central points. For each central point o, draw line segments from o to all visible vertices and intersections of the polygons. Any pair of such line segments, say the i^{th} pair, encloses a portion of the boundary, P_i and $T(Q)_i$, of the polygons P and $T(Q)$, respectively. The technique then proceeds to determine the image of $T(Q)_i$ under the transformation $S_t(T(Q)_i, s, \theta)$, where t is an intermediate time point between the initial and final time point and s and θ are the scaling factor and angle of rotation for the local scaling and rotation which resulted in $T(Q)_i$ in being transformed to P_i. Ultimately, all such local transformations $S_t(T(Q)_i)$ are combined to yield the deformed shape of $T(Q)$ at time point t.

The number of vertices and intersections can be at most $2(n+m)$ if the polygons are convex or stars and $O(n \times m)$ if they are arbitrary simple polygons, where n and m being the vertices in P and Q respectively. If the number of intermediate time points is t, the shape-deformation procedure takes $O(t \times (n+m))$ (or $O(t \times n \times m)$ if P and $T(Q)$ are arbitrary simple polygons) time to output the intermediate shapes at

all such time points. When t is large enough, a continuous motion and deformation can be obtained.

Determining local transformations in case of one-to-many mapping of polygons

When the procedure for determining the correspondence between polygon sets maps a polygon in one set to more than one polygon in the other, this polygon has to split up into multiple parts with each part corresponding to a polygon in the other set. map each . A symmetric case is that more than one polygon in one set are mapped to a single polygon in the other set. The following approach can be used to deal with the first case, and the second case can be handled similarly.

Let Q be the polygon to be broken up into k parts $P_1, P_2, ..P_k$. Our approach first computes the convex hull CH of $P_1, .., P_k$, and then determines the transformation for Q and CH under scaling, translation, and rotation. Let $T(Q)$ be the resulting image of Q from the transformation. To split $T(Q)$ into k parts, our technique first computes the central point for each $P_i, 1 \leq i \leq k$ and then construct a Voronoi diagram([34]) for the set of central points. The Voronoi diagram partitions $T(Q)$ into k regions. For each region R_i, our approach performs the shape-deformation algorithm given in last section to deform R_i into P_i.

5.2.4 Intensity Surface Recovery

Once the contours for all desired intermediate images have been generated, we need to recover the intensity surfaces corresponding for each pair of chromatin domains in P and Q. Recall that the intensities of these chromatin domains can be approximated by a set of normal distribution functions (see Section 5.2.1). Our objective is to reconstruct for each intermediate contour, the set of normal distribution functions that should represent its intensity surface.

Before discussing the details of the algorithm, we will formally define the notations. Let F_P and F_Q be the set of normal distribution functions for a corresponding pair of chromatin domain P and Q respectively. Consider $T_{P \rightarrow Q}$ to be the transformation which when applied to P, yields Q. This transformation is symmetrical i.e., $T^{-1}_{P \rightarrow Q}$ when applied to Q will give P. By the same token, the transformations for generating the kth intermediate image P_k from P can be represented as $T_{k_{P \rightarrow Q}}$.

Our algorithm works on the principle that the domains of normal distributions are equally affected by transformations such as rotation, translation, scaling and shape-

deformations, as are the contours containing them. Hence we use the transformations generated previously, to obtain the domains of normal distributions at the intermediate time points. Also, their intensities are likely to be influenced by the intensities of both the initial and final images. Therefore, we generate the intermediate domains of the normal distributions from either side.

The distribution functions of F_P (and F_Q) are first projected on to the xy plane to yield a set of regions, represented as D_P (and D_Q), as shown in Fig. 5.3. Though this procedure is "lossy", the outer periphery of the union of the members of D_P and D_Q makes up P and Q. $T_{k_{P\to Q}}$ is then applied to each member of D_P which results in a set of 2D domains. Similarly, $T_{k_{Q\to P}}$ is applied to D_Q to obtain another set of domains. These two set of domains can be thought of as "clones" of D_P and D_Q under transformations in the forward and reverse directions and cover the same region bounded by the contour of $T_{k_{P\to Q}}(P)$. At this stage, for each pixel inside the contour $T_{k_{P\to Q}}(P)$, we run a simple check for membership in both $T_{k_{P\to Q}}(D_P)$ and $T_{k_{Q\to P}}(D_Q)$. Note that owing to the construction, a pixel has to belong to at least one domain in both of these domain sets. Once the membership is established, we refer to their corresponding "clones" in D_P and D_Q and the pixel intensity is then expressed as an affine combination of the intensity values originally associated with the untransformed clones i.e. for a pixel p_{k_l} ,if $I_{D(P)_i}$ and $I_{D(Q)_j}$ be the intensity values associated with the clones $D(P)_i$ and $D(Q)_j$ of its parent domain, then

$$I_{p_{k_l}} = \alpha * I_{D(P)_i} + (1-\alpha) * I_{D(Q)_j}$$

where α is the ratio of time intervals from the initial to the intermediate time point and the intermediate to the final time point.

5.3 Results

We implemented our algorithms in C++ using CGAL-2.4 and LEDA-4.4 on a 1.6 GHz machine running GNU/Linux. For simplicity of presentation, we will discuss results for each sub-section in the order they appeared in the chapter. The images we consider are $2D$ images of GFP (Green Flourescent Protein) labeled replication sites. Movements of these sites represent the active process of replication. The images were obtained by flouroscopic labeling of the cell nuclei. This was followed by image acquisition by a optical microscope after a period of about two to three days.

We begin by presenting the results of image simplification algorithm (Section 5.2.1). We evaluated our algorithm using a set of 10 microscopic nuclear images as

inputs. These images were of sizes between 400×400 to 450×550, a standard size for miscroscopic images obtained from such microscopes. Roughly 25 - 60% of all pixels in these images were foreground pixels. Our method achieved a high accuracy for all test cases yielding images that were almost the same as the original image (see Fig.6.1 for an example). More than 80% pixels and 70% foreground pixels in the tested images had intensity differences smaller than 5%. It was also observed that more than 97% of foreground pixels and 98% of pixels had intensity differences smaller than 10%. The number of normal distributions generated was between 1000 to 3000 and this is directly proportional to the percentage of foreground pixels(which is obvious). The domain of each normal distribution consisted of several regions, and the total number of connected regions varied between $10,000$ to $60,000$. This was as a direct consequence of the percentage of foreground pixels. We also evaluated rectangular partitioning algorithm [70] on some data sets and found that it generates roughly 10 times as many rectangles as the number of connected regions returned by our algorithm.

After simplifying the images, we ran our segmentation algorithm on the simplified images. In order to evaluate our results, we also applied conventional watershed segmentation [94] on the non-simplified images. Figures 5.5 (a) and (b) illustrate the segmentation of a small portion of the image generated by our method and the watershed technique respectively. In each case, the focus boundaries are denoted by thick lines. In the watershed method, sometimes foci are split into several parts and diffused regions are incorrectly reported as foci, which is not desirable. However, using our method, each focus is precisely segmented and no diffused region were observed.

Next we discuss the results of our contour simplification algorithm. We mentioned earlier in Section 5.2.1 that simplification is done based on a user-specified error tolerance ϵ. Table-5.1 shows the average percentage number of points preserved as well as the average percentage area retained for different values of ϵ. It is evident that for smaller values of the error tolerance, the percentage reduction in points is quite high (about 70%) whereas the percentage reduction in area is almost negligible (1-2%). For our purpose, we choose 0.5 pixel as the user specified error margin for contour simplification. The intution behind choosing this value is that for 0.5 pixel error margin, the percenatge area reduced by only 1.41%. Also, for ϵ values greater than 0.5, the decrease in area is more pronounced compared to the reduction in number of points. For example, in Table-5.1, for 0.75 pixel value, the area reduction is about 8.5% whereas there is only a slight reduction in number of points from the 0.5 case. Figure 5.6 shows a polygonal contour before and after simplication with a

0.5 error tolerance. It is evident that the shape of the polygon remain unchanged even though there is a 27% reduction in the number of points.

Error Tolerance(in pixels)	Perc of points preserved	Perc of area preserved
0.25	48.689	99.98
0.5	30.015	98.59
0.75	29.37	92.49
1.0	28.25	84.28

Table 5.1: Results of Contour Simplification

For the correspondence finding and motion analysis, we tested our algorithms on a set of 12 images of replication sites taken at an interval of 3 secs each. The pixel size of the microscope was 7 microns. Some of these images contained *outlier sites*. These outliers are those sites that have a sufficiently strong signal but are geographically separated from most of the other replication sites. Biologically, they cannot be considered replication sites. Distance thesholding techniques often have trouble eliminating outliers as their distance separation from replication sites varies significantly from one data-set to another. As a result, a simple *distance thresholding* may not work satisfactorily. For our algorithm in Section 5.2.2, we exploited the fact that such outliers constitute only a small fraction of the total number of replication sites. To avoid iterating indefinitely in an attempt to find matches for all replication sites some of which may be outliers, we allow the algorithm to return a *yes* answer as long as it has found a correspondence for more than 95% of the replication sites in both polygon sets. In our analysis of the results of Section 5.2.2, we observed that the average global translation was about 0 to 2 pixels and the average radius of square denoting the neighborhood was about 3 pixel.

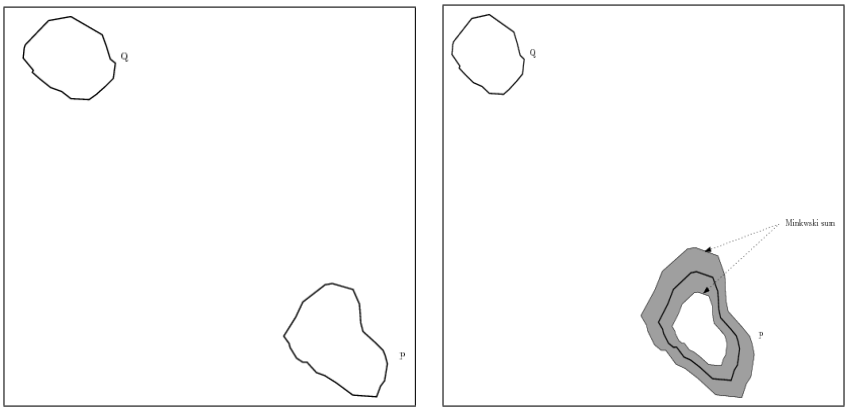
We now analyze the transformation values obtained from algorithm in Section 5.2.3. The goal is to obtain a good understanding of the motion parameters for replication sites. We approach this goal in two ways. First, we attempt to track individual replication sites from one time point to the next. Secondly, we derive motion parameters characteristic of the movements of these replication sites. Figure 5.8 (a) shows the trajectory of a single replication site starting from time point 0 upto time point 8. Analysis of these trajectories reveal that most replication sites move to and fro over a small region. We now quantitively analyze the motion characteristics of these replication sites. Observe an average measure of the motion values may be baised if the number of outliers present is greater than 5% . In such cases, outliers may correspond to actual relication sites and may influence or skew the average values

to a great extent. To avoid this difficulty, we generate normalised histograms (with bin locations are at an interval of one pixel) for the motion values between every two consequtive time points . Figure 5.8 (b) shows the average frequency count of histograms for time point 1 to 10 plotted against the displacement values measured in pixels. Analysis of the results show that for 90% of the sites, the average movement is in the range of 2-6 pixels. We also measured the Hausdorff distance after allignment by rigid transformations to obtain a measure of the deformation of the sites from one time point to the next. It was found that Hausdorff distance was very small (in the range of .01 pixel unit) for about 98% of the chromatin domains (Fig 5.8 (c)). The results obtained are consistent with results of manual measurements done on same types of images. Fig 5.7 (a) and 5.7 (b) shows examples of the motion and shape recovery techniques applied on polygonal contours.

To illustrate the results of the last subsection and demonstrate the usage of the above mentioned ideas as a whole, we show an example of intermediate images generated by our technique. Figures 5.9(a) and 5.9(g) show the segmented images of chromatin domains in the nucleus of a living cell at time 0 sec and time 3 sec, which are the initial and final time points respectively. The images generated at intermediate time points 1 sec and 2 sec are shown in Figures 5.9(c) and 5.9(e). The algorithm can be used to generate continuous motion by generating as many intermediate images as desired. This can serve the objective of providing biologists with movies depicting the processes involving genomic structures taking place inside the cell nucleus in an online real time fashion.

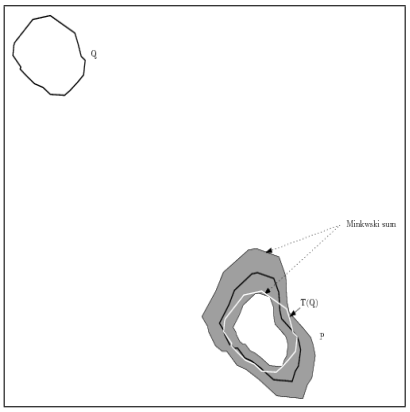
5.4 Conclusion

We have presented a novel geometric technique based approach for simplifying, segmenting and then recovering continuous motion, deformation and intensity surface for genomic structures in microscopic nuclear images of living cells. A key feature of our approach is to use normal distribution functions to capture the main structure of each nuclear image, to segment genomic structures from diffused signals, and to recover the intensity surfaces of the segmented genomic structures. Our technique considers both rigid and non-rigid motion and does not require detailed knowledge about the physics and mechanics of the chromatin domains considered. Also, our techniques have low time complexity which is particularly desirable for biological images involving a large amount of data.

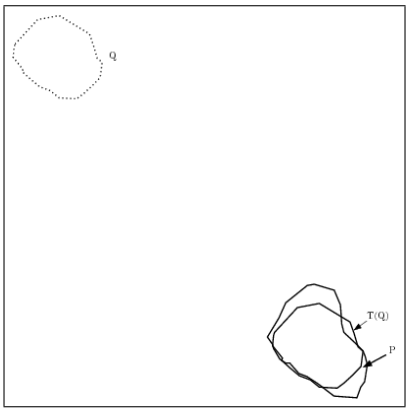


(a)

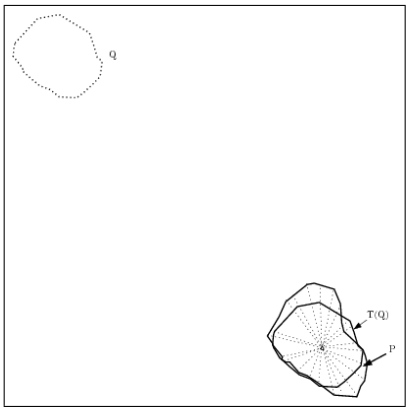
(b)



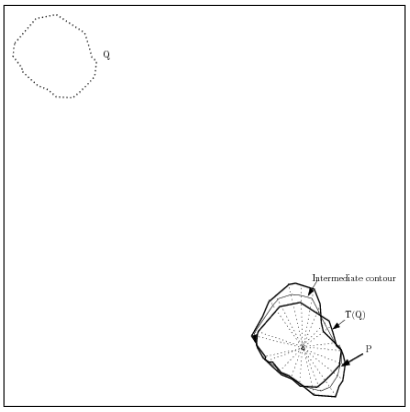
(c)



(d)

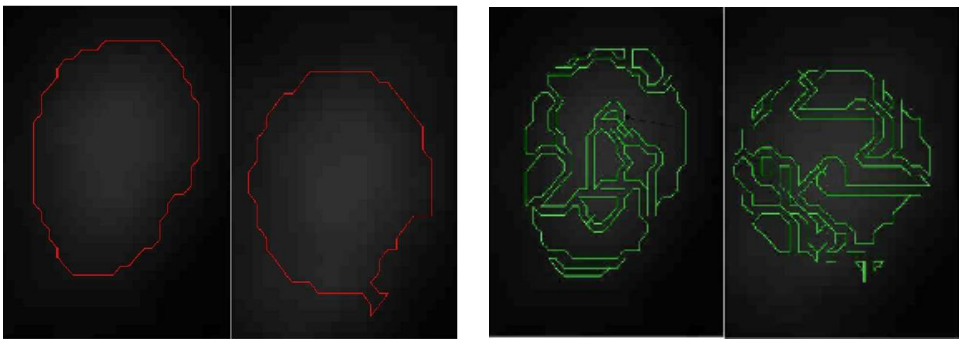


(e)



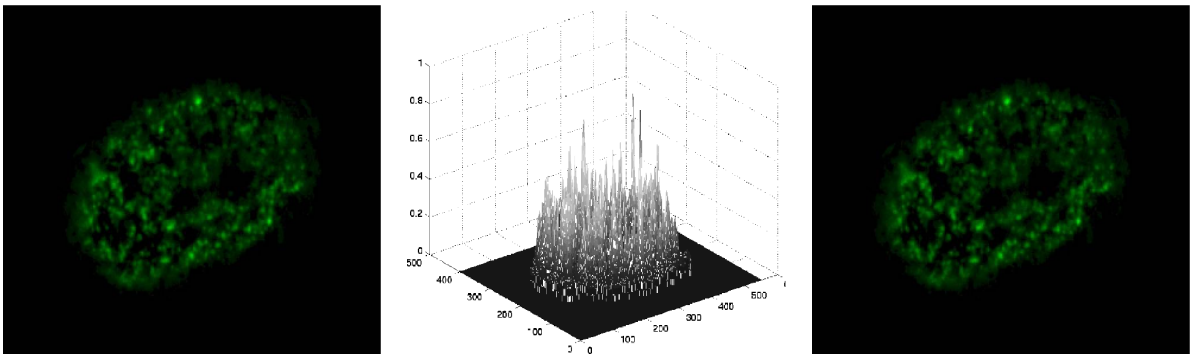
(f)

Figure 5.2: (a) Two corresponding polygons P and Q (b) Inner and outer boundary of the Minkowski sum of P and a square r (c)-(d) Matching under translation (e) Shape deformation from one central point and (f)intermediate shape generated



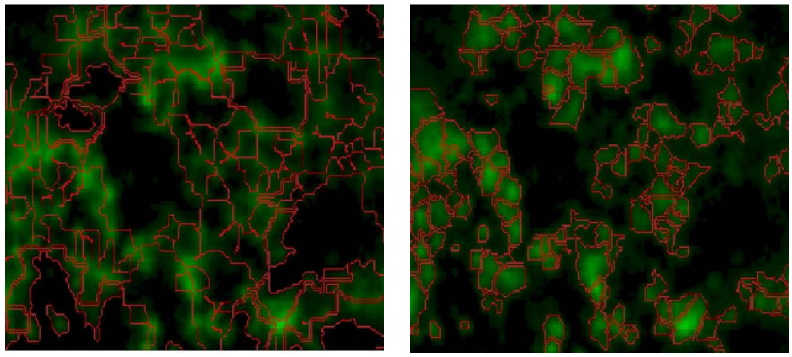
(a) (b)

Figure 5.3: (a) Two corresponding contours (b) Their normal distribution domains projected on a x-y plane.



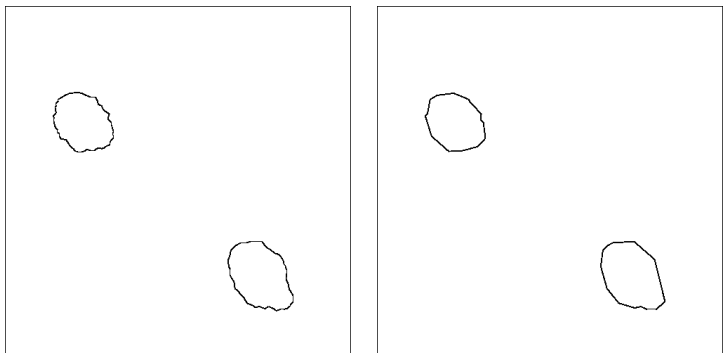
(*a*) (*b*) (*c*)

Figure 5.4: (a) Original cell nuclear image. (b) The corresponding intensity profile. (c) Approximated cell nuclear image.



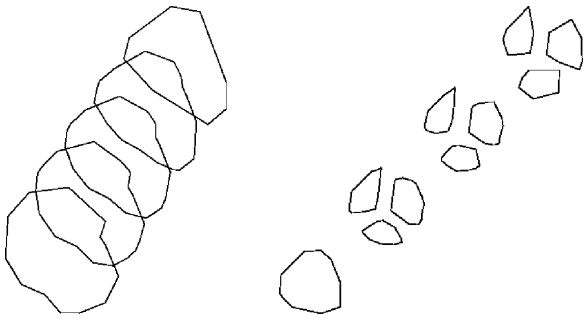
(*a*) (*b*)

Figure 5.5: (a) Foci segmented by watershed. (b) Foci segmented by our method.



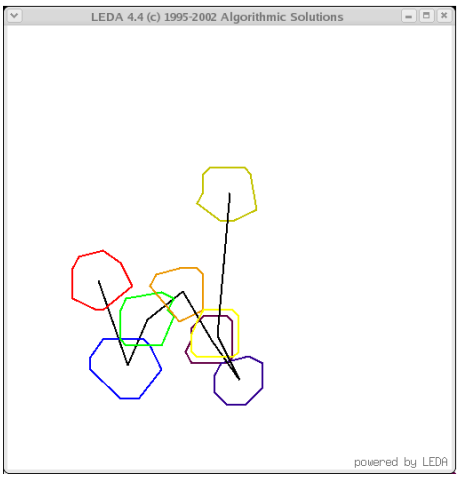
(a) (b)

Figure 5.6: Polygonal contours of the foci before and after contour simplification .

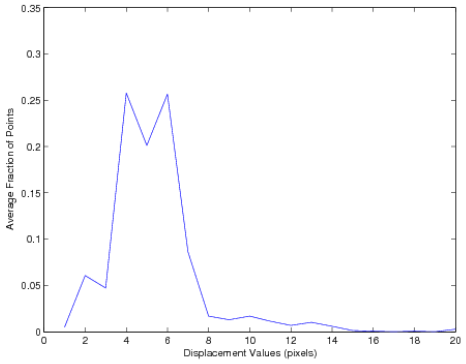


(a) (b)

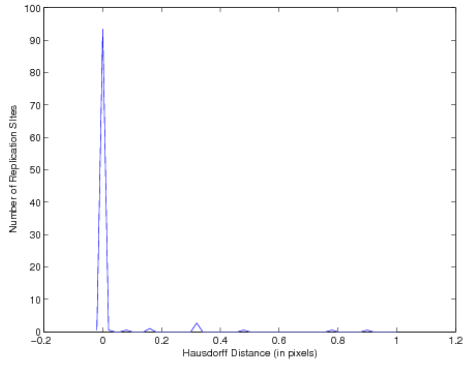
Figure 5.7: Shapes and positions of a deforming object at a sequence of intermediate time points in case of (a) one-to-one (b) one-to-three correspondence among polygons.



(a)



(b)



(c)

Figure 5.8: (a) Trajectory (in black) of a single site tracked over 8 consequtive time points. The replication site at each time point is shown in a different color and Histograms showing (b) Average movement and (c) Plot of Hausdorff Distance calculated as a function of the percentage of replication sites for images at an interval of 3 secs

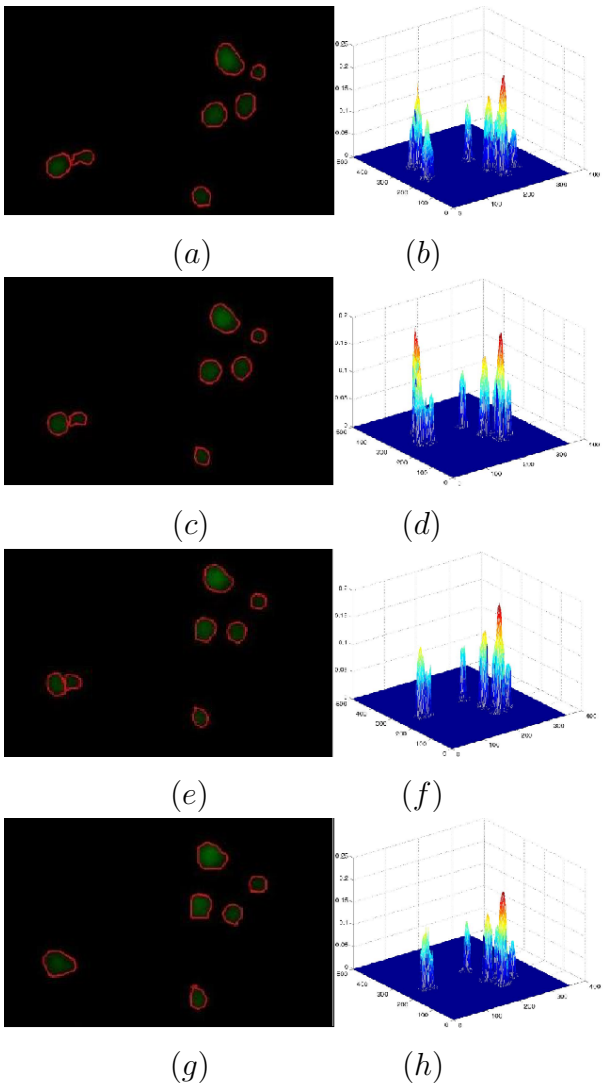


Figure 5.9: (a-b) Segmented image at 0 sec with intensity plot (c-d) Image generated at 1 sec with intensity plot (e-f) Image generated at 2 sec with intensity plot (g-h) Segmented image at 3 sec with intensity plot.

Chapter 6

Mobility Analysis of functional sites from time lapse microscopic image sequences of living cell nucleus

6.1 Problem Description and Previous Works

DNA replication is the process of copying a DNA strand in a cell prior to division and is among the most important processes inside the mammalian cell nucleus. These replication (or synthesis) processes occur at so called *replication sites* (RS, for short). The study of the basic properties of these sites is fascinating not only because it helps interpret the fundamental functions of a cell but may also lead to understanding variations between a cell's healthy and diseased states (such as cancer). Recent developments in microscopic imaging techniques[73; 83] have therefore focused on imaging RS in action in order to obtain a better understanding of their dynamics. The last few years in particular have seen several notable advances in *live* nuclear microscopic imaging and the development of associated software modules for image processing and analysis. This has been complemented by simultaneous developments in the usage of fluorescent proteins for staining these sites before imaging for better identification and visualization. These advances collectively have provided biologists for the first time a view of the spatial location, movement, and other behavioral aspects of RS within living cells in real time.

From a biologist's perspective, RS exhibit an extremely interesting characteristic in that they are constantly in motion. This interest stems partly from observations

reported in recent literature[9] which show a significantly faster movement of the sites encompassing the euchromatin DNA compared to those around the hetrochromatin DNA. Because it is well known that euchromatin is composed mostly of actively expressing genes whereas heterochromatin is largely inactive, the observed motion behavior strongly suggests a deeper connection between the sites' mobility patterns and the process of gene expression. These relationships can only be explored if the motion dynamics of these sites are well understood. The focus of our work is the design of algorithms that enables us to determine, study, and interpret these mobility patterns.

A closer analysis of the problem reveals several potential complications. For instance, a visual evaluation of the data suggests that the sites undergo an almost random motion between consecutive image frames. A temporal tracking algorithm adds little extra information towards any discernible patterns of movement except that it yields the sites' individual motion magnitudes from one image frame to the next. Moreover, details about the *kinds of motion dynamics* these sites undergo during various cell phases are not yet known. This makes the mathematical formulation of the problem challenging because we know neither the behavior of an ideal system (such as human vision/cognition) on this input nor the *kind of patterns* we would like to determine. Part of the reason that human vision also fails to extract any useful output from such data is that any intrinsic motion patterns are apparently irrecoverable from an overwhelming global to-and-fro-motion the sites exhibit. Given these difficulties, our objective is to search for some "method in madness" in an effort to represent the seemingly stochastic motion as a combination of a set of tractable functions.

In this chapter, we propose novel techniques to address the problems discussed above in an effort to study the mobility properties of the RS. We approach this goal with a two-fold objective. The first question we try to answer is whether given a set of points[1] apparently undergoing haphazard motion, are there small subsets of points that show *similar degree of motion*. To answer this question, we propose a technique based on high dimensional point set clustering for grouping sites based on their mobility properties. We then illustrate how we employ this idea to identify spatial patterns by splitting the nucleus into various almost-independent mobility zones. Our second independent goal is to identify sets of points that seem to be moving together, as if linked by a slightly deformable chain (sub-structure identification). Observe that the

[1] In the remainder of this chapter, we will use the terms *sites (RS)* and *points* interchangeably depending on the biological or geometric context of the discussion.

true motion pattern, say P, the sites undergo is unknown. Nonetheless, it is expected that if a group of points are indeed moving rigidly as a substructure, their mobility (high or low) should be same for each point in that group. Therefore, to evaluate our algorithms we compare the results obtained using our two techniques and observe that there is a high degree of territorial overlap between the rigid point sub-sets generated in the second step and the mobility zones determined from the first.

6.2 Background

We briefly discuss the physical aspects of image acquisition before going into the details of our algorithm. First, the sites are labelled by incorporating fluorochrome coupled nucleotides into the DNA of replicating cells. Upon completion of replication, this technique yields multiple fluorescent chromatin domains (CD). Fluorochrome labeled nucleotides are introduced by direct microinjection into cell nuclei and images are acquired in time-lapse mode two to three days after microinjection. The fate of nuclear functional domains can be followed in real time by fusing the Green Fluorescent Protein (GFP) to specific proteins involved in various cellular processes. In the case of DNA replication, PCNA (proliferating cell nuclear antigen) and Green Fluorescent Protein (GFP) are fused at the gene level. Mammalian cell nuclei, expressing these functional PCNA-GFP fusion molecules are imaged on an inverted epifloures-cent microscope. In both cases The microscope is equipped with a CCD camera and z-axis control for collection of optical section digital image sets. Images are then acquired as a sequence of $2D$ images at different resolutions (z-axis coordinates), $2-3$ seconds apart and a stack of $2D$ images is created for each time point. The images in the stack denote slices of the nucleus at different z coordinates. The contents of the nucleus i.e., the replication and transcription sites can span more than one optical sections in the stack. To compute the actual coordinates of the $3D$ points, maximal gray value regions in adjacent optical sections are matched by their area overlap and the control points are calculated as the geometric centers of the voxel regions.

6.3 Method

6.3.1 Temporal tracking of Replication Sites (RS)

The displacement (motion magnitude) of RS between consecutive image frames in the sequence can broadly be classified into two separate types of movement. The

first and arguably the more interesting type of movement is due to the individual motion of the sites. The second type of movement can be attributed to a nominal displacement of the complete nucleus from one image frame to the next. Before attempting to address (and determine) the patterns of movements of individual sites, we employ a simple preprocessing step to correct for the global displacement (an isometric transformation owing to the second type of movement), that the nucleus undergoes. This procedure also provides temporal tracking information that yields the point to point correspondences from a pair of images. While such a process is by no means *sufficient* towards our ultimate goal of motion analysis because it yields no information about the patterns of movement, nonetheless, it serves as a useful first step. For this purpose, we employ a simple technique proposed by Cho and Mount[24] to calculate an alignment between the pair of point sets representing RS in consecutive image frames. While the theoretical performance analysis of the algorithm guarantees an approximation ratio of 3 (alignment will be no worse than three times the unknown optimal alignment under Hausdorff distance measure), we observe that such an analysis is quite conservative. In practice, the technique performs favorably by calculating a transformation (rotation matrix, $\mathbf{R}$, and a translation vector, $\mathbf{t}$), that aligns the two point sets quite well. Once this alignment has been determined, we calculate point to point correspondences by using a combination of bipartite matching and rank-maximal matching coupled with certain assumptions about a neighborhood of motion. The size of the neighborhood is chosen based on the temporal resolution of the image sequence (usually, about 2 seconds) and the maximal to-and-fro motion a site can be expected to undergo within this time period. The results are then manually verified to ensure whether such a procedure returns accurate correspondences. In general, we obtain an accuracy of about 90% which is comparable to those reported in recent literature [22].

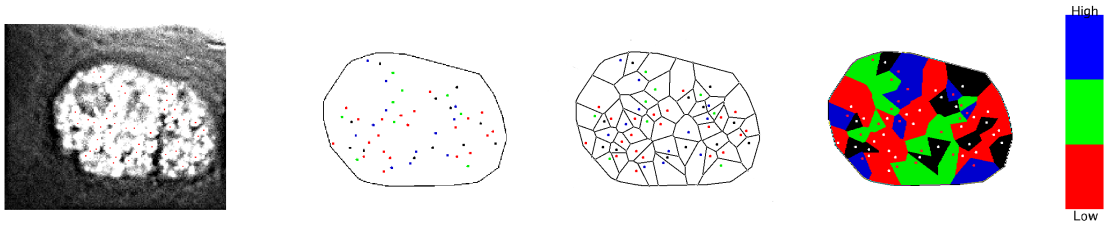
6.3.2 Determining Mobility Zones

To investigate the mobility properties of the sites, our main idea is to to zone (or cluster) the sites based on their motion magnitude and then use this information to spatially partition the image into mobility coded regions. We proceed as follows. The number of mobility zones (k) and the 'window' of time to be considered for determining the non-uniform motion of the sites are assumed given. Here, a fixed time window consists of d discrete intervals or image frames. The parameters k and d are then used to create (and populate) a high dimensional feature space S, where $S \in \Re^d$ if the left end of the time window is placed *on* (but not including)

time point t_0. S is populated by making use of the tracking information obtained in Section 6.3.1. Consider a graph, G, induced by the correspondence information determined by the temporal tracking algorithm. G has d 'levels' (or parts) where points in the image frame at time point, t_j, are represented as nodes in G at level j, say G_j. The information that a point p_i^j in the image at t_j corresponds to a point p_k^{j+1} can be easily represented in G by introducing a directed edge between the nodes corresponding to p_i^j and p_k^{j+1}. The weight of an edge from a node $n_i^j \in G_j$ to $n_k^{j+1} \in G_{j+1}$ can be calculated by considering the distance traveled by the point p_i^j from time point, t_j to t_{j+1} given by $\texttt{dist}(\cdot,\cdot)$ denoting the L_2 distance between p_i^j and p_k^{j+1}. Our feature space, S, can then be easily populated by considering each unique (not necessarily node disjoint) *correspondence path* from a node in G_1 to a node in G_d. The *inhomogenous* representation of a point in S can then be calculated by considering the edge weights (in order) of its corresponding correspondence path. Note that the those image points that could not be tracked will not have a complete correspondence path from G_1 to G_d and thus will not be represented in S.

Our purpose now is to determine k spatial clusters in S; each cluster will represent a 'mobility zone' and an inverse transformation on S will yield information about the mobility patterns of individual points in the input images. We based our algorithm on a hierarchical clustering technique known as *Agglomerative clustering* [50]. Hierarchical clustering algorithms work in arbitrary number of dimensions and group items into a hierarchy of clusters relying on distance measures to determine similarity between clusters. This cost function is then minimized in an optimization framework. Agglomerative clustering in particular works by considering each entity as an individual cluster, and then pairs of items are repeatedly merged until the total number of clusters in S is exactly k (similar to Kruskal's Minimum Spanning Tree algorithm). We employ a special type of agglomerative hierarchical clustering called CURE (Clustering Using Representatives) [42] that has a running time complexity of $O(n^2)$. It performs reliably, is robust with respect to the value of k, and produces high quality clusters even in the presence of outliers. Once the clustering is done, we assign to each cluster a unique 'color tag' as an identifier. An inverse transformation is then applied on S; this yields a color tag for each correspondence path in G. Clearly, if the points corresponding to two correspondence paths, P_1 and P_2, were assigned the same color (members of the same cluster in S), the colors tags for P_1 and P_2 will also be the same. The color tags of the correspondence paths are then transferred back to the sets of RS in the input image sequence. Observe that the clustering process uses *only* the motion parameters and is independent of the spatial

distribution of the points. However, *using* the clustering information, we can now analyze the sites' spatial properties in context of the motion they undergo and investigate if any interdependencies can be determined. We do this as follows. First, we partition the nucleus using Voronoi diagrams [34], dividing it into spatial regions of influence based on the distribution of sites. We then make use of the sites' color tags (obtained via an inverse mapping on S) to merge adjacent polygons having the same color tag to obtain larger cells obtaining a *chromatic* Voronoi Diagram. This final step provides us with a very useful tool for analyzing the spatial relationships of the RS. Notice that larger cells in the chromatic voronoi diagram are in fact a contiguous mobility sub-region and indicate a sub-structure of RS undergoing similar motion over the time window. By analyzing the chromatic voronoi diagrams for an entire sequence (all d image frames in a sequence), we can not only assess the reliability of the scheme (the partitioning should be fairly continuous over the given time window and should not change abruptly from one image frame to the next), but also analyze the motion dynamics of a given sequence of RS (see Fig. 6.1 for an illustration).
Remark: Our attempts at obtaining an equivalent clustering solution in $1D$ using the mean values of the feature vectors yielded significantly different results. We suspect that motion variability exhibited by a RS in d frames is not sufficiently captured by its mean velocity alone.



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

Figure 6.1: Illustration of (a) original cell nuclear image, (b) sites colored according to mobility based clustering, (c) Voronoi Partitions of the sites, (d) contiguous mobility zones determined by merging adjacent voronoi cells with the same color tag. The color scheme used to denote the zones is denoted by the color bar on the right. Regions containing sites which could not be tracked (about 10%) are shown in black.

6.3.3 Determining Rigid Substructures

Our strategy here is to determine subsets of points that approximately retain their substructure from one image frame to the next (and over the sequence). This is a

key difference to the previous motion-based clustering scheme because substructure similarity of the form we address here is *independent* of motion (displacement transformations such as translations and rotations). Similarity and agreement in the results of these independent techniques[2] on the same data set indicates a fair likelihood of accuracy in the absence of any gold standard data.

In recent literature, there has been a flurry of activity aimed at utilizing structural pattern identification techniques to study common substructures of proteins in bioinformatics research [101]. The common feather in most of these approaches is the representation of proteins as geometric graphs. Each node in the graph represents a 'unit' (or atom) in the protein and has real world coordinate values; the edges between the nodes represent the inter-atomic bonding. Each technique then tries to calculate a similarity measure between these graphs (or trees). Adapting these techniques to our problem, however, is rather difficult. because no apparent 'links' between the RS are known. Of course, an alternative is to look for patterns for every possible grouping of points. While this may work for a small number of points, our datasets are typically large. This would require a substantial amount of computation time making it practically infeasible.

Taking these issues into account, the approach we adopt is as follows. Consider two point sets, P_1 and P_2, corresponding to RS in consecutive image frames. For every triplet of point in the first set, say (p_i, p_j, p_k), a create a point p'_{ijk} in a search space, S, such that $p'_{ijk} = (d_{ij}, d_{ik}, d_{kj})$ where $d_{ij} = \texttt{dist}(p_i, p_j)$, $d_{ik} = \texttt{dist}(p_i, p_k)$, $d_{kj} = \texttt{dist}(p_k, p_j)$ and $d_{ij} \geq d_{ik} \geq d_{kj}$ ($\texttt{dist}(\cdot, \cdot)$ denotes the L_2 distance). Thus, we obtain two sets of points, P'_1 and P'_2 in S, each comprising of no more than n^3 points (if $\texttt{max}(|P_1|, |P_2|) = n$). The process is useful because it converts our substructure isomorphism determination problem into a point location problem as we shall discuss now. Consider a triplet of points that did not undergo *any* structural change from the first time point to the next. The reason why this transformation to a point location problem works is that the triplet of points at the time point t_{P_1}, say, (u_i, u_j, u_k), and the triplet of points at time point t_{P_2}, say, (v_x, v_y, v_z), will map to the *same* point in S. In other words, the notion of a ϵ-neighborhood in S is analogous to a quasi isomorphism between triplets of RS in consecutive image frames (where ϵ is a small constant). To exploit this neighborhood information, we make use of range-search trees[34] for point location queries in S. However, since the transformation to S loses all location information of triplets of RS, we check potentially isomorphic structures

[2]The previous technique performs clustering in a feature space based solely on motion while the second technique will consider 'spatial structures' and ignore motion.

against the tracking information calculated in Section 6.3.1 and verify if a given pair of two triplet of points are isomorphic under some rigid transformation T. Then, we gradually increase the sets of likely matches to include more than three points at a time. This strategy allows us to avoid paying an exorbitant cost associated with checking all combinations of $\binom{n}{r}$ point sets in consecutive image frames (where $r \geq 3$). Finally, the process yields isomorphic substructures in consecutive image frames.

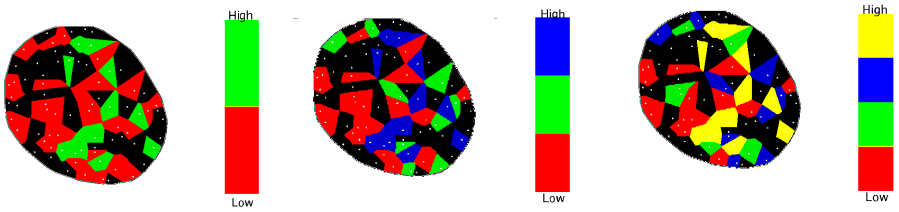
6.4 Results

Our algorithms were implemented in $C++$ using CGAL and LEDA on a machine running GNU/Linux. Evaluations were performed on 15 temporal data-sets ($2D$ + time), each consisting of about 10 image frames taken 2 seconds apart. For simplicity of presentation, we will discuss results for each subsection in the order they appeared in the chapter. Due to space limitations, we will omit details about the preprocessing phase. We will try to highlight our main observations here using a few datasets as illustrative examples instead of discussing results for each set individually.

Motion based high dimensional clustering: The correspondence information from Section 6.3.1 was obtained and our algorithms described in Section 6.3.2 were used to determine clusters using user provided parameter k and d based on the sites' motion magnitudes. The color tags of each of the k motion clusters were then used to 'tag' its member RS. A visualization of these color-coded sites in real space revealed a rather interesting property. We found that a significant number of spatially proximate sites (in real space) were allocated to the same cluster in the high dimensional feature space. Further, a partition of the nucleus (based on Voronoi Diagrams) showed a number of adjacent cells joining together to form contiguous mobility zones. This observation indicates that close-together sites show a similar degree of motion as previously illustrated in Fig. 6.1.

The robustness of our algorithm was further evaluated by varying the two input parameters: (1) value of k (number of clusters), (b) length of time window (dimensionality of the feature space, d). **(1) varying the value of k.** We noticed that the spatial correlations of sites that belong to the same mobility cluster in feature space are preserved under varying values of k. In Figs. 6.2 (a)-(c), we illustrate the results of color partitioning of the nucleus (represented by the first image frame in the sequence) based on mobility clustering for different k values. Observe that between Fig. 6.2 (a) and 6.2 (b), the overall partitioning schema remains approximately constant. However, based on the number of clusters desired (the chosen value of k), the higher

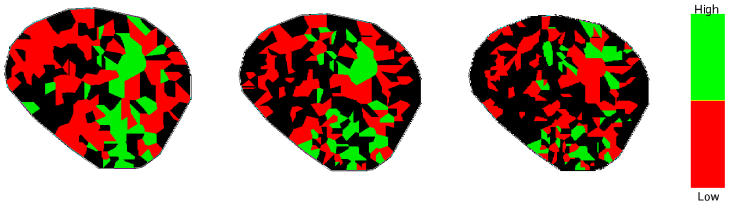
mobility zone in (a) was upgraded to being the highest mobility zone in (b) whereas the lower mobility zone of (a) showed more variation in movement splitting up into two zones. A similar phenomena is seen when we increase k from 3 to 4 in Figs. 6.2 (b) and (c). This trend conveys a strong sense in that the clustering technique is robust to the value of k. Therefore, one can increase the value of k if a more vivid detail is desired (more colors in the partition). **(2) increasing the length of the**



k=2 k=3 k=4

Figure 6.2: Nucleus of a cell showing varying number of mobility zone.

time window. We observed that a lengthier time window (higher dimensionality, d, of feature space) does not have a detrimental effect on the performance of the clustering. In fact, spatial relationships of the clusters determined for several different choices of time windows were very similar (for a fixed k value). We illustrate these results in Figs. 6.3 (a)-(c). The only deterioration we could notice was that as the length of the time window became progressively larger, the number of sites that could not be tracked also increased (showing more black regions in Figs. 6.3 (a)-(c)). We suspect that this might be as a consequence of inaccuracies in the temporal tracking (preprocessing) phase. **(3) Overall quality of clusters.** The overall quality of



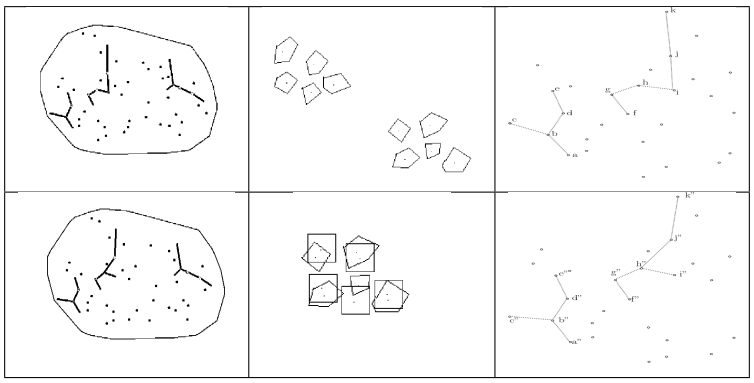
d=4 d=6 d=8

Figure 6.3: Nucleus of a cell showing mobility zones determined from time point 0 to 4, time point 0 to 6 and time point 0 to 8.

mobility based clustering was observed to be quite good (calculated using metrics suggested in [42] and other simple measures such as geometric standard deviations with respect to the centroid of the cluster). While we will avoid a detailed discussion on this topic due to lack of space but would like to point out that the standard

deviation of the clusters remained relatively constant as a function of an increase in the length of the time window (related to (2) above). However, we noticed an almost linear relationship between the compactness and the value of k chosen (related to (1) above).

Rigid substructure determination: The algorithm described in Section 6.3.3 were used to determine rigid substructures using parameters d and a maximum allowable deformation ϵ. Fig. 6.4 (a) shows some of rigid substructures obtained for a particular sequence of images where $d = 5$ (time point 0 to 5). Rigid substructures of points identified by our technique are illustrated as a minimum spanning tree. Small deformative changes in the rigid substructures across the time sequence manifest as variations in the spanning tree structure from one time point to the next (see Fig. 6.4). For example, in 6.4 (c) (which shows an enlarged view of one of the rigid substructures in Fig. 6.4 (a) with correspondence of points), point labeled j was connected to point i in time point 0 but its corresponding point j'' in time point 3 was connected to point h'', indicating that it moved closer to point h in the course of the time sequence, even though the overall substructure remained the same. **Agreement:** To evaluate the agreement of the two techniques, we superimposed the rigid substructures obtained using the algorithms in Section 6.3.3 'on' the voronoi regions generated based on mobility based clustering. The results obtained on a sequence are shown in Fig. 6.4(a) and (c) whereas the superimposed image is shown in Fig. 6.4(b). In all datasets, we observed that the rigid substructures belong almost entirely to the same zone.



(a) (b) (c)

Figure 6.4: A cell nucleus showing (a) rigid substructures (in a spanning tree form), (b) superimposed with mobility zones (color scheme same as in Fig. 6.3), (c) enlarged view of a sub-region for time point 0 (top) and time point 3 (bottom).

6.5 Conclusions

We have proposed several algorithms for determination and analysis of motion dynamics of functional sites. While the subproblems we address such as mobility based clustering and determination of quasi-isomorphic substructures are interesting in their own right, we are particularly excited by the 'motion patterns' in the movement of RS discovered by the technique. These results dispense (at least in part) the notion about the randomness of such movements. In fact, the strong correlation between spatial proximity and the degree of motion of the sites, coupled with presence of rigid substructures suggests that the movement of replication sites over small regions inside the nucleus are either inter-dependent or the sites are influenced equally by some unknown factor.

Chapter 7

Conclusion

In the last several years, there has been a significant interest in the development and application of computational techniques to understand complex biological phenomena. The community has successfully developed and applied sophisticated tools to answer many complex questions and solve computationally challenging problems. The ambitious Human Genome project completed recently would not be viable otherwise, and this recent success indicates that we will see continued synergy between computation and biomedical research.

Within biological research, the need for advanced computational tools is strongly felt in microscopic image analysis due to the volume of the data and the complex nature of questions that need to be answered. Images generated by microscopic devices are of large sizes, noisy, and difficult to interpret directly. Nonetheless, they contain information that if correctly interpreted offers an unique insight into the world of microscopic structures. We focus on several such problems motivated from this interpretation task. First, we propose a new segmentation algorithm for segmenting 3D image stacks called the cosegmentation problem. In the next chapter, we propose efficient techniques for the Generalized Median Graph problem. In Chapter 4, we study a special case of partial shape matching problem. Next, we design a suite of algorithms for determining motion and deformation of object form time lapse microscopic image sequences. Finally, we present methods to determine, analyze and interpret mobility pattern of functional sites in the nucleus. While these problems are motivated from biological imaging, we believe that our ideas and approaches are general enough to be useful in problems from other applications. For example, the cosegmentation problem can be used to segment stereo image pairs, and for multimodality segmentation of biomedical images. The median graph problem, as we demonstrate in Chapter 3 can be applied to applications in drug design the algorithms

for partial shape matching in Chapter 4 are well suited for applications in information retrieval. Finally, the problems in Chapters **??** have applications in motion tracking and registration problems in other settings. In the next section, we summarize our contributions and outline future directions for the ideas explored in this thesis.

7.1 Cosegmentation Problem

In Chapter 2, we study the cosegmentation problem where the goal is to segment the same object (i.e., region) from a pair (or more) of images. The segmentation problem is formulated as a partition of both images into foreground and background, with the additional constraint that the histogram of the areas classified as foreground should be similar. The main contribution of our approach is to show that defining the similarity in the L_2 (instead of L_1) sense gives some nice structure to the constraint matrix of the corresponding optimization model. We prove some interesting properties of the constraint matrix of cosegmentation LP, specifically that the optimal LP solution will only have values in $\{0, \frac{1}{2}, 1\}$. Half integrality leads to a simple rounding strategy that gives good segmentations in practice and also allows an approximation analysis under some conditions. A direction of future research will be to investigate if a faster algorithm can be designed by employing a more combinatorial approach for the same model. Recent results by Hochbaum has shown it is possible to devise a min-cut/max-flow formulations for optimization models which have half integrality properties. It will be interesting to see if such a formulation is possible for the cosegmentation problem.

7.2 Generalized Median Graphs

In Chapter 3, we study the Generalized Median graph problem where the task is to to construct a prototype or a 'model' from an input set of graphs. The problem effectively captures many object recognition and learning problems as well, where graphs are increasingly being adopted as a powerful representation tool. Existing techniques for this problem are evolutionary search based; we proposed a polynomial time algorithm based on a linear programming formulation. Within this new framework, one can optimize edit distance functions that capture similarity by considering vertex labels as well as the graph structure simultaneously. These methods have provided us with the tools necessary to build a topological map of all 23 pairs of the human chromosome.

Notice that notion of Median Graphs has a natural connection to graph entropy. Graph entropy is an information theoretic functional on a graph with a probability distribution on its vertex set (or edge set). In essence, it is a measure of degree of uncertainty in a graph. If we are able to quantify the degree of uncertainties in a set of graph, it can be leveraged to determine the substructures of low entropy common in all graphs and build a median for the set. Therefore, it may be interesting to explore if such an information theoretic algorithm can be designed for the median graph problem.

7.3 Fitting Polygonal Regions for matching 3D polyhedra

Matching geometric objects is a fundamental problem in computational geometry with several applications in biological imaging. In Chapter 4, we study an important partial matching problem from biological imaging. The input is in the form of sets of under-sampled slices of one (or more) unknown $3D$ objects, possibly generated by slicing planes of arbitrary orientations, the question we are interested in is whether it is 'possible' that two under-sampled sets have been taken from the same object. Alternatively, can we determine with 'certainty' that the given input samples cannot be from the same object. We present efficient algorithms for addressing these questions. Our algorithm is based on interesting geometric techniques and enables answering these queries either as plausible or a certain negative.

7.4 Determining Dynamics of Chromatin Domains

In Chapter 5, we propose new approaches for analyzing time-lapse microscopic nuclear images. Our techniques are mainly designed to address the problem of limited spatial and temporal resolution capacities of current microscopic imaging techniques which allow acquisition of images only in time-lase mode. This leads to significant loss of information from one frame to the next. Such data, if available, can be extremely helpful in the study of nuclear organization and function. We present a suite of geometric technique based approaches for solving the problem. Our techniques, working together, can simplify the raw images, segment them and then effectively recover complicated motion and deformation as well as the change of intensity surfaces from pairs of images in a microscopic image sequence.

7.5 Mobility Analysis of Functional Sites

In Chapter 6, we proposed a suite of novel techniques to determine, analyze, and interpret the mobility patterns of functional sites. Our algorithms are based on interesting ideas from theoretical computer science and database theory and provide for the first time the tools to interpret the seemingly stochastic motion patterns of the functional sites within nucleus in terms of a set of tractable 'patterns' which can then be analyzed to understand to derive quantitative information.

References

[1] IBM Frequent Subgraph Miner (http://www.alphaworks.ibm.com/tech/fsm).

[2] H. A. Almohamad and S. O. Duffuaa. A linear programming approach for the weighted graph matching problem. *Pattern Anal. Mach. Intell.*, 15(5):522–525, 1993.

[3] Helmut Alt and Leonidas J. Guibas. Discrete geometric shapes: Matching, interpolation, and approximation A survey. Technical Report B 96-11, 1996.

[4] M. Araujo, B. Renata, Caetano Traina Jr., A. J. Traina, M. Josiane, and Humberto Razente. Extending relational databases to support content-based retrieval of medical images. In *Proceedings of IEEE International Conference on Computer Based Medical Systems*, pages 303–308, 2002.

[5] M. Atallah. A linear time algorithm for the hausdorff-distance between convex polygons. *Information Processing Letters*, 17:207–209, 1982.

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

[7] S. Belongie, J. Malik, and J. Puzicha. Shape matching and object recognition using shape contexts. In *IEEE Pattern Analysis and Machine Intelligence*, volume 24, pages 509–522, 2002.

[8] R. Berezney, K. S. Malyavantham, A. Pliss, S. Bhattacharya, and R. Acharya. Spatio-temporal dynamics of genomic organization and function in the mammalian cell nucleus. *Advances in Enzyme Regulation*, 45:17–26, 2005.

[9] R. Berezney, K. S. Malyavantham, A. Pliss, S. Bhattacharya, and R. Acharya. Spatio-temporal dynamics of genomic organization and function in the mammalian cell nucleus. *Advances in Enzyme Regulation*, 45:17–26, 2005.

[10] R. Berezney, M.J. Mortillaro, H. Ma, X. Wei, and J. Samarabandhu. The nuclear matrix: a structural milieu for genomic function. In *Int. Rev Cytol*, volume 162A, pages 1–65, 1995.

[11] R. Berezney and X. Wei. The new paradigm: integrating genomic function and the nuclear architecture. In *J. Cell Biochem Suppl*, volume 31, pages 238–342, 1998.

[12] H. L. Boblaender. Polynomial algorithms for graph isomorphism and chromatic index on partial k-trees. *J. Algorithms*, 11(4):631–643, 1990.

[13] E. Borenstein, E. Sharon, and S. Ullman. Combining top-down and bottom-up segmentation. In *Proc. of IEEE Conference on Computer Vision and Pattern Recognition*, pages 46–46, 2004.

[14] E. Borenstein and S. Ullman. Learning to segment. In *Proc. of European Conference on Computer Vision*, pages 315–328, 2004.

[15] R. S. Boyer and J.S. Moore. A fast string searching algorithm. *Communications of ACM*, 20:762–772, 1977.

[16] Y. Boykov, O. Veksler, and R. Zabih. Fast approximate energy minimization via graph cuts. *Pattern Anal. Mach. Intell.*, 23(11):1222–1239, 2001.

[17] S. Boyle, S. Gilchrist, J.M. Bridger, N. L. Mahy, J. A. Ellis, and W. A. Bickmore. The spatial organization of human chromosomes within the nuclei of normal and emerin-mutant cells. *Hum. Mol. Genet.*, 10:211–219, 2001.

[18] U. Brandes, M. Gaertler, and D. Wagner. Experiments on graph clustering algorithms. In *Proc. European Symposium on Algorithms*, pages 568–579, 2003.

[19] R. Brunelli and O. Mich. On the use of histograms for image retrieval. In *Proceedings of IEEE International Conference on Multimedia, Computing and Systems*, 1999.

[20] C. Chen, H. Li, X. Zhou, and S. T. C. Wong. Graph cut based active contour for automated cellular image segmentation in high throughput rna interface (rnai) screening. In *ISBI*, pages 69–72, 2007.

[21] R. Chen and Z. Weng. A novel shape complementarity scoring function for protein-protein docking. *Proteins: Structure, Function and Genetics*, 51(3):397–408, 2003.

[22] X. Chen, X. Zhou, and S.T.C. Wong. Automated segmentation, classification, and tracking of cancer cell nuclei in time lapse microscopy. *IEEE Transactions on Biomedical Engineering*, 53(4):762–766, 2006.

[23] D. S. Cheng and Mario A. T. Figueiredo. Cosegmentation for image sequences. In *Proc. of International Conference on Image Analysis and Processing*, pages 635–640, 2007.

[24] M. Cho and D. Mount. Improved approximation bounds for planar point pattern matching. In *Proc. Workshop on Algorithms and Data Structures (WADS)*, pages 432–443, 2005.

[25] V. Choi and N. Goyal. A combinatorial shape matching algorithm for rigid protein docking. In *Proceedings of The Fifteenth Annual Symposium on Combinatorial Pattern Matching (CPM 2004)*, pages 285–296. Springer-Verlag Berlin, 2004.

[26] L. Coelho and M. F. M. Campos. Similarity-based versus template matching-based methodologies for image alignment of polyhedral-like objects under noisy conditions. In *Proceedings of Brazilian Symposium on Computer Graphics and Image Processing*, pages 155–162, 1997.

[27] Active contours for the movement and motility analysis ofbiological objects. V. meas-yedid and j. c. olivo-marin. In *International Conference on Image Processing*, volume 1, pages 196 – 199, 2000.

[28] D. J. Cook and L. B. Holder. *Mining Graph Data*, chapter 14. Wiley, New Jersey, 2006.

[29] T. Cootes, C. Taylor, D. Cooper, and J. Graham. Active shape models – their training and applications. *Computer Vision and Image Understanding*, 61(1):38 – 59, 1995.

[30] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to Algorithms*.

[31] D. G. Corneil and C. C. Gotlieb. An efficient algorithm for graph isomorphism. *J. ACM*, 17(1):51–64, 1970.

[32] T. Cremer and C. Cremer. Chromosome territories, nuclear architecture and gene regulation in mamalian cells. *Nature Reviews Genetics*, 2:292–301, 2001.

[33] D. Terzopoulos. D. Metaxas. Shape and nonrigid motion estimation through physics-based synthesis. In *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 580–591, 1993.

[34] Mark de Berg, Otfried Schwarzkopf, Marc van Kreveld, and Mark Overmars. *Computational Geometry: Algorithms and Applications*. Springer-Verlag, 2000.

[35] Regina Estkowski and Joseph S. B. Mitchell. Simplifying a polygonal subdivision while keeping it simple. In *Symposium on Computational Geometry*, pages 40–49, 2001.

[36] E.Welzl. Smallest enclosing disks (balls and ellipsoids. *New Results and New Trends in Computer Science, Lecture Notes in Computer Science*, 555:359–370, 1991.

[37] Andreas Fabri, Geert-Jan Giezeman, Lutz Kettner, Stefan Schirra, and Sven Schonherr. *On the Design of CGAL, the Computational Geometry Algorithms Library*. INRIA Sophia-Antipolis, 1999.

[38] P. W. Finn, S. Muggleton, D. Page, and A. Srinivasan. Pharmacophore discovery using the inductive logic programming system PROGOL. *Machine Learning*, 30(2-3):241–270, 1998.

[39] P. Foggia, C. Sansone, and M. Vento. A Database of Graphs for Isomorphism and Sub-Graph Isomorphism Benchmarking. In *Proc. IAPR Workshop on Graph-Based Repr.*, 2001.

[40] D. Genery. Visual tracking of known three-dimentional objects. *International Journal of Computer Vision*, pages 243–370, 1992.

[41] T. Granlund. *The GNU Multiple Precision Arithmetic Library*. GNU, 1996.

[42] S. Guha, R. Rastogi, and K. Shim. CURE: An efficient clustering algorithm for large databases. In *In Proc. of ACM SIGMOD Int'l Conf. on Management of Data*, pages 73–84, 1998.

[43] B. Behrends H. Alt and J. Blomer. Approximate matching of polygonal shapes. In *Ann. Math. Artif. Intell.*, volume 13, pages 251–266, 1995.

[44] H. Tanabe et al. Evolutionary conservation of chromosome territory arrangements in cell nuclei from higher primates. *Proc. of the National Academy of Sciences*, 99(7):4424–4429, 2002.

[45] A. Hlaoui and S. Wang. Median graph computation for graph clustering. *Soft Comp.*, 10(1):47–53, 2006.

[46] D. S. Hochbaum. An efficient algorithm for image segmentation, markov random fields and related problems. *Journal of the ACM*, 48(2):686 – 701, 2001.

[47] D. S. Hochbaum, N. Megiddo, J. Naor, and A. Tamir. Tight bounds and 2-approximation algorithms for integer programs with two variables per inequality. *Math. Programming*, 62(1):69–83, 1993.

[48] T. S. Huang. Modelling, analysis and visualization of non-rigid object models. In *IEEE 10th International Conference on Pattern Recognition.*, pages 361–264, 1990.

[49] B. Jahne. *Digital Image Processing*. Springer.

[50] A. K. Jain, M. N. Murty, and P. J. Flynn. Data clustering: a review. *ACM Comput. Surv.*, 31(3):264–323, 1999.

[51] X. Jiang and H. Bunke. Optimal lower bound for generalized median problems in metric space. In *Proc. of IAPR Structural, Syntactic, and Statistical Patt. Recog.*, pages 143–152, 2002.

[52] X. Jiang, A. Munger, and H. Bunke. On median graphs: Properties, algorithms, and applications. *Pattern Anal. Mach. Intell.*, 23(10):1144–1151, 2001.

[53] D. Justice and A. Hero. A binary linear programming formulation of the graph edit distance. *Pattern Anal. Mach. Intell.*, 28(8):1200–1214, 2006.

[54] Michael Kazhdan, Thomas Funkhouser, and Szymon Rusinkiewicz. Shape matching and anisotropy. *ACM Trans. Graph.*, 23(3):623–629, 2004.

[55] Donald E. Knuth, James H. Morris, and Vaughan R. Pratt. Fast pattern matching in strings. *SIAM Journal on Computing*, 6(2):323–350, 1977.

[56] B. Korte and J. Vygen. *Combinatorial Optimization: Theory and Algorithms*, page 101. Birkhäuser.

[57] B. Kotnyek. *A generalization of totally unimodular and network matrices*. PhD thesis, London School of Economics and Political Science, 2002.

[58] R. Kumar, K. Dana, P. Anandan, N. Okamoto, J. Bergen, P. Hemler, T. Sumanaweera, P. van den Elsen, and J. Adler. Frameless registration of MR and CT 3D volumetric data sets. In *Proceedings of the Second IEEE Workshop on Applications of Computer Vision, Los Alamitos, Calif.*, pages 240–248. IEEE Computer Society Press., 1994.

[59] C. Kambhamettu L. Zhou. Hierarchial structure and nonrigid motion recovery from 2d monocular views. In *CVPR' 00*, volume 2, pages 752–759, 2000.

[60] W. Lorensen and H. E. Cline. Marching cubes : A high resolution 3d surface construction algorithm. In *ACM SIGGRAPH*, 1987.

[61] Robin Lougee-Heimer. The common optimization interface for operations research. *IBM Journal of Research and Development*, 47(1):57–66, 2003.

[62] D. Lowe. Fitting parameterized three-dimentional models to images. In *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 441–450, 1991.

[63] P. Mahé, L. Ralaivola, V. Stoven, and J. Vert. The pharmacophore kernel for virtual screening with support vector machines. *J. Chem. Inf. Model.*, 46(5):2003–2014, 2006.

[64] K. Mehlhorn, S. Naher, M. Seel, and C. Uhrig. *The LEDA User Manual Version 3.8*. Max-Planck-Insitute for Informatik, 66123 Saarbrucken, Germany, 1999.

[65] F. Meyer. Image simplification filters for segmentation. *Journal of Mathematical Imaging and Vision*, 20:59–72, 2004.

[66] S. Muggleton and L. De Raedt. Inductive Logic Programming: Theory and Methods. *J. Logic Prog.*, 19(20):629–679, 1994.

[67] L. Mukherjee, V. Singh, J. Xu, K. S. Malyavantham, and R. Berezney. On mobility analysis of functional sites from time lapse microscopic image sequences of living cell nucleus. In *Proc. Medical Image Computing and Computer-Assisted Intervention*, pages 577–585, 2006.

[68] Lopamudra Mukherjee, Vikas Singh, Jinhui Xu, and Ronald Berezney. Fitting polygonal regions for matching 3d polyhedra. In Longin Jan Latecki, David M. Mount, and Angela Y. Wu, editors, *Vision Geometry XIV*, volume 6066, page 606607. SPIE, 2006.

[69] Lopamudra Mukherjee, Jinhui Xu, Artem Pliss, and Ronald Berezney. Efficient algorithms for motion and deformation recovery with biological applications. In *IEEE Systems, Man and Cybernetics*, 2004.

[70] S. Muthukrishnan, Viswanath Poosala, and Torsten Suel. On rectangular partitionings in two dimensions: Algorithms, complexity, and applications. *Lecture Notes in Computer Science*, 1540:236–256, 1999.

[71] R. Nagele, T. Freeman, L. McMorrow, and H. Lee. Precise spatial positioning of chromosomes during prometaphase: Evidence for chromosomal order. *Science*, 270:1830–1835, 1998.

[72] R. Nagele, T. Freeman, L. McMorrow, Z. Thomson, K. Kitson-Wind, and Hy Lee. Chromosomes exhibit preferential positioning in quiescent human cells. *J. Cell Biol.*, 112:525–535, 1998.

[73] B. Neumann, M. Held, U. Liebel, H. Erflea, P. Rogers, R. Pepperkok, and J. Ellenberg. High-throughput rnai screening by time-lapse imaging of live human cells. *Nature Methods*, 3:385–390, 2006.

[74] R. T. Ng and J. Han. Efficient and effective clustering methods for spatial data mining. In *Proc. Conf. on Very Large Data Bases*, pages 144–155, 1994.

[75] Hirobumi Nishida. Model-based shape matching with structural feature grouping. *IEEE Trans. Pattern Anal. Mach. Intell.*, 17(3):315–320, 1995.

[76] Kyoungju Park, April Nowell, and Dimitris Metaxas. Deformable model based shape analysis and recovery: Stone tool application. In *Proceedings of Applications of Computer Vision in Archaeology (ACVA)*, 2003.

[77] P.Berman, B.Dasgupta, and S.Muthukrishna. Exact size of binary space partitionings and improved rectangle tiling algorithms. *SIAM J.Discrete Math*, 15:252–267, 2002.

[78] C. A. Pelizzari, G. T. Y. Ghen, D. R. Spelbring, R. R. Weichselbaum, and C. T. Chen. Accurate three-dimensional registration of CT, PET and/or MR images of the brain. *J Comput. Assist. Tomogr.*, 13(1):20–26, 1989.

[79] C. A. Phillips and T. Warnow. The Asymmetric Median Tree: A New Model for Building Consensus Trees. *Discrete Appl. Math.*, 71(1-3):311–335, 1996.

[80] Artem Pliss. Chromatin dynamics is correlated with its replication timing. In *Faseb Summer Research Conferences*, 2005.

[81] J. P. W. Pluim, J. B. A. Maintz, and M. A. Viergever. Mutual-information-based registration of medical images: a survey. *IEEE Trans. on Medical Imaging*, 22:986–1004, 2003.

[82] M. Radermacher. *Three dimensional reconstruction of single particle in electron microscopy*. CRC Press, 1992.

[83] E. Rebollo and C. Gonzalez. Time-lapse imaging of male meiosis by phase-contrast and fluorescence microscopy. *Cell Biology and Biophysics*, 247:77–87, 2004.

[84] T. Riklin-Raviv, N. Kiryati, and N. Sochen. Prior-based segmentation by projective registration and level sets. In *Proc. of IEEE International Conference on Computer Vision*, pages 204–211, 2005.

[85] C. Rother, T. Minka, A. Blake, and V. Kolmogorov. Cosegmentation of image pairs by histogram matching – incorporating a global constraint into MRFs. In *Proc. of IEEE Conference on Computer Vision and Pattern Recognition*, pages 993–1000, 2006.

[86] A.J. Siegel S. Somanathan, T.M. Suchyna and R. Berezney. Targeting of pcna to sites of dna replication in the mammalian cell nucleus. *Journal of Cellular Biochemistry*, 81:56–67, 2001.

[87] Suleyman Cenk Sahinalp and Uzi Vishkin. Efficient approximate and dynamic matching of patterns using a labeling paradigm (extended abstract). In *IEEE Symposium on Foundations of Computer Science*, pages 320–328, 1996.

[88] Dietmar Saupe and Dejan V. Vranic. 3D model retrieval with spherical harmonics and moments. In *Proceedings of the 23rd DAGM-Symposium on Pattern Recognition*, pages 392–397. Springer-Verlag, 2001.

[89] N. Petkov S.E. Grigorescu and P. Kruizinga. Comparison of texture features based on gabor filters. *IEEE Trans. on Image Processing*, 11(10):1160–1167, 2002.

[90] S. C. Shapiro and W. J. Rapaport. The SNePS family. *Comp. & Math. Appl.*, page 243, 1992.

[91] J. Shi and J. Malik. Normalized cuts and image segmentation. *Pattern Anal. Mach. Intell.*, 22(8):888–905, 2000.

[92] P. Shilane, M. Kazhdan, P. Min, and T. Funkhouser. The Princeton Shape Benchmark, 2004.

[93] K. Siddiqi, A. Shokoufandeh, S. J. Dickinson, and S. W. Zucker. Shock graphs and shape matching. In *Proc. International Conf. on Computer Vision*, pages 222–229, 1998.

[94] Milan Sonka, Vaclav Hlavac, and Roger Boyle. *Image Processing: Analysis and Machine Vision*. O'Reilly, 1999. SON m 99:1 1.Ex.

[95] S. Toda. Graph Isomorphism: Its Complexity and Algorithms. In *Proc. Conf. on Found. of Soft. Tech. and Theoretical Computer Science*, page 341, 1999.

[96] A. Toshev, J. Shi, and K. Daniilidis. Image matching via saliency region correspondences. In *Proc. of IEEE Conference on Computer Vision and Pattern Recognition*, 2007.

[97] W. Tvarusko, M. Bentele, T. Mistelli, R. Rudolf, C. Kaether, D. L. Spector, H. H. Gerdes, and R. Eils. Time-resolved analysis and visualization of dynamic processes in living cells. *Proceedings of the National Academy of Sciences*, 96:7950–7955, 1999.

[98] J. R. Ullmann. An algorithm for subgraph isomorphism. *J. ACM*, 23(1):31–42, 1976.

[99] R. Veltkamp and M. Hagedoorn. State-of-the-art in shape matching. Technical Report UU-CS-1999-27, Utrecht University, the Netherlands, 1999.

[100] J. Z. Wang, J. Li, and G. Wiederhold. SIMPLIcity: semantics-sensitive integrated matching for picture libraries. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 23(9):947–963, 2001.

[101] X. Wang, J. Wang, D. Shasha, B. Shapiro, I. Rigoutsos, and K. Zhang. Finding patterns in three dimensional graphs: Algorithms and applications to scientific data mining. *IEEE Transactions on Knowledge and Data Engineering*, 14:731–750, 2002.

[102] C. Wei and C. Li. A general framework for content-based medical image retrieval with its application to mammogram retrieval. In *Proceedings of SPIE Medical Imaging:PACS and Imaging Informatics*, 2005.

[103] J. Winn and N. Jojic. Locus: Learning object classes with unsupervised segmentation. In *Proc. of IEEE International Conference on Computer Vision*, pages 756–763, 2005.

[104] X. Yan and J. Han. gSpan:Graph-based substructure pattern mining. In *Proc. IEEE International Conference on Data Mining*, 2002.

[105] S. Yu and J. Shi. Object-specific figure-ground segmentation. In *Proc. of IEEE Conference on Computer Vision and Pattern Recognition*, pages 39–45, 2003.

[106] S. X. Yu, R. Gross, and J. Shi. Concurrent object recognition and segmentation by graph partitioning. In *Neural Information Processing Systems*, pages 1383–1390, 2002.

[107] L. Zhang, H. Xiong, K. Zhang, and X. Zhou. Graph theory application in cell nuleus segmentation, tracking and identification. In *BIBE*, pages 226–232, 2007.