UNIVERSITY OF CALIFORNIA
RIVERSIDE

Image/Time Series Mining Algorithms: Applications to Developmental Biology, Document Processing and Data Streams

A Dissertation submitted in partial satisfaction
of the requirements for the degree of

Doctor of Philosophy

in

Computer Science

by

Oben Moses Tataw

June 2013

Dissertation Committee:
- Dr. Amit K. Roy-Chowdhury, Co-Chairperson
- Dr. Eamonn J. Keogh, Co-Chairperson
- Dr. Stefano Lonardi
- Dr. Tao Jiang

The Dissertation of Oben Moses Tataw is approved:

_____

_____

_____
Committee Co-Chairperson

_____
Committee Co-Chairperson

University of California, Riverside

Acknowledgements

Looking back at my experiences, I think of the popular proverb that says '*It takes a village to raise a child*'. To me, the village is all the people who have made it possible for me to get through this journey, some of whom I might never get a chance to thank. The child is me, the student.

I would like to start by thanking my academic advisors, Dr. Eamonn J. Keogh and Dr. Amit K. Roy-Chowdhury, for standing with me, challenging me, and guiding me through a long and trying journey. In times of personal agony, they were there to provide support and encouragement, especially at those times when I could not justify this journey. I thank you both for all your support and the enormous sacrifice of your time towards my success. Dr. Keogh was my faculty mentor during my undergraduate years and was a motivating factor in my decision to pursue graduate education. I am honored to have him co-chair my committee with Dr. Roy-Chowdhury.

I also want to thank Dr. Stefano Lonardi and Dr. Tao Jiang. I thank them for serving on my dissertation committee and for all the encouraging words throughout this process. All the courses I took with them, the discussions we had through the years, from my undergraduate years up until graduate school, have had a positive impact on my academic life. It is an honor to have them on my committee. In addition, I thank Dr. Tao Jiang for his role in my acceptance into the ChemGen IGERT program. Through that program, I was lucky to meet Dr. Roy-Chowdhury and my IGERT advisor, Dr. G. V. Reddy. I owe a lot of gratitude to Dr. Reddy for his role in leading my IGERT project and helping us understand the biological context of our work. My gratitude also goes to Dr. Thomas Girke,

who served on my orals committee. Next I would like to thank my colleagues in the data mining lab and the Video Computing Group. Their feedback and suggestions contributed to my growth as a student. Outside of school, I have an entire community of friends and well wishers who never miss a beat pushing and encouraging me to continue working hard, reminding me each day that my success is far bigger than me. I thank them all.

I survived years of challenges because of the unwavering support I received from my family. I owe them the greatest thanks. I am forever indebted to my wonderful wife, Noella Tataw, and lovely daughter, Orli Tataw. I thank my wife for loving me and staying up with me through long nights. She always has a way to get me motivated and focused. I am thankful for that. Like they say, '*Behind every successful man, there is a strong woman.*' She reminded me every day of the importance of hard work. She constantly reminded me about the joy and satisfaction I would feel once I get to the end. Whenever I face obstacles, she reminds me that '*Every disappointment is a blessing.*' I also thank my loving parents, brothers, and sisters, without whom I would not even have been in a position to benefit from the wonderful opportunity I have in this PhD program. Together, they understood the importance of education and provided whatever support they could.

I also must thank Dr. Julia Bailey-Serres, the director of the ChemGen IGERT program, Dr. Ernest Levister of the J.W. Vines foundation, the CSE administrative staff and many more I cannot list. They have all been instrumental in the progress I have made.

Finally, I give thanks to God almighty, for giving me the strength and most of all for bringing all these wonderful people into my life.

Dedicated to *papa Moses Oben Tataw* (RIP)

ABSTRACT OF THE DISSERTATION

Image/Time Series Mining Algorithms: Applications to Developmental Biology, Document Processing and Data Streams

by

Oben Moses Tataw

Doctor of Philosophy, Graduate Program in Computer Science
University of California, Riverside, June 2013
Dr. Amit K. Roy-Chowdhury and Dr. Eamonn J. Keogh, Co-Chairpersons

Interdisciplinary research in computer science requires the development of computational techniques for practical application in different domains. This usually requires careful integration of different areas of technical expertise. This dissertation presents image and time series analysis algorithms, with practical interdisciplinary applications to developmental biology, historical manuscript processing, and data stream processing. Inspired by the NSF IGERT program, this dissertation presents algorithms for analysis of growth dynamics at the shoot apex of Arabidopsis *thaliana*. A robust understanding of the causal relationship between gene expression, cell behaviors, and organ growth requires the development of computational techniques for quantitative analysis of real-time, live-cell meristem growth data. This requires the development/application of image analysis tools and novel time series alignment algorithms. Image analysis is necessary for the computation of growth features, but this leads to a time series of unsynchronized growth data, which requires a robust alignment method. Towards this end, we present two time series alignment algorithms. This dissertation further considers image mining in historical doc-

ument processing. An application of the Minimum Description Length principle (MDL) to develop a symbols clustering algorithm is presented. The developed algorithm produced one of the first practical applications of MDL to real-world, real-valued data such as images. Moreover, we introduce a novel premise that a clustering algorithm should have the freedom to *ignore* some data. Extensive empirical results show that the MDL-based algorithm outperforms the popular K-Means clustering algorithm, given the same input data, distance measure, and the correct value of K in K-means. The new algorithm could have significant impact, as clustering is a critical subroutine in almost all historical document processing systems. Finally, we present an algorithm for detecting rare and approximately repeating sequences in unbounded real-valued data streams, given limited space. This algorithm employs the novel integration of SAX time series representation with a Bloom filter to develop a robust cache maintenance policy that allows us to overcome known challenges to a previously unsolved frequent pattern mining problem. Our contribution lies in the fact that we solve this problem for real-valued data, whereas only the discrete-valued case has been considered in the literature.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

INTRODUCTION

Interdisciplinary research in computer science requires computer scientists to apply their knowledge towards creating practical solutions in other domains, including the promotion of basic scientific research. Flagship programs such as Integrative Graduate Education and Research Traineeship (IGERT), a United States National Science Foundation (NSF) program, have played a critical role in promoting and funding such interdisciplinary research. This research often requires the development of computational techniques to solve practical problems in other domains, usually demanding the application of many different areas of computational expertise. For example, to develop tools for analysis of live-cell growth at the shoot apex of Arabidopsis, both image processing and time series processing skills are required and must be integrated.

This dissertation focuses on the development of image and time series analysis algorithms with significant interdisciplinary implications. Application domains examined range from developmental biology to data stream processing and historical manuscript mining. The dissertation path was motivated by involvement in the chemical genomics IGERT (ChemGen IGERT) program at UCR (Fig. 1.1). The IGERT project involved research of the dynamics of growth at the shoot apex of *Arabidopsis*. The need for image and time series analysis techniques inspired our desire to further explore independent problems in the areas of image analysis (MDL clustering) and time series analysis (streaming data processing).

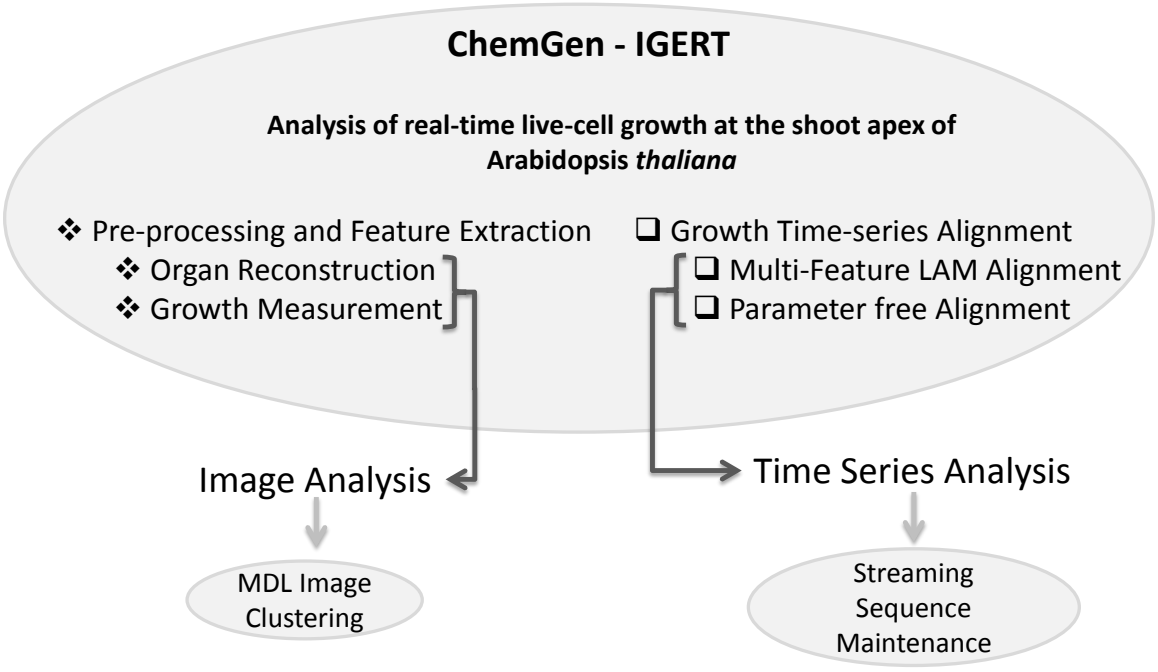

Fig. 1.1: Overview of Dissertation. The dissertation develops image and time series analysis algorithms with application to a diversity of scientific domains. The projects were motivated by involvement in NSF ChemGen IGERT program in the department of Botany and Plant Sciences.

This dissertation begins by presenting time series alignment algorithms developed as part of research on growth dynamics at the shoot apex of model plant *Arabidopsis thaliana* Fig. 1.1.*top*. To study the molecular control of organ growth, the causal relationship between gene expression, cell division, and organ growth needs to be established. Doing this requires the development of computational techniques to process and analyze time series of live-imaging data. Live imaging captures real-time, organ primordial and gene expression dynamics at cellular resolution. Understanding spatial relationships from live imaging data requires the spatial mapping and temporal alignment of different functional domains into a single template. A temporal alignment is a necessary step for any proper

gene expression analysis that is invariant to spatiotemporal changes. For this reason, two growth time series alignment algorithms are proposed: the first one is based on an existing landscape matching algorithm [2] and the second is a more robust parameter free alternative to the first. Both methods are relatively simple, but as we shall show, this simplicity is a great *strength*. In particular, the methods are essentially parameter free and are thus easier to use and less likely to be made to overfit on small datasets, especially by an overzealous practitioner. This work also required the application of image analysis techniques for basic volumetric image processing, organ reconstruction and growth measurement from reconstructed surfaces. The need for image and time series analysis skills required in this study motivated us to explore other image and time series mining problems as part of this dissertation.

Next, this dissertation addresses a purely image analysis problem with implications in historical document processing (Fig. 1.1.*bottom.left)*. Specifically, we consider a practical application of the well-known Minimum Description Length (MDL) principle to clustering glyphs (i.e. letters, symbols, etc.). Based on this principle and the premise that clustering algorithms should be allowed the freedom to ignore some data, an MDL-Clustering algorithm is presented and shown to be more robust than the ubiquitous K-Means alternative, even when both use the same distance measure and when K-Means is given the (unrealistic in the real world) advantage of knowing the correct value of K. The algorithm presented in this work has a practical impact in historical document mining and optical character recognition systems. It is a significant contribution because individual character clustering is often the first and most critical tasks in any higher-level document pro-

cessing system. This demonstrates a practical application of MDL on real world data such as images, and uses MDL directly in the clustering algorithm, as opposed to using MDL simply as a model selection criterion [41]. The key limitation that has kept MDL from being applied to such real-world data is that it is only defined for discrete data such as natural language strings or DNA strings; however, most images are intrinsically real-valued data matrices. As we shall show, we can address this issue to produce a practical algorithm.

Finally, this dissertation considers a challenging time series problem with potential impact beyond time series streams (Fig. 1.1.*bottom.right)*. Given an unbounded stream of time sequences (think *heartbeats* in an ECG stream) and bounded space, a novel algorithm for detecting 'rare', but approximately repeating, real-valued sequences is presented. This problem is a variant of the challenging frequent items mining problem, which is still more or less an open problem. We present an algorithm that integrates the SAX time series representation [63] and a Bloom filter [59] to implement an efficient cache maintenance policy that is more robust than a randomized alternative, the only obvious straw man. The novelty of our contribution lies in the integration of the SAX transformation with a Bloom filter in the implementation of a cache replacement policy. Moreover, as far as we know, this is one of the first frequent items mining algorithms designed to deal with *real-valued* time series.

Motivated by involvement in interdisciplinary research supported by the NSF IGERT program, we have developed algorithms with impact in plant biology, streaming data processing, as well as historical document processing. In chapter 2,

we present details on the IGERT-supported work in this dissertation, including a full background on the biological motivation, novel application of existing image processing techniques, and the development of novel alignment algorithms for proper temporal synchronization of growth data. Chapter 3 will present details on our MDL clustering algorithm. Finally, in Chapter 4, our cache maintenance algorithm for streaming data is presented.

CHAPTER 2

QUANTITATIVE ANALYSIS OF LIVE-CELL GROWTH AT THE SHOOT APEX OF ARABIDOPSIS THALIANA: ALGORITHMS FOR FEATURE MEASUREMENT AND TEMPORAL ALIGNMENT.

Study of the molecular control of organ growth requires establishment of the causal relationship between gene expression and cell behaviors. We seek to understand this relationship at the shoot apical meristem (SAM) of model plant Arabidopsis *thaliana*. This requires the spatial mapping and temporal alignment of different functional domains into a single template. Live cell imaging techniques allow us to observe real time organ primordia growth and gene expression dynamics at cellular resolution. In this paper, we propose a framework for measurement of growth features at the 3D reconstructed surface of organ primordia, as well as algorithms for robust time alignment of primordia. We computed areas and deformation values from reconstructed 3D surfaces of individual primordia, from live cell imaging data. Based on these growth measurements, we applied a multiple features landscape matching algorithm (LAM-M), to ensure a reliable temporal alignment of multiple primordia. Although the original landscape matching algorithm (LAM) motivated our alignment approach, it sometimes fails to properly align growth curves in the presence of high noise/distortion. To overcome this shortcoming, we modified the cost function to consider the landscape of the corresponding growth features. We also present an alternate parameter free growth alignment algorithm which performs as well as LAM-M for high quality data, but is more robust to the presence of outliers or noise. Results on primordia and guppy evolutionary growth data show that the proposed alignment frame-

work performs at least as well as the LAM algorithm in the general case, and significantly better in the case of increased noise.

2.1 Introduction

Advances in live cell imaging techniques such as those developed in [20], present opportunities to tackle important plant development questions. Live imaging allows us to investigate the causal relationship between cell behavior, organ growth and genes that work in networks. This is possible because with live imaging, we are able to observe in real time, organ primordia growth and gene expression dynamics at cellular resolution. In this paper, we present a framework for quantitative study of primordia growth at the shoot apical meristem of Arabidopsis, (shown in Fig. 2.1), based on live imaging data.

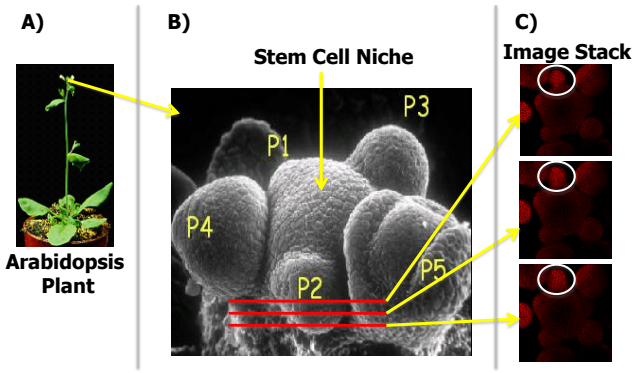

Fig. 2.1. Location of the Shoot Apical Meristem in model plant Arabidopsis *thaliana*. (A) Location of SAM. (B) Detailed view of SAM showing multiple primordia (P1-P5) at different developmental stages. (C) Illustrative sample of 3 out of 23 slices from an imaging session. White circles indicate location primordia per slice.

In doing so, we present algorithms for temporal alignment of primordia growth data and any other developmental growth data for that matter. Time synchronization is often a necessary step in any meaningful analysis of time lapse data. Time lapse data generated

from live imaging experiments is very noisy and often requires robust time synchronization for reliable deduction of these relationships. In addition to plant biology, a similar problem is faced in other areas of science including biological and medical sciences, where experimental data is often very large and usually unsynchronized.

Quantitative time series data generated from these experiments is sometimes collected through the measurement of interesting variables over varying timeframes. Such measurements and subsequent analysis are essential in efforts to gain greater insight into biological systems.

In the medical community, the study of multiple characteristics of human respiration can lead to the discovery of important dynamics in patient outcome studies [9],[28]. In order to achieve an acceptable level of reliability, such data needs to be properly time synchronized across multiple patients. In proteomics studies, there is a need for quantitative comparison of multi-class Liquid chromatography-Mass spectrometry (LC-MS) data [3]. The very nature of data generation in LC-MS studies is itself a source of large variability. Given the degree of noise and variation, a robust alignment of generated time series is often a necessary step towards reliable multi-class comparison.

One common and necessary requirement in these different domains and many others is the need to collect a large amount of unsynchronized data, properly align them and then perform further analysis.

Although there has been a fair amount of progress in this problem domain, to the best of our knowledge, no one has looked at the applicability of similar techniques in the study of plant development. Specifically, we do not know of any similar study applied to

primordia growth dynamics at the shoot apex of Arabidopsis *thaliana*. Given high resolution data, many existing techniques have proven to be reliable and robust. However our problem presents many challenges in the form of low resolution time series, with high variability. Specifically we are challenged by the fact that we have about 5 to 13 time points per observed event. With such low resolution, the effect of noise and variability can be greatly magnified.

2.2 Main Contribution

This paper makes four specific contributions in the study of biological growth at the level of the organ (e.g. primordia):

- We show how to measure growth features from reconstructed 3D structures of organ primordia, based on live imaging data. (Section 2.6.1).
- We show the application of deformation field morphometry (DBM) for quantification of deformation at the surface of organs. (Section 2.6.1.2).
- We present a modified landscape matching algorithm for the alignment of low resolution, noisy time series using multiple features. (Section 2.6.2.1.1).
- We present a parameter free growth alignment algorithm. (Section 2.6.2.2).

2.3 Background

Although a full biological description of the primary system under study is beyond the scope of this paper, we present a brief description and motivation for our work. The Shoot Apical Meristem (SAM) is made up of stem-cells that provide cells for the development of all above ground plant structures. The organ primordia are regions of the SAM that develop into different plant organs. At each point in time, the SAM con-

tains multiple primordia at different developmental stages, as shown in Fig. 2.1.B. The SAM is subdivided into different functional domains, with unique and overlapping gene expression patterns. Differential expression analyses have led to some understanding of the expression patterns of many SAM genes. In [26], analysis of just 3 cell types revealed more than 2000 genes with distinct and overlapping expression patterns. However, such an analysis is static and does not provide the dynamic context of observed patterns. It also does not provide the relationship between observed expression pattern, cell-cell communications, cell expansion/growth rates and organ growth.

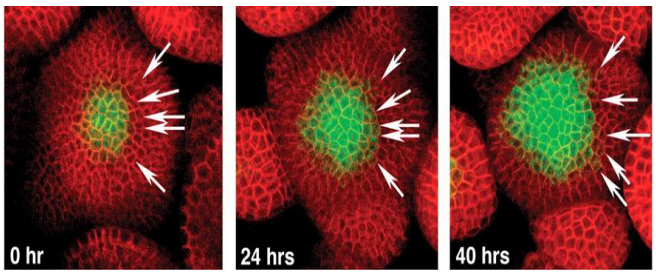

Fig. 2.2. Live imaging data of clavata-3 gene expression showing expression rates at 0hr, 24hrs and 40hrs. Live-imaging techniques allow for real time observation of growing SAMs. Proper alignment allows for dynamic gene expression analysis that is invariant to spatiotemporal changes. Data from [19]

Data for our project was collected using the live imaging technique developed in [20]. Live imaging allows for real time observation of primordia growth at the resolution of the cell. It also allows for real time observation of gene expression and cell division over a period of time, while the plant is alive and growing. These techniques use laser scanning confocal microscopy to obtain 3D volumes of SAM struc-

ture at different time instances throughout an observation period (usually several days), without damaging the growing plant. With live-imaging, we are able to capture the dynamic context of discovered genes, as illustrated in the images in Fig. 2.2, which were published in [19]. Live imaging also allows for computational tracking of cell division patterns as in [13]. We are proposing a framework that will lead to the possibility of performing dynamic analysis of live imaging and other growth data. Such an analysis will be invariant to spatiotemporal changes in plant structure over time.

Data collected at each time instance of live imaging is in the form of a 3D stack of slices representing the view of the SAM at different focal planes from top to bottom. Analyzing primordia morphogenesis based on measurable growth features is an important problem to plant biologists. It is important precisely because genes drive primordia growth. As a result of this, biologists need to understand the spatiotemporal dynamics of the interaction between gene expression and primordia growth. We seek to deduce the principles underlying the relationship between gene expression, cell division, cell-cell communication and overall primordia growth. Such an understanding will move us a step closer towards developing a dynamic gene expression atlas for the SAM of model plant Arabidopsis thaliana. Working towards this goal (see big picture in Fig. 2.3), we solve two important problems:

- Measurement of growth features at the surface of primordia
- Alignment of time series of growth features

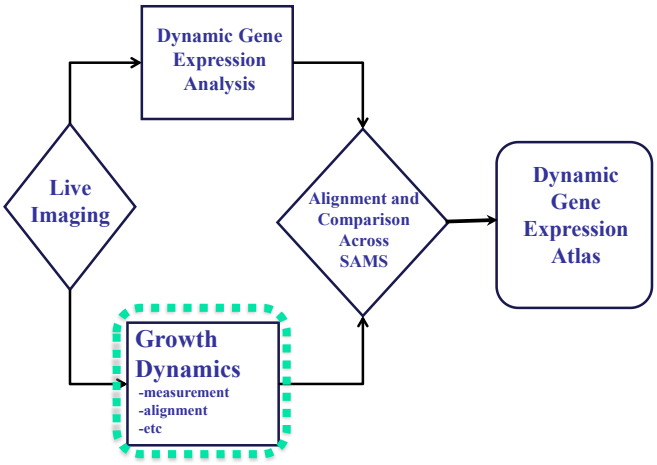

Fig. 2.3. Motivation for our study of growth at the SAM of Arabidopsis *thaliana*. Proper temporal alignment of growth data will allow for reliable development of dynamic models that integrate gene expression and quantitative data, both at the global (SAM) level and at the resolution of the cell.

Given the advantages of live imaging, one might imagine that we could just label all discovered genes in a single plant and perform our analysis using live-imaging. We are however limited in live imaging, because technical limitations mean we can only use 2 to 4 gene markers at a time. As a direct result of this restriction, we cannot study more than a few genes at a time using a single plant. Given that there are thousands of genes, and there is variation in SAM size and shape, we must perform analysis of different genes in different plants, and then integrate the results into a common template. Given that the SAM consists of multiple primordia at different developmental stages, time lapse data for a set of primordia have varying start and end times. Such data needs to be time synchronized in order to properly perform reliable analysis. This also requires complex alignments and comparisons across plants which calls for computational tools

for SAM growth analysis. For this reason, our study of automatic growth alignment is a necessary first step in this process.

2.4 Relation To Previous Work

The nature of the problem in our work is such that we are analyzing multiple agents per subject. These agents are undergoing similar biological processes (growth), but at any given moment, they are at different stages of that process. The challenge does not only come from the variability across SAMs, there is also great variability within the same subject (SAM).

In terms of measuring growth features, there has been some work within the past decade [5],[7],[11]. Our work differs from these studies in many ways. First of all our goal is to study the spatiotemporal dynamics of growth and use the output of our studies in the development of a dynamic model. Secondly, our dataset is significantly different. In [5], they used replicas taken from the surface of individual apex as input into their shape tracking system. In contrast, we use live imaging data that allows us to track gene expression at the resolution of the cell, while they used replicas taken from the surface of each individual apex to track shape changes. In [7], similar imaging equipment was used for data collection. However they used a different technique in their live imaging experiment. Their technique differed in the sense that they employed a multi-angle image acquisition approach, and then applied an image reconstruction technique that integrated these images into a single image with better resolution. They had a 24 hour time interval between observations, a constraint that was imposed by their approach. Such

a long time window can present the risk of missing significant growth dynamics. In our case, we allowed 6 hour intervals between observations and took images from a single angle (horizontal). With a 6 hour interval, attempting the imaging approach employed in [7] will be very traumatic to the plant, which will most likely not survive the adverse effect of laser intensity.

Our proposed system takes 3D surface points as input. These points represent the reconstructed 3D shape of primordia. Since we are focused on the study of global patterns in primordia growth, any contour based 3D reconstruction scheme is suitable to use with the proposed framework. Many image reconstruction techniques have been proposed in the past decades [1],[6],[14]. Methods like the classical Marching Cubes approach [14] or the more recent technique based on the Multi-level Partition of Unity (MPU) models [1] are examples of techniques that have been shown to produce acceptable results in many application domains.

In this work, we further show how to compute local surface deformation from 3D reconstructed live imaging data. Our approach to computing deformation was inspired by the phenomenon of deformation based morphometry (DBM). Although DBM originated from solid mechanics, it has been widely embraced in the medical imaging community to study a variety medical conditions. For example DBM has been applied to study the effects of alcoholism in [21], changes in human brain [17],[22], as well as analysis of gray matter deformation in [4]. The validity of many variations of DBM has also been the focus of previous studies

[8][23]. Although the adoption of DBM in the bio-medical imaging community has seen wide success, the approach has never before been used to study primordia development in the way we are proposing here.

Another major problem we address in this paper is that of time series alignment. In fact, we present two solutions for the temporal alignment of growth data, a task that is at the heart of this paper. Time series alignment is a problem that has attracted significant interest in the past. A widely adopted solution to this problem is dynamic time warping (DTW), an algorithm that has attracted interest from the data mining community [10],[27], although it came about as a result of work in the speech recognition community. Over the years, many variants of the DTW algorithm have been developed [16]. However, the key to most of these is the utilization of a robust cost function, in cases of high variability in amplitude and time. In addition to DTW, there are other approaches that have yielded reasonable results [2],[12]. In [2], a landscape based cost function was applied to develop what is referred to as the Landscape matching algorithm (LAM). This method was used to align data from respiration patterns of multiple patients. A statistics based approach was employed in [12], where multiple time series are simultaneously aligned and eventually a latent trace was inferred from the aligned set. This approach is very robust. However, because our plant growth data has very low temporal resolution, we cannot guarantee the correctness of such an approach. Our approach to align varying length and noisy sequences is detailed in Section 2.6.2.

2.5 Detailed Methodology

In this section, we give a detailed presentation of the entire proposed framework, from growth measurement, to eventual growth alignment of time lapse data. Although the primary goal of this paper is to present solutions for aligning growth data, we think for completion, it is important to give details on how we measure some of the features being aligned.

2.6 Overview of Proposed Framework

At any moment in time, the SAM is made up of multi- ple growing primordia at different stages of growth. The proposed framework is designed to handle the analysis of primordia from different plants. As shown in Fig. 2.4, the system is made up of two distinct modules: the growth computation module (GCM) and the growth alignment module (GAM). The GCM includes sub-modules to measure growth. For example in the study of primordia growth, the GCM sub-modules will play the role of measuring growth features at the surface of reconstructed primordia. The output of this module is a time series of primordia growth measurement, for all plants under study. Once we have a collection of time series, the growth alignment module performs a multiple series alignment on all the primordia. Deformation values computed by the GCM are used to improve alignment results.

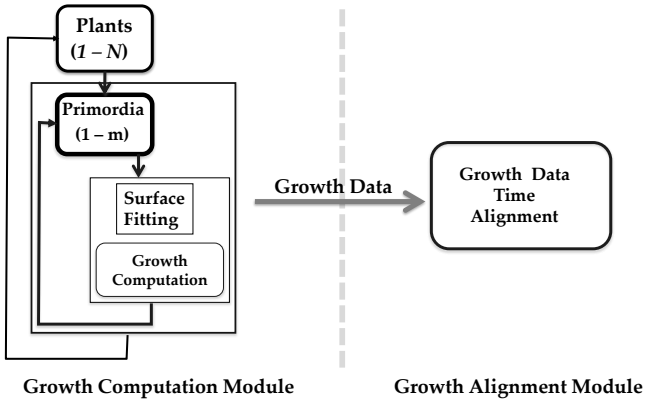

Fig. 2.4. The proposed system is made up of the growth computation module (GCM), and the growth alignment module (GAM).

The next sections will give details on the methodology used for growth computation, and the alignment of measured features.

2.6.1 Growth Measurement

2.6.1.1 Surface Area Calculation

Given a stack of boundary points representing a primordia, we use existing tools to generate a smooth surface as shown in Fig. 2.5. The surface area of our reconstructed primordia surface is computed by performing a Delaunay triangulation, and then taking a summation on the areas of the triangles. Summation of the areas of all these surface triangles gives us an accurate estimate of the surface area of the primordia. Although there are many ways of computing the area of a triangle, we chose to use Heron's formula for computing the area of our triangle.

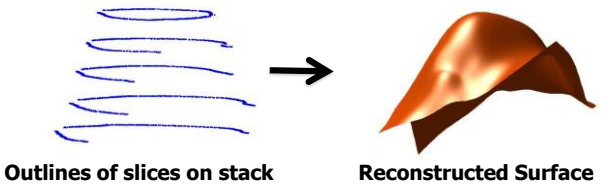

Fig. 2.5. Sample input into the system (left) is in the form of slice contours which are then used to reconstruct a smooth surface. Here we show a Matlab™ visualization of the reconstructed surface.

2.6.1.2 Deformation Computation

One of the fundamental patterns biologists seek to understand is the variational nature of deformation on the surface of the organ over time. We cannot begin to deduce the dynamic nature of organ deformation without first of all deriving a quantitative representation of such activity. In order to solve this, we used deformation based morphometry (DBM), an approach that has been widely adopted in the medical image analysis community [4],[8],[17],[21],[22]. Our DBM approach is composed of three major steps:

- Establish Point Correspondence between target surface T and source surface S, where T is the primordia surface at time t+1 and S is the same primordia surface at time t.
- Compute deformation field based on point correspondences.
- Quantify deformation by looking at the Jacobian at each point on the surface.

2.6.1.3 Finding Correspondences

Optimal correspondence can be achieved by solving the registration problem between surfaces S and T. This is done by addressing it as an optimization problem, where the goal is to minimize some error functional. Once we have our two point clouds properly registered, we chose as optimal, the correspondence that led to convergence in our optimization. The approach we employed is based on the well known Iterative Closest Point (ICP) algorithm for rigid registration of point sets [15]. Given the system we use for primordia growth data collection, surfaces generated from our data set are almost registered. This observation makes our data suited for use of the ICP algorithm, given its tendency to get stuck in local minima. Leveraging off the formalisms in [15], we now give a simple description of the basic ICP algorithm.

Given an initial state, our task is to minimize the sum of squared differences between the transformed target points and their correspondences on the source surface. This is done by minimizing the error function E_M, through an iterative process of computing correspondences and then optimizing the transformation parameters based on these correspondences.

$$E_M = min \sum_{i=1}^{n} [\|M(T_i) - S_i\|^2]$$

where E_M, is the error due to transformation M, and n is the number of correspondences and i is the current correspondence from a point in T to a point in S. Note that each point on the target has a correspondence to a point on the source surface.

The first step in the process of finding the optimal correspondence is to compute the initial transformation parameters between the two 3D point clouds representing our surfaces. Let the rotation parameter be r_c and the translation parameter be t_c. Given these initial or current state parameters, the next step is using the current state to find new correspondences $C_{(S_i,T_i)}$.

$$C_{(S_i,T_i)} = \arg\min[\|(r_c T_i + t_c) - s_i\|^2]$$

where $C_{(S_i,T_i)}$ is the i^{th} correspondence between T and S. Based on these new correspondences, update r_c and t_c, using the following equation

$$[r,t] = argmin[\left\|(r_c T_i + t_c) - S_{C_{(S_i,T_i)}}\right\|]$$

where $[r,t]$, is the updated rotation and tranlation parameters, based on previous state and correspondences computed from those previous values.

If the transformation error due to the updated transformation parameters is not less than a pre-set threshold, repeat the process over by computing new correspondences based on current transformation. The correspondence that leads to convergence is selected as the optimal correspondence.

2.6.1.4 Quantifying Deformation

Once correspondence between points on surfaces at time t and t + 1 have been established, we use these correspondences to derive a displacement field. Displacement fields are represented in the form of displacement vectors. The structural difference between the two surfaces is encoded in this displacement field and this fact gives us a way to directly quantify deformation.

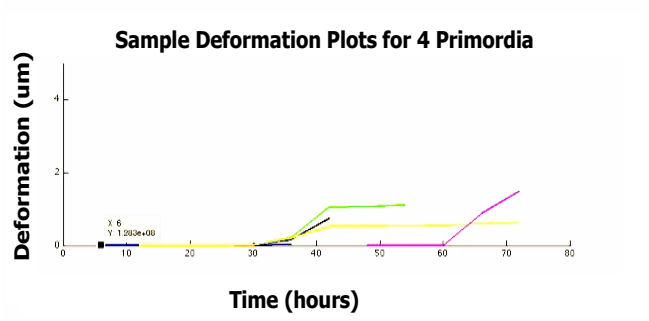

Fig. 2.6. Sample plots of total deformation computed from four different primordia. Total deformation is calculated by taking the sum of deformation values at points on the surface.

Given a displacement field from point correspondences, we quantify deformation by evaluating the local change at the resolution of each point on the surface. This is done by taking the determinant of the Jacobian matrix D at each surface point. Let U be the displacement vector at point $P(x, y, z)$ on the target surface. The deformation at P is defined as $\det(D)$, where D is defined as follows:

$$D_U = \begin{bmatrix} \frac{\delta U_x}{\delta x} & \frac{\delta U_x}{\delta y} & \frac{\delta U_x}{\delta z} \\ \frac{\delta U_y}{\delta x} & \frac{\delta U_y}{\delta y} & \frac{\delta U_y}{\delta z} \\ \frac{\delta U_z}{\delta x} & \frac{\delta U_z}{\delta y} & \frac{\delta U_z}{\delta z} \end{bmatrix}$$

where U is he displacement vector at the current point. A sample plot of total deformation over time for multiple primordia is shown in Fig. 2.6. As expected, primordia initially exhibit little or no deformation, but that changes at later stages of growth.

2.6.2 Growth Alignment

In our proposed system, output from the GCM is a time series of surface areas (or another growth measurement), computed for each primordia considered. Growth alignment as used in this paper refers to the temporal alignment of times series of growth measurements for multiple primordia. In all our experiments, we will be aligning growth data from multiple objects. However, the fundamental step in all these is the alignment of one growth time series against another. As such, our description of solutions for alignment of growth data will focus on alignment of two time series. After that, we will finally discuss how to apply these solutions as a sub-routine in the alignment of multiple time series.

Now we give a high level presentation of the alignment problem addressed here.

The Alignment Problem:

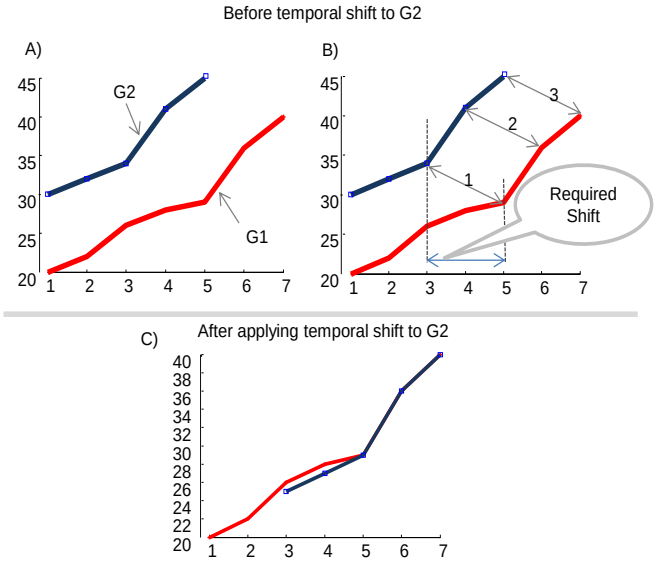

Fig. 2.7. The alignment problem we seek to solve involves finding the shift St necessary to bring G2 into an alignment with G1, such that some cost function is optimized.

Given two time series G_1 and G_2, the growth alignment problem is defined as the task of finding the temporal shift S_t, such that when applied to G_2, it leads to the best possible alignment of G_2 to G_1. As shown in Fig. 2.7.B, there are points in the two time series that are most similar and thus can be used as a basis for finding the required shift. Fig. 2.7.C shows the expected output, at least in this toy example, that is should come out of a good alignment. In this case, series G_2 has been shifted by a factor of two in the temporal direction. Note that in Fig. 2.7.A and Fig. 2.7.B, G_2 is plotted with a vertical shift of +5 to allow for easy display and illustration of the problem.

This definition of our alignment problem requires that the most similar subsequences between the two time series fall within the overlapping region that resulted from applying

the shift S_t to G_2. What we have left now is to present our solution to this problem, given the application domain in consideration. We now present two solutions to the problem of aligning low temporal resolution time series, with a focus on the alignment of plant growth data. The first approach is built on the Landscape Matching algorithm (LAM), while the second is a new parameter free alternative.

2.6.2.1 Landscape Based Alignment

We now give a brief review of the Landscape Matching (LAM) algorithm presented in [2]. This algorithm utilizes a landscape vector λ, calculated from the overlapping regions of the two series shown in Fig. 2.7.A, to calculate a matching score using the following cost function from normalized values of G_1 and G_2 :

$$M_{(G_1,G_2)} = argmax\ \frac{1}{L_1}\sum_{n=1}^{L^*}\frac{\left(1+\lambda^{\theta}{}_{(G_1^*,G_{2,s_t}^*)}(n)\right)}{1+|\ G_1^*(n)-G_{2,s_t}^*(n)\ |},$$

where L^* is the length of the overlapping region, λ is the landscape vector, L_1 is the length of G_1, s_t is the temporal shift of the current iteration, and G_1^* and G_2^* are the overlapping segments.

This algorithm considers all possible temporal shifts, with the goal of maximizing the landscape function $M_{(G_1,G_2)}$. As we shift G_2 across G_1 from left to right, starting with an initial overlap, we get different matching scores for resulting overlaps. The shift that leads to the highest matching score is selected as the required shift for the best possible alignment.

In Fig. 2.8, we show an example of calculating λ from the overlapping regions of two time series. Calculation of λ requires us to set a threshold parameter θ, within which we consider two corresponding points from the time series to be a match. The matching locations are then sorted in ascending order by time.

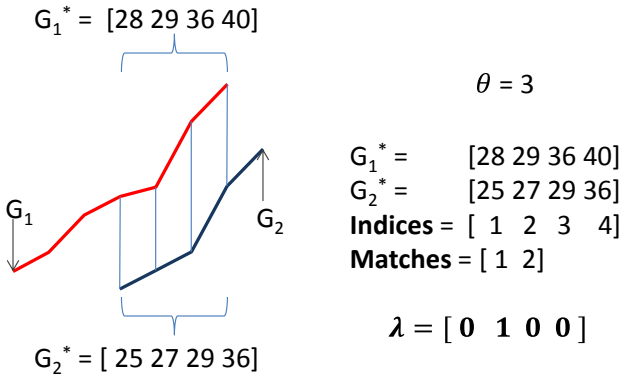

Fig. 2.8. Calculating the landscape vector. The threshold is set to 3 for this example.

The first position in λ and any other index location that is not in the list of matching positions is set to 0. For the rest of the index locations in the overlapping regions, the values are set to the absolute difference between the value at the index position in G_1, and the value of the prior matching index in G_1.

2.6.2.1.1 Multiple Features Landscape Based Alignment

The LAM algorithm requires that the threshold parameter be meaningful, since the result of the landscape function is sensitive to this parameter, and as such the algorithm can sometimes be sensitive to noise and outliers. With this motivation in mind, we decided to generalize the LAM function so that it can be used for alignment based on multiple

growth features. The intuition of this generalization is that, provided other features have a property that significantly discriminates different developmental stages for example; we could leverage such property to improve the alignment due to the first feature. Consequently, a false positive due to noise from a single feature will be compensated by a strong match due to other features.

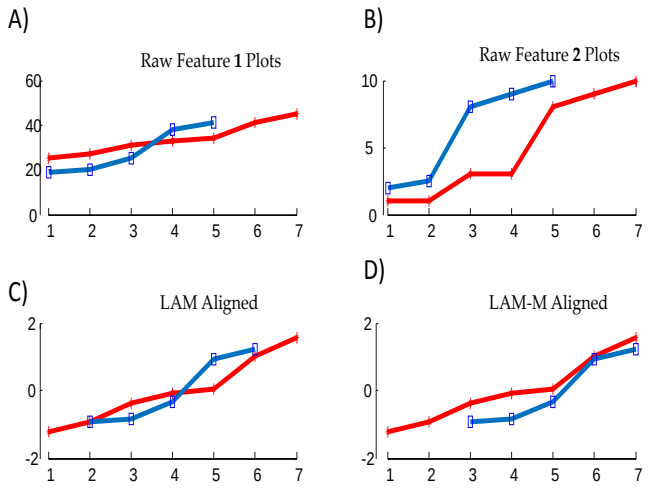

Fig. 2.9. Plots showing an example where single feature LAM algorithm fails to find the proper shift. By incorporating a second feature, LAM-M computed the correct shift.

For multiple features, we want to find the shift such that the cumulative landscape function of all features is maximized. For a particular feature f_i, we define the new landscape function for normalized values of the time series as follows:

$$m_{f_i} = \frac{1}{L_1} \sum_{n=1}^{L^*} \frac{\left(1 + \lambda^{\theta}{}_{f_i,(G_1^*,G_{2,s_t}^*)}(n)\right)}{1 + |\, G_1^*(n) - G_{2,s_t}^*(n)\,|}$$

The new cost function for multiple features is then defined as:

$$M_F = \prod_{i=1}^{k} m_{f_i}$$

This equation shows the new cost function for a multiple-feature LAM, which for ease of presentation, we refer to as LAM-M. The landscape vector for each feature is still calculated in the same way, and the goal now is to maximize the joint landscape of all the features.

In Fig. 2.9, we show an example where the basic LAM algorithm fails to properly align two fairly similar time series. In Fig. 2.9.B, we also show raw plots for a second feature, in this case deformation curves. In Fig. 2.9.C, we show alignment due to the original LAM algorithm, while in Fig. 2.9.D, we show how incorporation of multiple features leads to a correct alignment of the two time series.

2.6.2.2 Parameter Free Alignment

Although the LAM algorithm works well, and our extension allows us to improve performance in the rare cases were LAM fails, we recognize two issues that could be problematic in some cases: first of all our data could be such that, the landscape function is too sensitive to the threshold parameter lambda. The second issue is that we might not have the luxury of a second meaningful corresponding feature. For these reasons, we now present a parameter free simple alternative to the LAM algorithm.

Our proposed parameter free algorithm looks to minimize the mean of Euclidean distances between subsequences in two time series G_1 and G_2. Unlike the LAM, it does not seek to analyze the entirety of the overlapping regions at any moment in time. It simply looks at the most similar subsequences by minimizing the mean of the Euclidean distances between corresponding points on the subsequences/motifs. Note that we use Euclidean distances here, but the distance metric could be substituted.

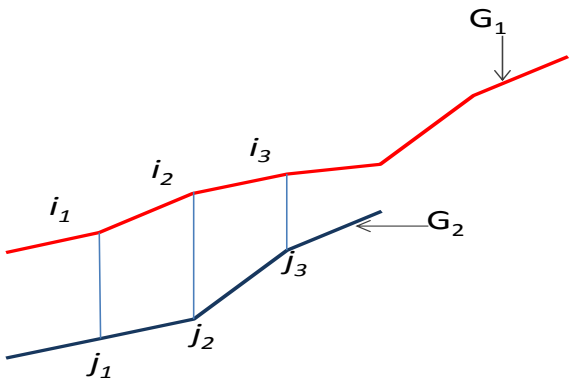

Fig. 2.10. Calculating the similarity score between two subsequences. Subsequences are labeled with i and j indices

Once we find the most similar subsequences, we calculate the shift necessary for the best possible alignment, based on the first matched indices of the subsequences. Because the most similar subsequences do not have to include the entirety of overlapping regions between two time series, this algorithm tends to be more robust to outliers that could throw off the landscape function in the LAM algorithm.

Given two time series G_1 and G_2, we find subsequences of all lengths from G_2, such that the maximum subsequence length is at most the length of the shorter of the two time series. Within this space of subsequence lengths, we find the closest pairs between G_2 and G_1. Let the length of subsequence g_2 be l_2. We now define closeness score $S_{(g_1,g_2)}$ as follows:

$$S_{(g_1,g_2)} = \frac{1}{L} \sum_1^L d(g_{1,i}, g_{2,i}),$$

where $d(g_{1,i}, g_{2,i})$ is the distance between the corresponding points in the subsequence.

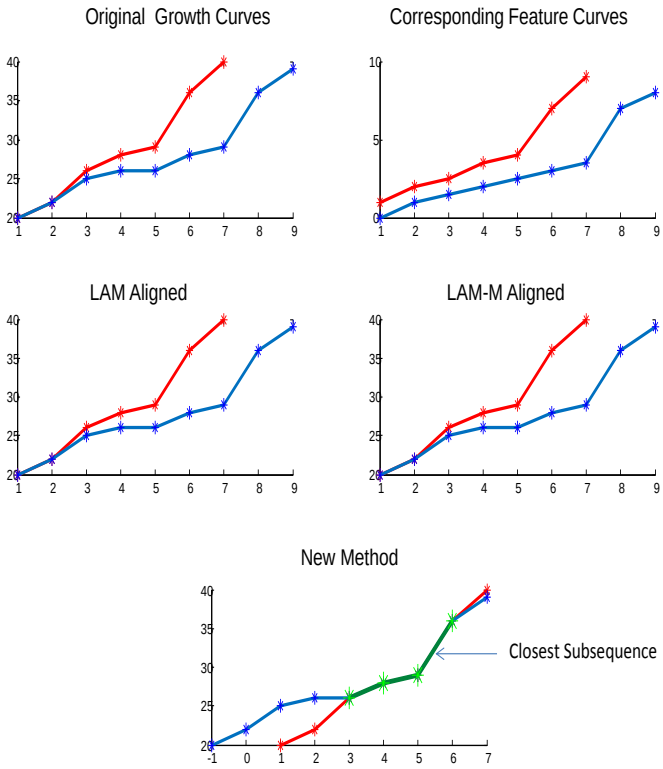

Fig. 2.11. In some cases, even the presence of a second feature might not be useful. As we see here, even LAM-M failed for this test data set. However, the new parameter free algorithm correctly aligned both time series.

In Fig. 2.10, we show an illustration of subsequence closeness scoring, for a subsequence of length 3. The process involves the summation of distances between corresponding points in subsequences being considered. In order to penalize matches that result from short sequences, we divide this sum by the length of the subsequence. We acknowledge that this penalty is a somewhat naïve approach to favor longer subsequences. Others have presented intuitive approaches in the Minimal Description Length (MDL) space [18]. However, empirically our choice of simply taking the mean is extremely ro-

bust over many datasets. The matching pair that minizes the mean of the distances between corresponding points forms the basis for calculating the required shift S_t on G_2.

This parameter free algorithm sometimes outperforms LAM, even in the presence of multiple features. This is due to the fact that it relies on subsequences, as opposed to an entire overlapping region, and as such, it is less susceptible to outliers. Of course it is also true that there may exist a threshold at which LAM-M might do just as well.

2.6.3 Multiple Growth Alignment

Now that we have shown two alternatives to aligning two time series, we now show how we use them to align multiple time series. In the algorithm shown in Table 2.1, we assume the parameter free alignment algorithm. However, it can easily be changed to use the LAM-M optimization strategy instead.

This algorithm iteratively aligns the best possible pairs of time series, until there is no more time series to add to the aligned set. In Table 2.1.line 19, we make a call to our choice of alignment algorithm, in this case our parameter free algorithm. Lines 14 to 19 find the best match and shift between the current time series, and the rest of the time series that have yet to be aligned.

Note that this algorithm does not traverse the list of time series L, in order. For simplicity, we set the starting point to be the first time series in our list. This index location gets placed on the stack of touched time series in Table 2.1.line 7. In Table 2.1.line 15, we check to make sure we only compare against untouched time series. The time series that is best matched to the current source series becomes the source series in the next iteration and this is set in Table 2.1.line 26. In Table 2.1.line32 we consider the global context as

we shift the current matches. The algorithm terminates once the list of touched indices is greater or equal to the length of input set of time series.

Table 2.1. Main Multiple Alignment Algorithm

Input:	*L* : list of time series
Output:	*S*: list of required shifts for time series in L.
1	Scores = {} // structure to hold similarity scores
2	S = [ ]
3	i = 1;
4	S(i) = 0;
5	C = [ ]; //empty set
6	**while** len(C) < len(L)
7	g = L(i)
8	C(len(C) + 1) = i;
9	**if** (~isset($Scores_i$.distance))
10	$Scores_i$.distance = inf
11	$Scores_i$.shift = 0
12	bestScore = inf
13	bestInd = i
14	bestShift = 0
15	**for** j from 1 => len(L)
16	**if** j ϵ C
17	continue
18	**end if**
19	h = L(j)
20	[shift score] = alignSeries(g,h)
21	**if**(score < bestScore)
22	bestScore = score
23	bestShift = shift
24	bestInd = j
25	**end if**
26	**end for**
27	i = bestInd
28	$Scores_i$.distance =bestScore
29	$Scores_i$.matchIndex = bestInd
30	$Scores_{bestInd}$.distance = Score
31	$Scores_{bestIndex.}$matchIndex = i
32	$Scores_{bestIndex}$.shift = $Scores_i$.shift + bestShift
33	S(i) = $Scores_i$.shift;
	end while

2.7 Results And Analysis

We show alignment results on both synthetic and real data. We generated synthetic data to test the limits of the multi-feature landscape alignment (LAM-M). Synthetic data allows us to properly validate the alignment algorithm, since we are absolutely certain of the ground truth. In addition to aligning time series of primordial growth, we also show application of our algorithms on a longitudinal data set of individually marked guppies (Poecilia *reticulata*) in the wild.

2.7.1 Multiple-Features Landscape Alignment with Synthetic Data.

To generate synthetic data, we start with a single known time series, which we call the 'Main' pattern. From this pattern, we then generate multiple sub-patterns of variable lengths by applying spatiotemporal random noise to different segments of the 'Main' pattern. This allowed us to create noisy samples that were sure to test the limits of our multiple features algorithm.

A plot of the unsynchronized version of the data is shown in Fig. 2.12. Note that the starting measurements of the generated growth curves are almost the same. This property can also be observed in the live imaging data. This is due in combination of the nature of primordia growth, as well as human error during imaging, as well as the plant trying to adjust from the trauma caused by initial exposure to laser light. The fact that the generated data has this property makes it that more suitable for testing the multiple feature landscape alignment.

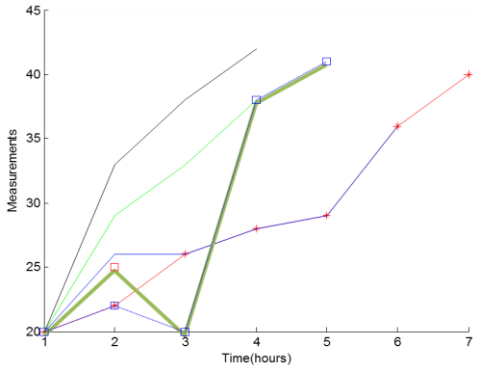

Fig. 2.12. Raw plot of the un-aligned synthetic data. This data has similar properties to those observed on real primordia growth data.

In Fig. 2.13, we show the results of aligning the raw time series using the LAM algorithm. Here we see that one series is seriously misaligned due to noise and outlying data points that cannot be overcome by the basic landscape function. To overcome this problem, we applied the LAM-M, to take advantage of the corresponding deformation values from the 'Main' trace and the incorrectly aligned curve.

As expected, the algorithm that incorporated the effects of multiple features was able to give a perfect alignment. Note that such results are only possible because this example shows the presence of a strong indicative second feature. As we showed before, there are cases where even the presence of a second feature is not helpful.

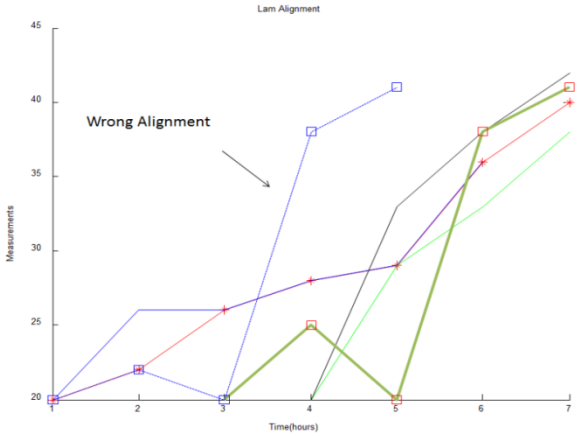

Fig. 2.13. Alignment of synthetic data using original LAM algorithm. Although LAM does fairly well, it failed to properly align one growth curve. We know this because we know the ground truth.

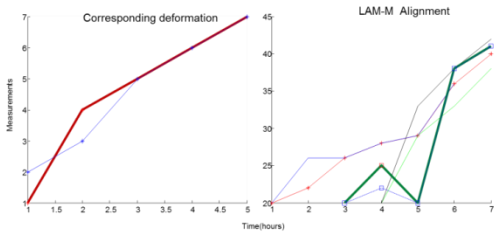

Fig. 2.14. Perfect alignment with LAM-M, using multiple features. (Left) Deformation plot for main trace and wrongly aligned series. (Right) Properly aligned set, due to strong second feature.

2.7.2 Results on Live Imaging Data.

Data used in this project was collected from a live imaging experiment that lasted up to 72 hours. Input into the system is a set of surface points for each primordia in a single plant over time. We use these surface points to compute the time series of surface areas for each primordia. For our experiment, we maintained a time interval of 6 hours between data points. An example of a single primordia input at a single time instance is

shown in Fig. 2.5.right. Given a stack of images, primordia contours shown in Fig. 2.5.left can be automatically detected using available contour based feature detection techniques [24][25].

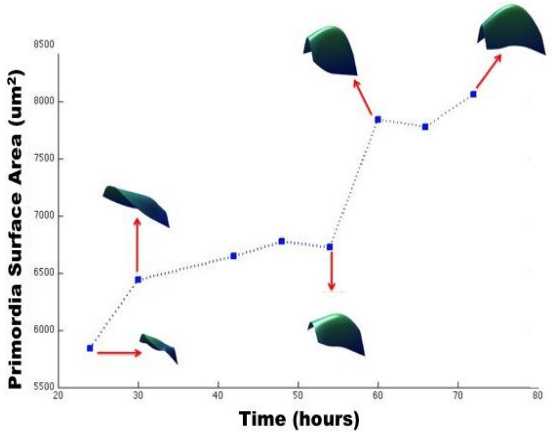

Fig. 2.15. Sample growth pattern, showing single primordia growth time series.

At the beginning of each live imaging experiment, developing primordia are at different developmental stages. At some point during the experiment, mature primordia that have been growing through out the observation period differentiate into other organs and exit the meristem. At that point we stop observing such primordia. Such exit is always accompanied by the emergence of new primordia. This behavior is the reason why in most of the results we show here, primordia will exhibit varying lengths of observation, a situation that begs for robust time alignment. The rest of this section will present results of temporal alignment of time series of surface areas from a single plant, and then from mul-

tiple plants put together. In Fig. 2.15, we show an example of surface area growth pattern for one primordia. This figure shows the general pattern of primordia growth, which is multi-phasal. It is obvious from Fig. 2.15 that the sizes of the reconstructed surface area correlates with the calculated surface areas.

The SAM exhibits a great deal of variation within and across plants. As such, we tested our alignment algorithm on primordia from a single plant, as well as primordia from different plants put together. This experiment will test the stability of both the landscape based alignment and the parameter free alignment, which we call Minimum Mean of Distances alignment (MMD).

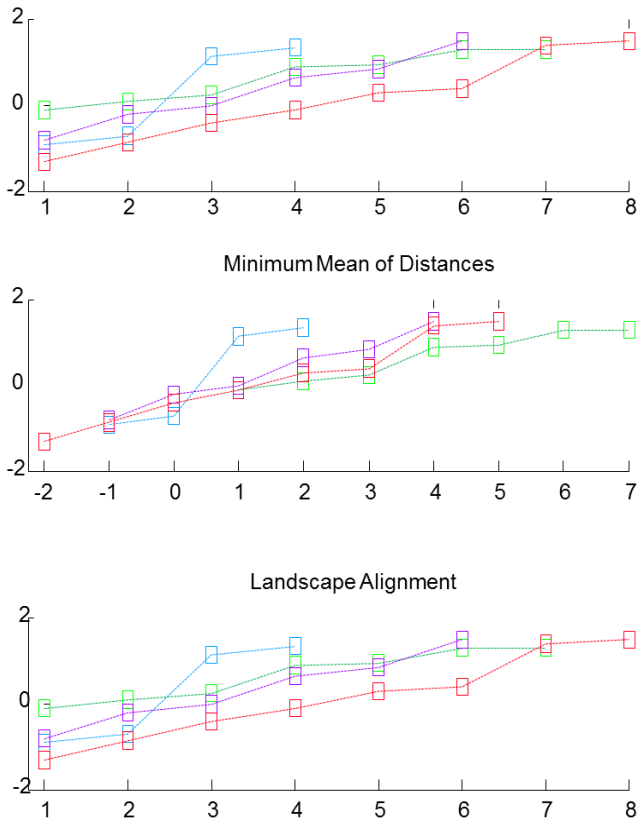

Fig. 2.16. Alignment of multiple primordia from a single plant. Raw growth curves (top). Alignment with parameter free algorithm labeled Minimum Mean of Distances (middle). Landscape based alignment (bottom). Even with a threshold parameter of 2, the landscape algorithm failed to find proper shifts.

2.7.3 Multiple Primordia from a Single Plant

The general nature of primordia growth and data collection introduces many opportunities for noise and outlier introduction. As such, the requirement for a threshold parameter is sometimes a limiting factor. As shown in Fig. 2.16.*bottom*, the alignment results from the landscape based alignment are no different from the original unsynchronized raw time

series. This sensitive exhibited by the LAM based alignment occurred in multiple other plants. We show another example in Fig. 2.17.

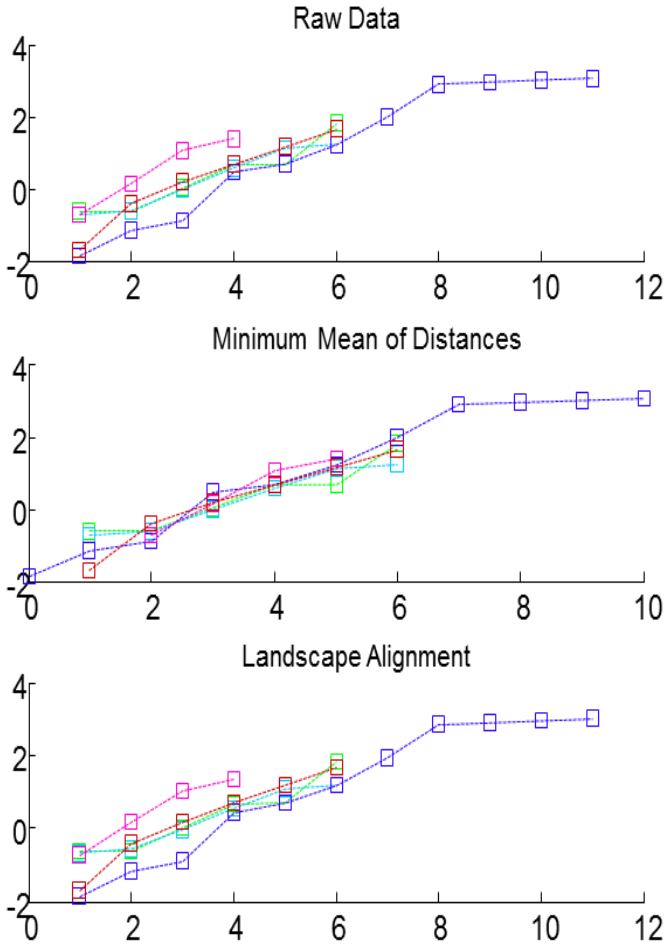

Fig. 2.17. Another example of parameter free Minimum Mean of Distances alignment being more robust than alignment based on landscape matching.

It is worth noting that LAM based alignment sometimes does yield reasonable results, especially in the presence of deformation. In Fig. 2.18, we show an example where both algorithms presented achieved reasonable alignments. Note that these results were ob-

tained after computing the corresponding deformation of the primordia in question and applying the multiple feature version of LAM.

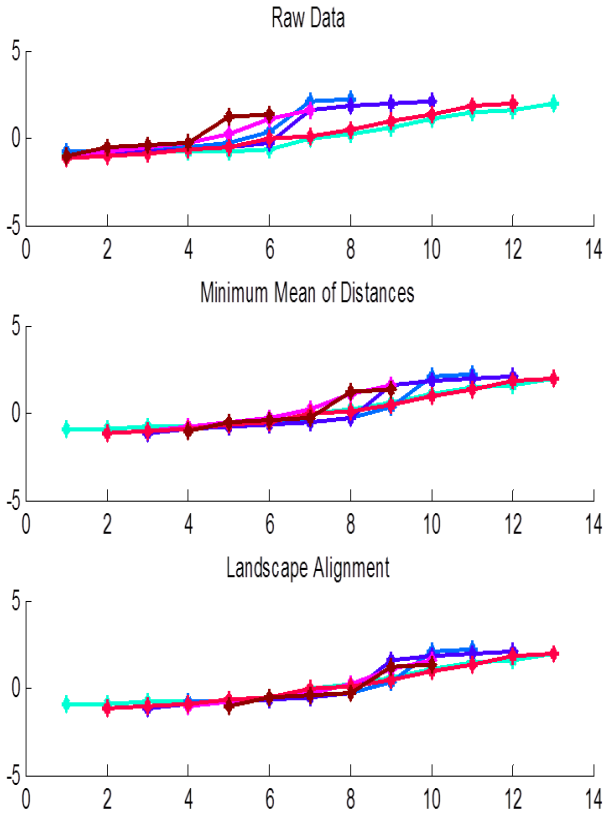

Fig. 2.18. Both algorithms yielding similar alignment results.

All these examples suggest that in the absence of a very strong second feature, our parameter free Minimum Mean of Distances (MMD) alignment algorithm is generally superior to the landscape based alignment. It is true that without a reasonable minimum subsequence length, MMD alignment could also fail, because it might elect very short

sequences that give the minimum score, yet in the context of surrounding points, does not make for a good match. In this situation and in the presence of a strong second feature, the LAM-M multiple feature alignment will be the preferred option.

2.7.4 Alignment of Multiple primordia from multiple plants

In order to test the robustness of our algorithm, we show performance of our alignment algorithm when we mix primordia from different plants. For this experiment, we aligned a total of 24 primordia, using both algorithms presented Fig. 2.19. Even with expected variation across plants, we show that a robust alignment routine can achieve an acceptable alignment.

These results are very significant, especially in the study of the principles that govern meristem growth/maintenance. As mentioned earlier, live imaging limits the number of markers we can use at any given instance. As such it requires many plants to study a few genes. Proper alignment is the only way we can begin any accross plants analysis. Based on these results, LAM based alignment schemes seem to be very sensitive to growth type data like the multi-phasal growth exhibited by primordia. LAM will work well for datasets like respiration data, since the patterns sort of periodic and variability between subjects is not as high as we see in meristem development. In such case it might be better to look at entire landscapes, rather than merely subsequences, even though one could argue that the space of overlapping landscapes is a subset of all possible subsequences.

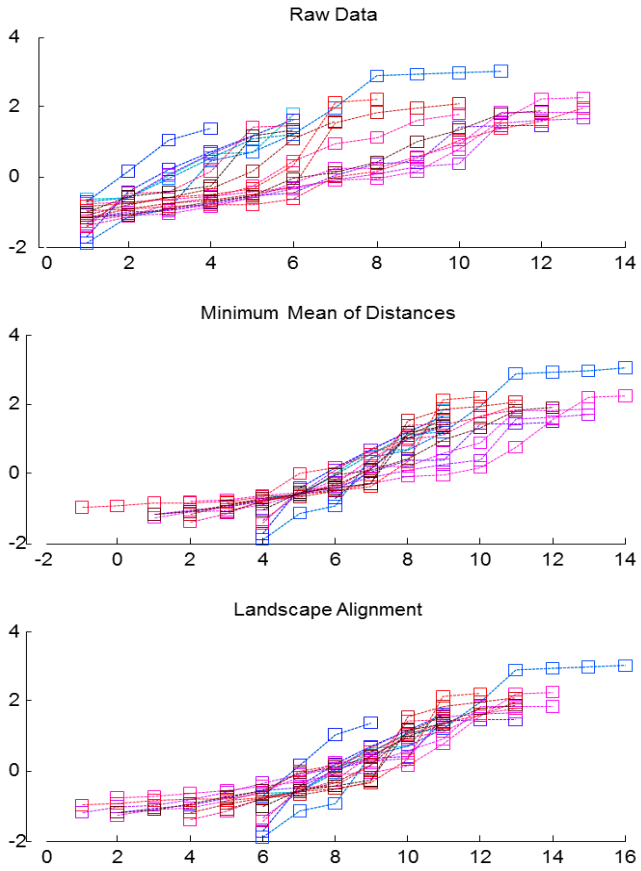

Fig. 2.19. Alignment of primordia collected from 5 plants. Total number aligned is equal to 24.

2.7.5 Results On Guppy Evolutionary Data.

The guppy data comes from a long-term project on rapid life-history evolution for which guppies were introduced in four isolated sections of streams in The Northern Range of the island of Trinidad. All guppies are marked with a unique combination of visible implant elastomer and are thus individually identifiable. The stream sections are censored monthly, when all individuals are caught, identified, and measured (standard length and weight), and new recruits are given a new unique mark. Only individuals above 14mm of

standard length are marked. Because birth is not observed, the age of individuals is unknown. A robust temporal alignment algorithm is therefore required for proper analysis. Data for this experiment was extracted from the time lapse average length measurements taken over a period of fifteen months. To show applicability, we randomly selected individuals, but made sure they were from the same cohorts or from cohorts that were not introduced into the ecosystem more than two months apart.

As we show in Fig. 2.20, the parameter free algorithm based on Minimum Mean of Distances achieved a very desirable alignment. Independent domain expert biologists who granted us the right to test on their data actually confirmed our alignment results as reasonable.

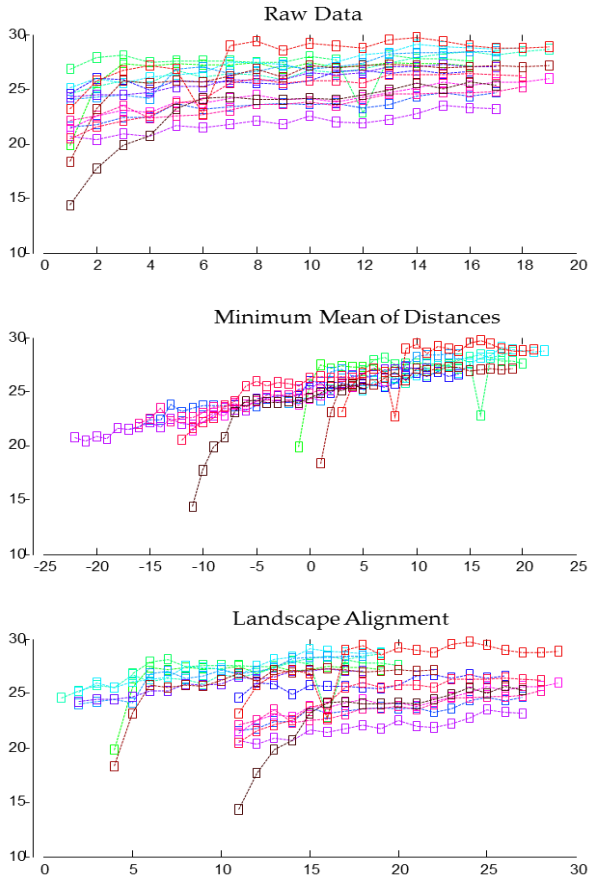

Fig. 2.20. Alignment of growth in length, of Trinidadian guppy. Each plot represents a single individual's growth pattern over time. The Y axis is the normalized lengths, while the x-axis represents time.

These results on guppy evolutionary data are significant, and present a new opportunity for biologist trying to study the effects of ecosystem conditions on adaptive evolution. Biologists now explore the possibility of estimating the age of individuals, or properly comparing individuals, in a way that is invariant to seasonal variation.

Given the variation in the data, it is no surprise that the parameter free MMD algorithm resulted in a more natural alignment. This is again another case where significant variation and noise poses a constraint on LAM based alignment.

2.8 Conclusion

We have proposed a framework for measurement of growth features and alignment of unsynchronized primordia growth data from live imaging experiments. To move closer to the big picture goal of developing a dynamic gene expression atlas, a necessary first step is the development of robust temporal growth alignment algorithms. We presented an extension to an already reliable alignment algorithm (LAM), as well as presented a parameter free alternative to growth alignment. The extension to the LAM algorithm to allow for multiple feature optimizations allowed us to overcome unique challenges presented in data with increased noise. However, this extension is sensitive to outliers and high variation. We have shown cases where LAM-M outperforms LAM basic, and also cases where even LAM-M fails, yet our parameter free algorithm succeeds. To demonstrate the diversity of our algorithm, we also showed results on guppy evolutionary growth data. This tells us that in the presence of high variation and increased noise, our parameter free MMD alignment algorithm is superior to LAM based alignment algorithms. A robust alignment of time lapse data gives developmental biologist the ability to begin to deduce the causal relationship between gene expression, cell behaviors and organ growth. Such an analysis would not be reliable without an alignment framework such as the one we proposed.

CHAPTER 3

CLUSTERING OF SYMBOLS USING MINIMAL DESCRIPTION LENGTH

The clustering of glyphs (individual letters/characters/symbols) is typically the first step in document processing algorithms and a critical enabling technology for most historical documents indexing techniques. Given the importance of the problem there has been significant research effort in the last decade on clustering techniques for atomic symbols. However, much of this work has focused on proposing novel distance measures which are purported to be best for a given language or type of text. In this work, we take a step back to consider the problem from domain and language agnostic perspective. In particular, we claim that, independent of the distance measure used, any method that attempts to cluster *all* the data is almost certainly doomed to failure. We explain this observation, and introduce a clustering method based on Minimum Description Length that can overcome it. We demonstrate the utility of our technique on diverse historical manuscripts.

3.1 Introduction

The clustering of glyphs (atomic letters/characters/symbols/graphemes) is often the first step in document processing algorithms [35][48], and an enabling technology for many Optical Character Recognition (OCR) algorithms. Recent commercial and government initiatives such as the Million Book Project and Google Print Library Project have placed tens of millions of pages of historical documents online and created an escalating need for automatic character recognition techniques that can deal with inevitable degradations and distortions encountered in historical texts. Most use (sometimes human assisted [35])

clustering as a preprocessing step in building a *character* (and hence, *word*) recognition system.

Given the growing importance of the problem, there has been a huge research effort in the last decade on clustering techniques for atomic symbols [35][48]; however, much of this work has focused on proposing novel distance measures which are purported to be best for a given language or type of text. For example, Dynamic Time Warping [48], Generalized Hough Transform [49], Mahalanobis distance, Blurred Shape Models [36] etc., all have their advocates. In some cases the distance measures are specialized down to the language [35][48] or style of writing (Fig. 3.1).

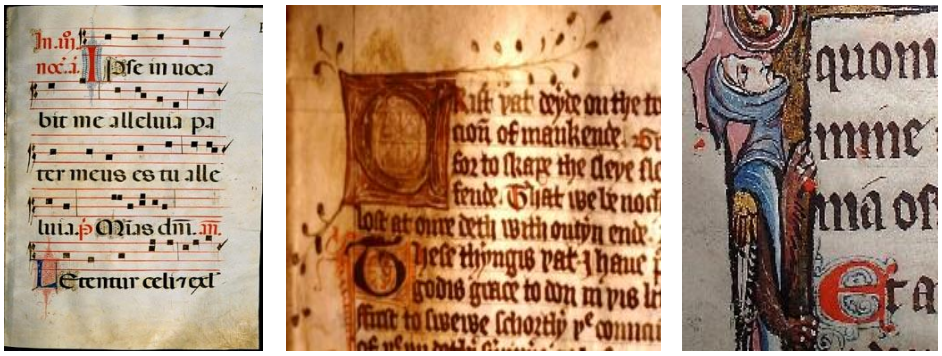

Fig. 3.1: Examples of early printed manuscripts. From left to right, lyrics and music of a Christian hymn, an early manuscript of uncertain provenience from the British library, a nearly illuminated New Testament.

In this work, we take a step back to consider the problem from a completely domain and language agnostic perspective. In particular, we claim that independent of the distance measure used, any method that attempts to cluster *all* the data is almost certainly doomed to failure. In other words, we must design clustering algorithms that are able to ignore

some of the data, in particular *Hapax legomenon*/rare words or badly degraded characters. This idea seems to open a "chicken-and-egg" paradox. When clustering, we need to ignore rare and unusual characters, yet we typically define rare and unusual characters with reference to some clustering. However, as we shall show, we can bypass this problem with a clustering method based on Minimum Description Length (MDL) [31][41].

The rest of this paper is organized as follows: Section 3.2 presents representation and clustering issues addressed in this work, Section 3.3 presents related work, Section 3.4 introduces our algorithm, including details on representation and the intuition behind the choice of MDL. Finally, we present our experimental results in Section 3.5.

3.2 Background

Perhaps the most important decision made in any pattern recognition effort is the choice of a representation for a given data set. This choice of representation for the data informs the distance measures and algorithms available for clustering, classification and query-by-content. In the next section, we explain why we opted to use a very simple representation.

3.2.1 Representation Issues

The last two decades have seen a plethora of proposed representations for glyphs, including chain codes [53], upper/lower profiles [51], Zernike moments [44][55],and various feature vectors/graphs/trees[35][30]. However, while an MDL approach (cf. Section 2.2) does not strictly *require* a simple representation; simple representations do facilitate a concise explanation and simple implementation. Moreover, there is increasing evidence

that both simple representations and simple distance measures can be *very* competitive [33].

Thus, in our research efforts we have adopted perhaps the simplest representation of glyphs, *bitmaps*. We do this with two caveats. First, as with most other glyph processing efforts, we first use an off-the-shelf binarization algorithm to produce black and white bitmaps. Second, as shown in Fig. 3.2, our algorithms will work directly with *downsampled* images.

Fig. 3.2: Two examples of the letter ã taken from a fifteenth century Flemish "Book of Hours". In both cases, we see the original data, the binarized image, and the downsampled binary image we propose to work with.

This step may seem counterintuitive. In many data mining and image processing algorithms downsampling is *only* used to make algorithms *efficient* [33], presumably at some cost to *effectiveness*. However, most modern digital archives of historical manuscripts are highly *oversampled*, and this spurious resolution does not help accuracy, and may even hurt it. To see this we conducted a simple test. We performed a leave-one-out classification experiment on a dataset of 52 symbols from 4 classes[1].We used the total number of differing pixel as the distance measure (cf. Section 3.4.1). We repeated this test at different sampling scales from the original resolution to a drastically undersampled 1/64 of the original resolution shown by inset images in Fig. 3.3.

[1]We defer a detailed discussion of the dataset until our experimental section. However, we note in passing that all experiments in this work are reproducible, and all code and data is available at [56].

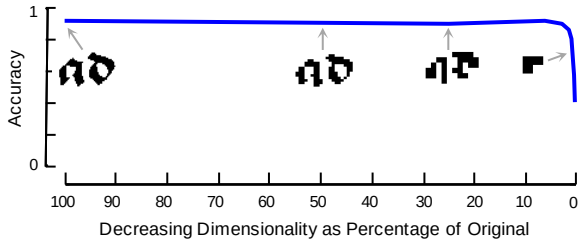

Fig. 3.3 : Nearest neighbor classification results at different sampling scales. The inset images show representative examples at four scales.

The results may perhaps be a little surprising. They show that we can *greatly* reduce the amount of information given the classifier without reducing accuracy. In fact, reducing the dimensionality of the data gives a slight improvement, presumably as the downsampling smoothes the data. Moreover, note that the accuracy reported is competitive with results published for this and similar datasets [45].

In retrospect however, this result is not too surprising. There is a large gap between *actual* versus *intrinsic* dimensionality of the data here. The actual dimensionality of the data depends on the photographic resolution, which, with modern cameras is essentially arbitrarily high. The intrinsic dimensionality of glyphs is more difficult to state, but is surer *much* lower. For example, industrial designers have long exploited the fact that all English alphanumeric characters can be represented with just a 5×7 grid [34]. Fig. 3.3 strongly suggests that we can comfortably work in a dimensionality much lower than the original.

We can cast more light on the results shown in Fig. 3.3. In Fig. 3.4.*top*, we show two symbols, A and B derived from highly stylized versions of the symbol ð (Latin small letter *Eth*) from a historical manuscript.

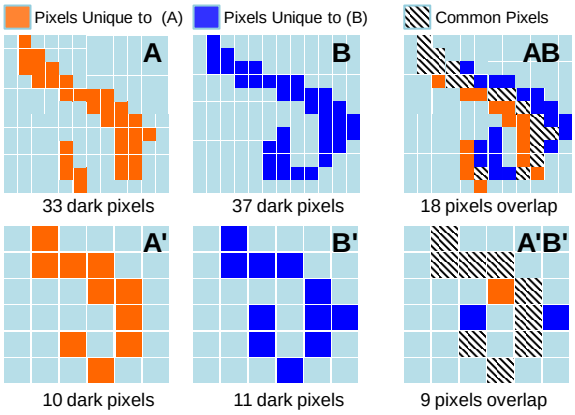

Fig. 3.4 : *top row*) Two examples of the same symbol. When compared (*far right*) they share only 54% of pixel locations. *bottom row*) The symbols can be compared at a lower resolution, where they have 80% of their pixels in common.

While clearly the same symbol, if we attempt to see how many pixels in common their best alignment has, we discover it is only 18 pixels, out of a maximum 33 possible. This 18/33 ratio would give us little confidence that they are the same underlying symbol, and make clustering such data difficult.

In Fig. 3.4.*bottom*, symbol A and symbol B are down-sampled to one-quarter their area, to produce symbols A' and B'. If we now test to see how many pixels in common they have, we find it is 8 pixels, out of a maximum 10 possible. This 8/10 ratio is more suggestive of the fact that the two symbols ultimately correspond to the same ground truth class label.

To summarize this section, empirically it seems that little, if anything is lost by working with simple and low-dimensionality data representations. Moreover, as we hinted here and will show more forcefully in Section 4, a simple pixel-level distance measure can be surprisingly competitive.

3.2.2 Clustering Issues

Virtually all clustering research in this domain uses K-Means clustering [46][40]. K-Means is attractive because it is simple, readily available and has certain performance guarantees (i.e., convergence). However, we believe that K-Means has two serious flaws for the task at hand. The first, which is very well known, is that it requires the user to predict the correct number of clusters. This is challenging in any domain, and particularly so in historical manuscript processing where we might expect 20 or more clusters.

The second weakness of K-Means is not well appreciated in the literature, but is a critical insight that motivates our work. We can demonstrate this with a simple experiment. In Fig. 3.5, we show a small real dataset. The data falls into three classes, although some of the badly faded ‘e’’s resemble the letter ‘c’, etc. Here both our proposed MDL clustering method (which will be explained in Section 3.4) and K-Means can both correctly cluster the data, although the latter must be told the number of clusters.

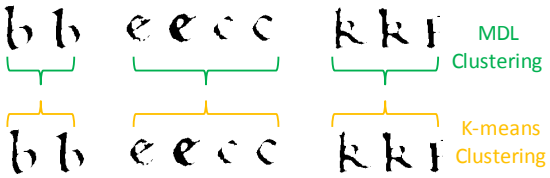

Fig. 3.5: A small dataset of faded letters, from three classes. Both our proposed MDL clustering method and K-Means can correctly cluster these, although the latter had to be told K = 3.

Suppose we repeat the experiment, but this time add a new letter, a ‘q’, to our dataset. Fig. 3.6 shows that the MDL clustering approach cannot fit the ‘q’ into any existing cluster, and thus decides to *ignore* it. In contrast, K-Means does not have the ability to ignore data. What will it do?

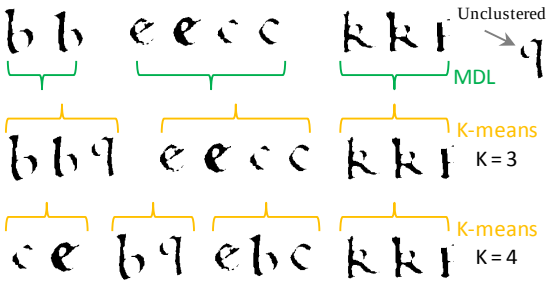

Fig. 3.6: *top*) Our MDL clustering algorithm can ignore data it cannot explain. In contrast, K-Means must explain all the data and for both K = 3 and K = 4 it does so badly.

First consider the case where we tell K-Means to use *three* clusters. In that case, it joins ‘q’, with the ‘b’’s to form the cluster {‘b’,‘b’,‘q’}. We would be disappointed to find such a diverse cluster, which clearly does not map to the objective truth.

However, we had given K-Means the wrong value for K. What happens if we give it K = 4? The answer is perhaps surprising. The output is much *worse*. Not only is ‘q’ not correctly placed in its own cluster, but it has caused *other* clusters to be incorrect. In particular, the two ‘b’s are split into two different clusters, and two ‘e’s are also fragmented into two clusters.

This problem is explained by two facts: K-Means cannot ignore data, and because of its objective function, the sum of *squared* distances, it is quadratically sensitive to even a

single outlier. To give an example in a domain familiar to most readers, if we attempt to spatially cluster almost any type of business/sports-venues/places-of-worship, etc in the forty-eight contiguous United States, the clusters are almost always intuitive and interpretable. However, if we add Hawaii to our dataset, the results are typically bizarre and unintuitive. While Hawaii is only a tiny fraction of the dataset, its great geographical distance from the rest of the data makes clustering with K-Means a challenge.

As hinted by this demonstration, we believe that the key to clustering, especially in domains characterized by highly skewed class sizes, is to allow the clustering algorithm the freedom to *ignore* some data. As we shall show, our algorithm has exactly this ability.

3.3 Related Work

Given the vast amount of literature on data clustering, we will not attempt a thorough overview of clustering. Instead we highlight some work that is closely related to our algorithm. For a detailed review of clustering, we refer the reader to [29][32] and the references therein.

We have designed our algorithm to exploit the advantages of the Minimum Description Length (MDL) principle. As we shall show later, this principle allows us to ignore symbols that do not belong in any cluster.

An MDL framework for data clustering was introduced in [41]. Although the basic intuition of MDL also motivates our algorithm, there is a fundamental difference. The algorithm presented in [41] has a goal of finding a clustering that maximizes the joint probabilities of the data, and involves initial clustering using parametric models. As such,

MDL is merely used as a model selection criterion. In contrast, in our proposed framework we derive an algorithm that is based solely on the basic MDL principle in the context of symbols clustering. Moreover, the representational power of our framework is much more expressive such that it does not *have to* account for all the data, but is willing to leave some data as "unexplained".

The MDL principle has also been applied in a variety of other related contexts, including as a step in a graphical symbol-learning algorithm [43], and as a model selection criterion in word clustering algorithms [42]. In the graph based symbol-learning algorithm, the MDL principle is applied as a tool for extraction of substructures from a constructed relational graph where each node is a *grapheme* [43]. However, this work is only defined for symbols captured with pen-stroke information (i.e., from a tablet device). Finally in [43], the use of MDL is a peripheral step in clustering, unlike our proposed algorithm which uses MDL *directly* to guide the clustering procedure.

3.4 MDL For Clustering Glyphs

In Section 4.2, we give a detailed visual intuition behind MDL for clustering glyphs, before formalizing our ideas in Section 4.3. First, we must consider some data representation issues.

3.4.1 Representational Preliminaries

To facilitate an intuitive description of our MDL method, we will use a toy data set *S*, made up of six characters as shown in Fig. 3.7.

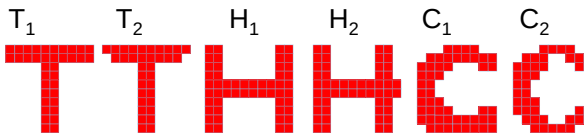

Fig. 3.7: A toy example of a dataset that will allow exposition of the MDL principle for glyphs.

Note that while this dataset is contrived for simplicity, it is at the approximate dimensionality we plan to use for clustering real data.

As noted above, our approach is based on the MDL principle [31][41][42]. MDL requires a data model that can encode the regularities within the data, such that the underlying cost of representation is minimized.

For glyphs that have approximately equal numbers of dark/light pixels, a simple binary matrix representation may be the most efficient encoding of the data. However, if the matrix is "sparse", as is the case for most glyphs, then it may be more efficient to encode *just* the locations of the dark pixels. This is depicted in Fig. 3.8 where we represent two of our toy examples this way. To locate each dark pixel we must encode its x-location and its y-location. As both have ten possible values, this requires $2 * \lceil log_2(10) \rceil = 8$ bits. For the 'T' illustrated in Fig. 3.8, which has 36 dark pixels, this means it requires 288 bits to represent.

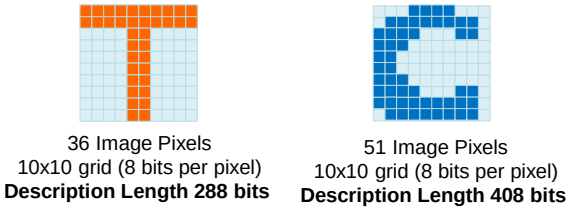

Fig. 3.8: Two symbols encoded by a list of pointers to their dark pixels

3.4.2 The Intuition behind MDL

Using the representational scheme discussed in the previous section it requires 2,120 bits to represent all six items in data set *S* (cf. Fig. 3.7). We can denote this as the Description Length (*DL*) of the data, using the notation $DL(S) = 2{,}120$ bits.

The reader will appreciate that our toy dataset contains significant redundancies. We may be able to exploit these redundancies to produce a more compact representation of the data, i.e. we may be able to *compress* it. Note that compression itself is not the end goal, the intuition behind MDL is that the model that compresses the data the best, also reflects the underlying structure, the *natural clustering* of the data.

For simplicity and ease of presentation, we have adopted the following encoding model: given two symbols, we encode the second one *given* the first one, simply by storing *just* the differences between the two symbols, together with a pointer to the location of the second.

In the terminology of the MDL, one symbol is a *hypothesis H* about the data. For example, the naive Description Length of the two symbols T_1 and T_2 (cf. Fig. 3.7) is 560 bits:

$$DL(\{T_1, T_2\}) = DL(T_1) + DL(T_2) = 288 + 272 = 560$$

But we can ask if it is more efficient to treat T_1 as hypothesis H_1 and encode T_2 given H_1:

$$DL(\{T_1, T_2\}, H_1) = DL(H_1) + DL(T_2 \mid H_1)$$

We can write this last term in a more general form:

$$DL(T_2 \mid H_1) = \alpha\beta + \rho,$$

where α is the number of *different* pixels between *hypothesis* H_1 and symbol T_2, β is the number of bits needed to encode a single location in our image grid, and ρ is the number of bits needed to encode a pointer to *hypothesis* H_1.

As shown in Fig. 3.9, $DL(\{T_1, T_2\}, H_1) = 288 + 2 \times 8 + 3 = 307$

Hence, this encoding saves us 253 bits. Note that to encode the second symbol in this manner, we need to indicate which image was the source image (the *hypothesis*). As there are six symbols in our toy example it takes 3 bits to encode this.

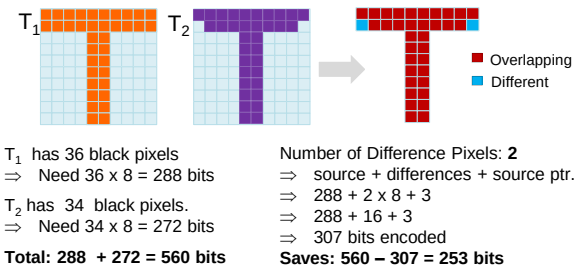

Fig. 3.9: An example of an encoding that leads to a savings of 253 bits

Note that we labeled our hypothesis with a subscript '1'. This is because in general there are many possible hypotheses about the data. For example, we may have (incorrectly) suspected it might be fruitful to encode T_1 with reference to another symbol C_1, let us call this H_2. As shown in Fig. 3.10, this is a bad idea. While $DL(\{C_1, T_1\}) = 696$ bits, we find the description length using H_2 is: $DL(\{C_1, T_1\}, H_2) = DL(H_2) + DL(T_1 \mid H_2) = 731$.

Thus, this unintuitive decision would have cost us 35 *extra* bits.

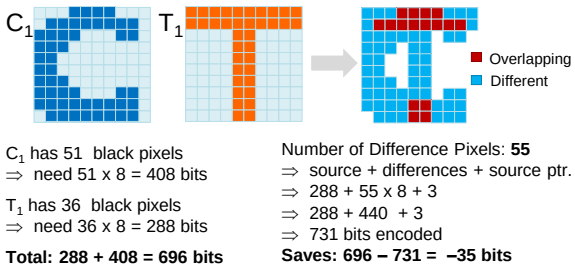

Fig. 3.10: An example of an encoding that leads to a loss of 35 bits. That is, the number of bits necessary to encode the data increases by 35 bits

These two simple examples contain the major intuition of MDL. The description lengths of rival hypothesis is a cost function that can be used to differentiate between poor decisions (combining C_1 and T_1 into the same cluster) and good decisions (combining T_1 and T_2 into the same cluster).

As we shall show more formally in the next section, we can exploit this cost function to allow a bottom-up approach to clustering, similar to bottom-up hierarchical agglomerative clustering algorithms [39].

3.4.3 Notation and description of the operators

We need to make some generalizations to produce a workable clustering algorithm. First, as the reader may have suspected, we may have multiple hypotheses about the data. Thus our final hypothesis may be a set of hypotheses such as $H_{\text{final}} = \{H_1, H_2, H_3\}$.

Creating a single hypothesis is naturally performed with a *'create'* operator. In our toy example, we had only pairs of symbols. However, we might have had say a dozen examples of the symbol 'T'. In this case, we want to form a single hypothesis from one 'T', and encode all the remaining examples relative to it one at a time. We can achieve this with an *'add'* operator.

Finally, we may find that at some point in the search process that it is fruitful to combine two hypotheses into one. For example, we may start our search with one hypothesis that represents the symbol 'I', and another that represents an italicized version of it '*I*'. Later in the search process, we may wish to overcome this greedy search decision with a *'merge'* operator.

With three available operators (*create*, *add*, *merge*), we can search the space of possible clusterings, using the description length of the hypotheses to guide the search. Thus, all that remains is to formalize these intuitions, which we do in the next section.

The input into the algorithm is a set of symbols (*S*) to be clustered, formally defined as:

$$S = \{s_1, s_2, \ldots, s_n\},$$

where n is the cardinality of *S*.

The output from the algorithm is a cluster set ***C***, defined as:

$$\boldsymbol{C} = \{c_1, c_2, \ldots, c_k\},$$

where *k* is the number of clusters in ***C***.

Note that *both* individual symbols and clusters are represented by lower-case italicized letters, because individual letters are a special case of clusters (i.e., clusters of size one). In fact, it is possible that our algorithm could find no exploitable structure in the data, giving us the pathological but legal case of ***C*** = *S*.

The description length of our output cluster set ***C***, is defined as:

$$DL(C) = \sum_{i=1}^{k} DL(c_i)$$

where k is the number of clusters in the set and each cluster c_i is a non-overlapping subset of *S*.

The ability to measure the *DL* of a cluster set gives us a primitive we can use to measure the number of bits saved (bitSave) after applying the *create*, *add*, and *merge* operators. The bitSave from creating a new cluster from two atomic symbols s_1 and s_2 to form a new cluster c' is defined as follows:

$$DL(s_1) + DL(s_2) - DL(c')$$

The bitSave resulting from adding a symbol s to an existing cluster c to get a new cluster c' is defined as follows:

$$DL(s) + DL(c) - DL(c')$$

The bitSave resulting from merging two clusters c_1 and c_2 into a new cluster c' is defined as follows:

$$DL(c_1) + DL(c_2) - DL(c')$$

The task of the *merge* operator can be cast as an optimization problem with the goal of finding the cluster set C' that has the minimum description length is defined as the following:

$$C' = argmin_C\{DL(C)\},$$

In order to merge clusters c_i and c_j into the same cluster, all of the following conditions must be true:

$$DL(c_i, c_j) < DL(c_i) + DL(c_j)$$

$$DL(c_i, c_j) \leq DL(c_i, c_x) \qquad \forall\, x > j$$

$$DL(c_i, c_j) \leq DL(c_x, c_j) \qquad \forall\, x > i$$

The decision to merge is driven by the goal of minimizing the description length of the final cluster set.

3.4.4 Algorithm in detail

Given the set of operators (*create*, *add*, *merge*), and the objective function (bitSave) our clustering algorithm simply reduces to a search algorithm. Because finding the optimal bitSave can be mapped onto finding the Kolmogorov complexity of an object, it is incomputable in general, thus we must be content with an approximation. Our proposed algorithm is a bottom-up search algorithm, which is very similar to hierarchical agglomerative clustering [54]. As shown in Table 3.1 the algorithm's input is a set of symbols, and the output is a set of clusters. We do not know in advance (nor can we control) the number of clusters that will be generated by the algorithm. The algorithm simply terminates when there are no more bits to be saved or in the pathological case that all objects are merged into a single cluster.

Table 3.1: MDL Clustering Algorithm

Input: *S* : Set of signals/images **Output**: *Clusters* : Final cluster set	
1	*Cluster* = {}
2	// now iteratively merge until *bitsave* ≤ 0
3	**while** *bitsave* > 0
4	*Remove_symbol* = {}
5	[*rm C' max_save*] =*AddNCreate*(*Cluster,S*)
6	*Remove_symbol = rm*

```
7     // now see what we save from merge
8      // will chose best option between this and AddNCreate
9     [C''  max_save''] =MergeAll(Cluster)
10     if max_save > max_save''
11        bitsave = max_save
12        Cluster = C'
13       S = S – S[Remove_symbol]
14     else
15        bitsave = max_save''
16        Cluster = C''
17     end if
18  end while
```

In describing our algorithm we will use a trace of the algorithm ran on our toy dataset. This step-by-step trace is shown in Fig. 3.11.

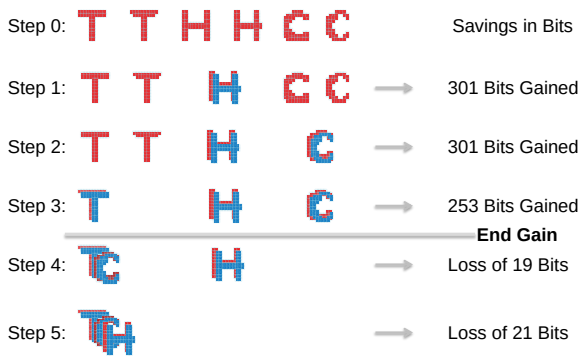

Fig. 3.11: A trace of our algorithm using an illustrative set of six characters. After step 3, the algorithm terminates since further clustering leads to an increase of bits needed to represent the data.

At the first iteration of the algorithm, we are only able to use the *'create'* operator, since all we have at this stage are atomic symbols. The algorithm considers all possible clusters that can be created and chooses to cluster the two symbols that save the most bits shown in Fig. 3.11.Step1. In Step2, we could either be creating a new cluster with two atomic

symbols or adding an atomic symbol to the cluster created in Step1. However, since creating a new cluster saves more bits than adding a symbol to the cluster created, the algorithm elects to create a new cluster in Fig. 3.11.Step2. As a result, although we have the *'add'* and *'create'* operators at this stage, the algorithm elects to use the *'create'* operator again because this operation leads to the most bits saved. The *'add'* or *'create'* operations are handled in Table 3.1.*line 5* of the algorithm by calling on the '*AddNCreate*' function shown in Table 3.2.

Table 3.2: *AddNCreate* Function

Input: *Cluster* : Cluster set
S : Set of symbols/images
Output: *Remove_symbol* : Index of symbols to remove
C' : Final cluster
max_save : Maximum savings so far

```
1   max_save = 0
2   C' = {}
3   for s1 ∈ S
4     for s2 ∈ S
5       new_clust = {s1,s2} // create a new cluster
6       c_save = DL(s1) + DL(s2) – DL(new_clust)
7       if c_save>max_save
8         C' = new_clust
9         Remove_symbol = S([g h])    // g,h is indices on S.
10          max_save = c_save
11      end if
12    end for
13    Merged_index = length(Cluster) + 1
14    for cs ∈ Cluster
15      t_clust = cs U {s1}
16      c_save = DL(cs) + DL(s1) – DL(t_clust)
17      if c_save > max_save
18        C' = t_clust
19        Remove_symbol = S([h])
20        Merged_index = m              // m is current index
21          max_save = c_save
```

22	**end if**
23	**end for**
24	**end for**
25	*Cluster*(*Merged_index*) = *C'*
26	*C'* = *Cluster*

In Fig. 3.11.Step3, we have all three operators available. After a call to the 'AddNCreate' function, we make a call to 'MergeAll' function in Table 3.1.*line 8*, passing in the current cluster set as an argument. The 'MergeAll' function shown in Table 3.3 merges two clusters that lead to the most savings in bits. After Fig. 3.11.Step3, we no longer have any atomic symbols present, and at this point, all we have is the merge operator. Note that in actuality, we could have more atomic symbols available. However, in this example, all atomic symbols are used up at Fig. 3.11. Step3. As shown in Fig. 3.11. Step4 and Fig. 3.11. Step5, further merging of clusters leads to a negative bitSave. The algorithm thus stops at Fig. 3.11. Step3 because at this stage, there is no gain in further clustering.

Table 3.3: *MergeAll* Function

Input: *Cluster* : Cluster set **Output**: *C'* : Possible Final Cluster set *max_save*: Maximum savings	
1	*C'* = *{}*
2	*merge_indices* = [ ]
3	*max_save* = 0
4	**for** $c_1 \in$ *Cluster*
5	**for** $c_2 \in$ *Cluster and* $c_1 \sim = c_2$
6	*new_clust* = $c_1 \cup c_2$
7	*c_save* = $DL(c_1) + DL(c_2) - DL($*new _clust*$)$
8	**if** *c_save* > *max_save*
9	*C'* = *new _cluster*
10	*merge_indices* = [i j] // i is current index
11	max_save = c_save

12	**end if**
13	**end for**
14	**end for**
15	*Merged_index* = *length*(*Cluster*) + 1
16	*Cluster*(*Merged_index*) = *C'*
18	*Cluster* = *Cluster* – *Cluster*(*merge_indices*)
19	*C'* = *Cluster*

3.4.5 Scalability

The time complexity of our algorithm presented in Table 3.1 is dominated by calls to the *AddNCreate* and *MergeAll* subroutines in Table 3.1.*line 5* and Table 3.1.*line 9*. We thus focus our asymptotic overview on these two functions.

The *AddNCreate* function consist of two main parts: creating a cluster from two atomic symbols and adding a symbol to a cluster. To create a cluster, we chose the pair of symbols that lead to the most bits saved. This requires checking all pairs of symbols, leading to a time complexity of $O(n^2)$ where n is the number of symbols in S. Note that in subsequent iterations, this step could be optimized by looking up best pairs from cached values. In fact, it is ideal to pre-compute these pairwise MDL distances, such that this part of the algorithm just becomes a simple table lookup. Also, using a memoization technique, by keeping the calculated values and repacking of the second best pairs and so on, we can reduce the number of calculations to $O(n)$.

Adding a symbol to an existing cluster requires looking at all symbol/cluster pairs. With k as the number of clusters so far, this process leads to a time complexity of $O(k*n)$, where k is never equal to n. The *AddNCreate* function has an order of $O(n^2) + O(k*n) = O(n^2)$. However, with a memoization technique, only the new/updated clusters need to calcu-

late/re-calculate their description lengths. Hence, the time complexity of *AddNCreate* reduces to order $O(n)$.

The *MergeAll* function selects the best pair of clusters to merge by looking at all pairs of clusters from the set of clusters so far, selecting the pair that leads to the most bits saved. This leads to an order $O(k^2)$ for the *MergeAll* operation, where k is the number of clusters so far. However, we need to calculate only pairs of the new/updated clusters to all others. Hence, the number of calculations of *MergeAll* is only $O(k) = O(n)$.

The complexity of the algorithm due to the *AddNCreate* and *MergeAll* operations is thus $O(n)$. Note that the most number of clustering operations that can be performed due to the while loop in Table 3.1.*line 3* is at most order $O(n)$. Consequently, the whole algorithm computes the description length at most $O(n^2)$ times, using memoization techniques.

3.5 Experimental Evaluation

As we noted in Section 3.2.1, all experiments presented in this work are completely reproducible. To make this possible, we will make all code and data available at [56].

Our two concrete goals are to show that our algorithm works on diverse datasets, and that it outperforms the K-Means algorithm, the most obvious choice of a strawman. To be fair to K-Means we give it the exact same input as our proposed algorithm, and we let it "cheat" by telling it the objectively correct value of K. We do not measure the speed of the algorithms, as the offline processing of historical manuscripts is not typically a time constrained problem. We note in passing that the time of our algorithm is not significantly worse than to K-Means, especially since K-Means requires multiple random restarts to mitigate against an unlucky choice of random seeds.

Most of the data used in this project was collected by Dr. Partha Pratim Roy and his team at Ecole Polytechnique de l'Universitè de Tours [47][52]. We gratefully appreciate their willingness to share his data. In Fig. 3.12, we show representative examples of the data we are clustering. The results of running our algorithm in comparison to using K-Means are shown in Table 3.4.

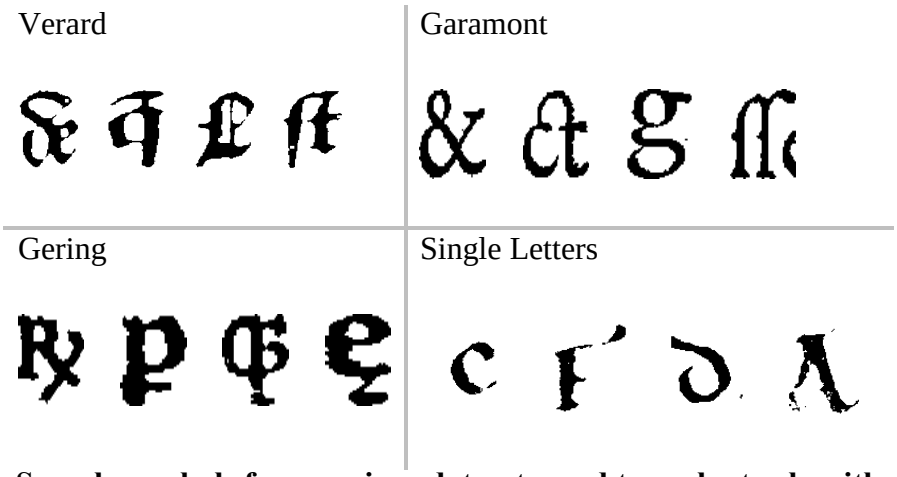

Fig. 3.12: Sample symbols from various datasets used to evaluate algorithm

3.5.1 Clustered Data

3.5.1.1 A complete trace on a small dataset

In Section 3.4.2, we presented a trace of the proposed algorithm on a toy dataset. Here we show how the algorithm does on a real dataset. In Fig. 3.13 we present the trace of the algorithm on a small set of degraded characters.

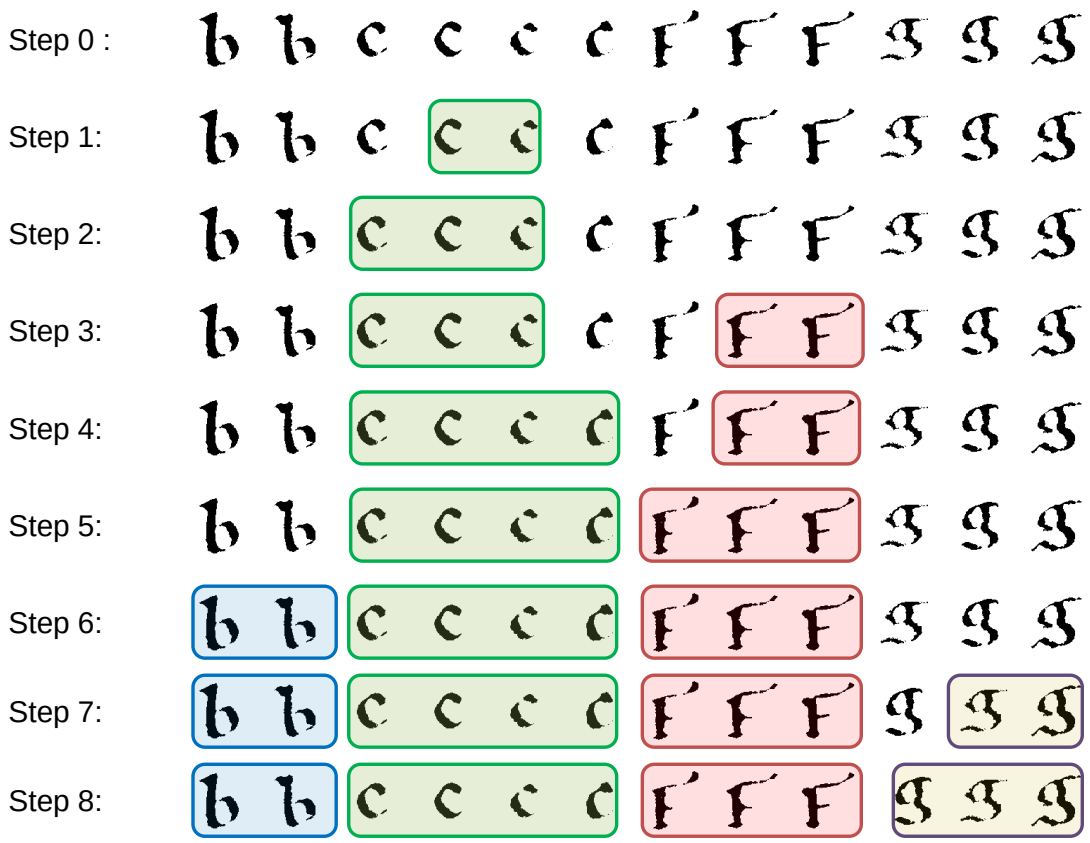

Fig. 3.13: A trace of our algorithm on a set of letters

As expected, the algorithm iteratively selects operations that lead to the most bits saved, and after the 8th iteration, it stops. This happens to be an example that nicely clusters into four groups.

3.5.1.2 Change in Description Length

As the MDL clustering algorithm evolves, the *DL* of the dataset continues to decrease. Each time we execute the *'Add'*, *'Merge'* or *'create'* operations, two things can happen: we either *gain* by needing fewer bits to encode the data, or *lose* by needing more bits to encode the data. Our use of the MDL principle requires us to opt for the choice where we gain through a decrease in *DL*. As a result, our algorithm terminates when further cluster-

ing does not lead to a savings in representation bits. When the *DL* of our dataset stops decreasing and begins to increase, we know we have reached a stoppage point.

In order to observe the evolution of description over time, we let the algorithm run beyond the stoppage point. Tracing the change in the *DL* as the algorithm evolves, the natural stoppage point becomes obvious. After the stoppage point in the algorithm, any symbol in the dataset that is not yet in a cluster at this point is simply just ignored. As shown in Fig. 3.14, this corresponds to the point where we have the best possible natural clusters. Beyond this point, we see a continuous increase in the DL of the data due to further clustering.

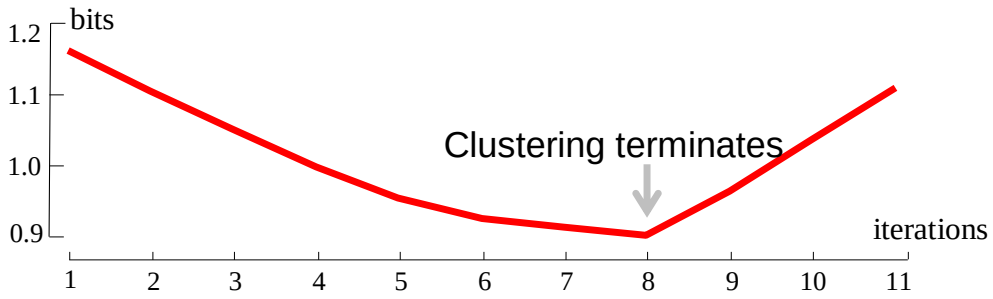

Fig. 3.14: Change in total description length of data in Fig. 3.13. Figure shows that after the 8th iteration, further clustering is useless.

Running the algorithm on the larger sample of this dataset made up of 142 letters achieved an ARI of 0.98. This is better than results we got from running K-Means on the same dataset, with the right k, which achieved an ARI of 0.784.

3.5.2 Comparison with K-Means

In this section, we show performance on old English characters and three font sets from [47]. A sample of the symbols in these datasets is shown in Fig. 3.12.

3.5.2.1 Experimental Design

In order to assure a fair comparison, we used the Adjusted Rand Index (ARI) as the metric for comparison. We also ensured that the basis of the distance measures in both algorithms is as similar as possible. Remember that *DL* in our algorithm is influenced the percentage of bits that differ between a symbol and its *hypothesis*. As such we ensured that the K-Means algorithm uses similar criteria in its distance comparison. For all experiments, we ran K-Means with the *hamming* distance, which is fitting, given that we are working with binary data.

As we stated in Section 3.2.2, K-Means has a weakness that requires to know the correct number of clusters, K. For a fair comparison, we eliminate this weakness by providing K-Means with the objectively correct value of K. Although this is cheating in favor of K-Means, we thought it will be the only fair way to truly test our algorithm.

Another weakness in K-Means is the fact that it does not ignore data and its objective function is very sensitive even to a single outlier. Therefore, in all our experiments, in addition to providing the right K to K-Means, we computed the ARI with the dataset as is (ARI K-Means) and with the dataset minus those symbols that were ignored by our MDL algorithm (ARI K-Means – Ignored Removed).

3.5.2.2 Performance Results

Table 3.4: Comparison of Clustering Performance

Source	Number of Objects	MDL Ignored	ARI MDL	ARI K-Means (right K)	ARI K-Means (Ignored Removed)
Vérard [47]	37	3	1.0	0.77 (K=8)	0.84 (K=8)
Garamont [51]	134	11	1.0	0.79 (K=18)	0.836 (K=18)
Gering [47]	121	14	1.0	0.85 (K=17)	0.86 (K=17)
Single Letters	142	20	0.98	0.784 (K=12)	0.873 (K=12)

In Table 3.4, we show the results from running the proposed MDL clustering algorithm, compared with running K-Means on the same data set. By cheating and giving K-means the right K, one would expect K-Means to give the best possible results. But as we show on the table, MDL Clustering outperforms K-Means in this data set. When we remove the MDL ignored symbols, K-Means had an improved performance, yet it still lacked behind MDL clustering.

3.6 Conclusion And Future Work

3.6.1 Conclusion

Clustering of symbols/glyphs is a critical subroutine in almost all historical document processing systems. As far as we know, MDL has never before been used in a task such as clustering of images. The fact that MDL is only defined for discrete data such as natu-

ral language or DNA strings is a major factor that has kept if from being applied to real-world data such as images. This is a major limitation given the fact that most images are intrinsically real-valued. Based on a fundamental premise that clustering algorithm be allowed the freedom of ignoring some data, in addition to a data representation that allowed us to exploit regularities in data, we have presented a practical application of MDL on real-world data like images. Our results compared to the popular K-Means algorithm show the MDL based algorithm is not only capable of ignoring outliers, it is also robust to outliers and produces much better clusterings, measured by the ARI of the clusters. This algorithm could have a significant impact in historical document processing systems.

3.6.2 Future Work

Although the results presented are impressive, we can even do better. The algorithm presented does not take advantage of image processing techniques that could significantly improve on the results. For example we use a simple center of mass spatial alignment before computing the *DL* between two images. However, such an alignment routine is not scale or rotation invariant. As such, the algorithm as presented does not deal with any invariance. The next logical step in this effort could be a generalization of the methods presented, including introduction of techniques to allow for both rotation and scale invariance.

CHAPTER 4

MINING STREAMING DATA FOR RARE AND APPROXIMATELY REPEATING REAL-VALUED SEQUENCES, USING SAX AND BLOOM FILTER

The frequent item mining problem continues to be an active area of research. The most common variant of this problem is that of finding discrete frequent items from an unknown alphabet, given an unbounded stream and bounded space. This variant of the problem is still an open problem and various approximate solutions have been proposed. In this work, we consider a much more difficult problem with potential impact in time series data mining. We seek to find *approximately* repeating items that may be rare, given an unbounded stream and bounded space. This is a more challenging problem given that we are dealing with real-valued time series streams, making it impossible to check for equality. To the best of our knowledge, all existing algorithms are designed only for discrete-valued streams. We present a novel algorithm that integrates Symbolic Aggregate Approximation (SAX) with a Bloom filter to build a robust cache maintenance policy that allows us to address known challenges to this problem. This novel integration allows us to implement an algorithm that selectively keeps possible candidates in cache longer, therefore increasing the probability of cache hit on subsequent occurrences of the rare candidate sequence. Results show a significant improvement compared to the alternative of a randomized cache maintenance algorithm.

4.1 Introduction

The frequent item mining problem is an area of research that continues to receive a lot of attention from the data mining community [61]. The problem comes with different variations and assumptions. However, it can be informally and intuitively stated as: Given a small amount of memory and an infinite stream, maintain a count of the most frequent items (or some other statistic) in that stream. The most common variant, the problem of finding discrete frequent items from an unknown alphabet, given an unbounded stream in bounded space, is known to be unsolvable in general (based on a simple adversarial argument [61]). Thus, many approximation algorithms for this problem have been developed through the years [61].

In this work, we consider a much more difficult problem in this research area. Instead of simply looking for *exactly* repeating frequent items, we present an algorithm to find *approximately* repeating items that may be rare, given an unbounded stream and bounded space. In its simplest form, given a time series subsequence from a data stream, we want to answer the question of whether or not we have ever seen such a subsequence before. Note that this is an unsolved problem in the *discrete* space; thus our problem formulation is much more difficult because we strive to solve this in the context of real-valued streaming data. We now give a formal description of the specific problem we seek to solve.

Problem Statement: Given a continuous time series stream that possibly goes on forever, we want to detect repeated subsequences in the stream. We assume the user will provide the target subsequence length n, and a threshold t is either given or can be learned

from the data. The value t is used to decide if two time subsequences can be considered similar and thus said to be repeated. Thus subsequences $S_{i:i+n}$ and $S_{j:j+n}$ are said to be repeated if $ED(S_{i:i+n}, S_{j:j+n}) \leq t$. In order to prevent trivial matches [65][66] where a subsequence is said to be similar to itself or where a subsequence overlaps with itself, we further constrain $|i-j| \geq n$.

To visualize this problem a little more clearly, we can imagine an adversarial game played against a "demon." The demon watches an unending data stream go past. Every now and then he inserts a subsequence corresponding to a particular behavior into the stream. We are able to see the stream a little further downstream and our task is simply to locate the demon's subsequence. We know nothing about this sequence, but are aware of the fact that it does exist, and if we are patient, more examples will be inserted into the stream. However, each time the demon inserts the subsequence of interest, he will add a little noise or distortions to it, making our task more difficult. For the sake of presentation, we refer to this subsequence of interest as a '*hopeful*' sequence.

Further complicating our task is the relative rarity of the target subsequence to the size of memory available to buffer the streaming data. For example, for every 10,000 sequences, we may only expect to see a target subsequence once. We refer to the average frequency of insertion of a target sequence as *f*. The reader can appreciate that our "malicious demon" model can model several interesting domains.

Given that these are real-valued sequences, this task presents the following problems:

1. Two real-valued items are never expected to be exactly equal, resulting in a difficulty of defining similarity. Thus, our algorithm needs to know the similarity thresholds within which two items are equal.
2. We cannot directly take advantage of frequent pattern mining algorithms like those reviewed in [61], given that they are only applicable to *discrete* data.

The first problem is not addressed in this paper. To test our algorithms, we assume that the threshold can be set, perhaps by previous experience with a similar problem.

For the second problem, we present a novel algorithm which incorporates SAX representation and Bloom Filter (BF) to provide a solution that is significantly better than a randomized algorithm.

4.2 Related Work

Our proposed algorithm makes use of the Bloom filter [59] and SAX (Symbolic Aggregate ApproXimation) time series representation [63]. SAX is needed to convert real-valued time series to their discrete symbolic representation. SAX is by far the most famous method for discrete representation of time series. SAX conversion leads to reduction in both dimensionality and numerosity. A Bloom filter is simply a space-efficient data structure (bit array), which uses a variable number of hash functions to hash elements into various index positions in the array. If an element hashes into an index position, the bits at the position are set to 1. We know an item has been seen before if it hashes to a bloom location that is already set. Bloom filters are widely used in the computer networks community [57]. In its general form, it has mostly been used to answer simple queries to determine membership in a class. Although the use/application of the bloom

filter is more common in the computer networks community [57], it is beginning to gain attention in the frequent items research space [60][65][67]. Most of these use some counter based variant of the Bloom filter. For example in [65], a counting Bloom filter was used to implement a 'top-k' items search algorithm. The algorithm in [65] relies heavily on the Bloom filter data structure for bookkeeping and estimating the 'top-k' frequencies, often requiring a second pass to choose the 'top-k' item set. Although their use of the Bloom filter is intuitive, their algorithm is not suitable to solve the problem we are solving. First, the algorithm was designed for discrete data, while we address the much more difficult problem of real-valued streaming data. Secondly, the algorithm in [65] limits the number of active counters at a time, which may pose a problem for unbounded streams, especially if the input query is simply to tell whether or not a particular item in the stream has ever been seen before. Just like in [65], the methods in [60] & [67] were all designed for discrete data and solve a much simpler problem in general.

4.3 Algorithm Details

We begin by stating the intuition behind our algorithm. First we consider the most naive algorithm for the task at hand. We could simply cache as much of the data as memory allows and inspect the cache for a pair of subsequences that are within the threshold t. This idea would require us to have a policy to discard one item from the cache at each time step, *just* before we ingest the next subsequence from the stream. Two possibilities for implementing this idea are Random discarding and First In First Out (FIFO). The general flow of this process is shown in Fig. 4.1.

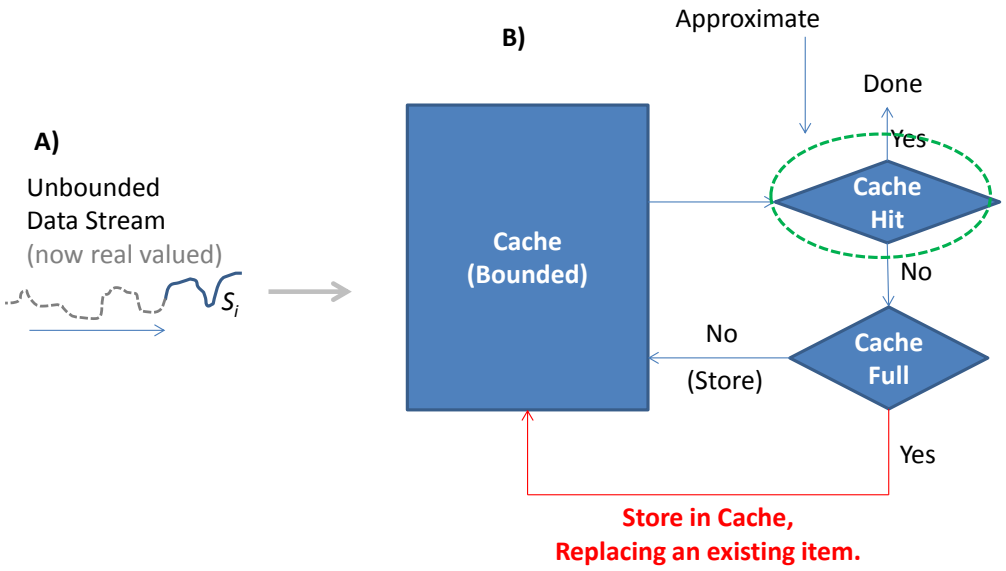

Fig. 4.1: High level illustration of the cache maintenance process. Cache replacement (red) is either done by random discarding or FIFO. We want to do better than randomized cache maintenance.

The solution we seek is therefore a cache maintenance algorithm at its core. Imagine our system as a 'black box' whose only task is determining whether or not an input sequence is in its cache. For now, there is no need to care about the internal structure of the cache. All we care is that this 'black box' has a cache with bounded space. It is therefore not possible to store all sequences from the stream until we get a hit. As shown in Fig. 4.1, given an input sequence S_i, the 'black box' initially checks to see if there is a cache hit. If the current query generates a hit, then the process stops and it returns the match. If there was no cache hit and there is space available in cache, it simply places S_i in the available slot. In case of a cache miss and a full cache, the 'black box' randomly discards one of the existing items in the cache to make room to store S_i. Note that a cache hit for real-valued sequences is not exact. It is merely approximate, or within some threshold.

As the reader will appreciate, the effectiveness of this algorithm depends critically on C, the size of the cache, relative to f. If $C \gg f$ then there is a high probability that the second time we see a target subsequence, we still have the first occurrence in the cache. Thus, as soon as we discover that they are within t of each other, we can report success. However, in the practical problems we hope to address, we expect that we have $C \ll f$. In such situations we can rarely expect to be so lucky as to have two examples of the target subsequence in memory at once.

Our intuition to mitigating this problem is to attempt to produce a cache replacement policy that is less likely to throw out a target subsequence than either the random discarding or FIFO replacement policy. Our idea is to use the bloom filter with the discretized (by SAX) subsequences to check to see if a newly inserted subsequence has been seen before. Because the discretization process of SAX is necessarily lossy, the bloom filter can have both false positives and false negatives[2]; however all that we are asking of the Bloom filter based cache replacement policy is that it biases us to be less likely to discard target subsequences.

In Fig. 4.2, we show a high level flow diagram of the proposed algorithm. The algorithm uses a bloom filter to determine whether or not a particular sequence is '*hopeful*'. A '*hopeful*' sequence is one that is more likely to be the 'malicious' sequence introduced in our demon model. In other words, it is likely to be our target '*rare*' sequence. Our use of the Bloom filter is to let us determine whether or not to store a cache item with a '*valid*' flag attached to it. A '*valid*' flag of 1 indicates that this is a '*hopeful*' sequence, preventing

[2] Note that if we were dealing with intrinsically discrete data, then a Bloom filter can only have false positives, not false negatives.

us from replacing it in subsequent cache replacements. Any cache item with a '*valid*' flag is not replaced because the flag indicates that it is probably a '*hopeful*' item. In Fig. 4.2, we also show the integration of a SAX conversion module before the Bloom filter. Note that SAX representation is for the purposes of Bloom filter utilization.

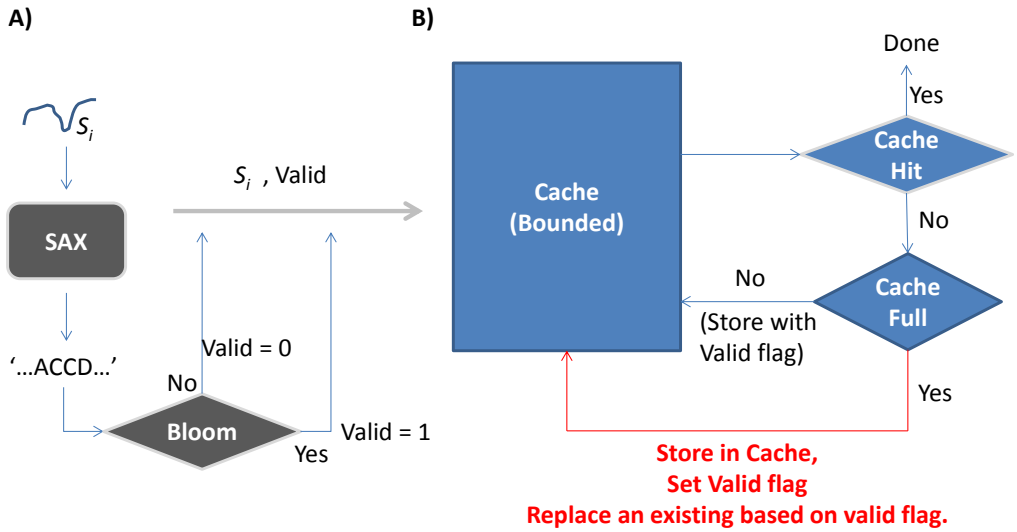

Fig. 4.2: High level flow diagram of the proposed algorithm. A) The input sequence is converted into a SAX representation and passed to the bloom module. B) The input to the cache maintenance is now the sequence and a valid flag, indicating whether or not this sequence is '*hopeful*'.

Bloom filters work best with discrete data and are not very useful with real-valued data. Therefore, before we check a sequence for a bloom hit, we first of all discretize it using SAX. However, in order to maintain accuracy, only the original real-valued sequences are stored in cache, and our check for a match is done on these original real-valued time series. This allows us to contain any unexpected error due to SAX limited to false positives on a Bloom check. Our implementation of the Bloom filter uses two hash functions. Our choice to limit the number of hash functions to two was motivated by research re-

sults reported in [58], where it was shown that two hash functions are sufficient to minimize the false positive rates of bloom filter queries.

4.4 Experimental Results

Data Generator: In order to simulate a streaming data environment, we developed a data generator that continuously produces two types of real-valued sequences to model our demon model: a random walk and sequences from the UCR gun data set with random noise added to avoid exact matches. As shown in Fig. 4.3, the data generator was set up such that the probability of generating a random walk sequence was 0.999, which means the probability of seeing a '*hopeful*' sequence is therefore only 0.001. In order to test the performance of our algorithm on discrete data, the data generator was also configured to generate discrete '*hopeful*' and random sequences. For all discrete experiments, the probability between 'hopeful' and random sequences/strings was the same as for real-valued data stream generation.

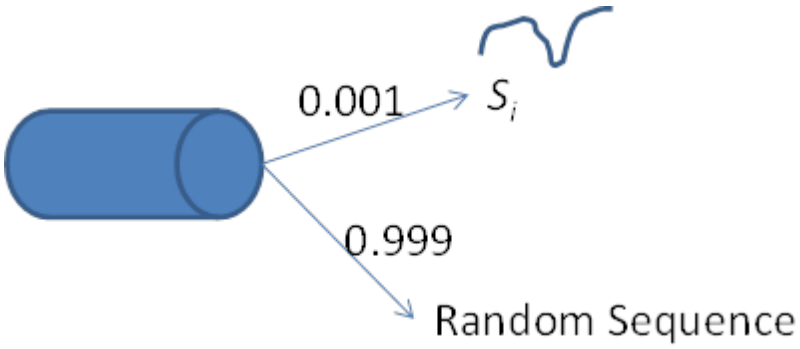

Fig. 4.3: Data stream generation process. The stream used for all experiments is such that the probability of a 'hopeful' sequence is 0.001. This setup allows us to make sure the 'hopeful' sequences are truly rare.

In all the experiments, we ran our algorithm and the randomized algorithm, comparing the results to see how well a Bloom filter based algorithm could do. We also tested both algorithms on discrete and real-valued sequences. Note, however, that with discrete data, we do not need to transform the sequences using SAX for Bloom integration.

We measured performance in terms of the number of input sequences ingested from the stream, which we refer to as 'Time Steps.' We also measured performance in terms of the number of times we see a '*hopeful*' sequence before we get a cache hit. We ran the experiments on cache sizes of 100, 50, 25 and 10. The experiments were run 500 times per cache size, saving the number of misses and time steps per run. We then used this data to look at their behavior under different values of C.

4.4.1 Experiment on Discrete Data

Experiments on discrete data are meant to highlight the major improvement of the Bloom enabled cache maintenance approach, compared to the randomized cache replacement algorithm. In these experiments, we configured the data generator to generate discrete strings with a cardinality of 150 and a dimensionality of 4. Each element in a sequence can be any of the characters {0 1 2 3}. Each generated string sequence from the stream has a 0.999 probability of being a random string generated from the character set and a 0.001 probability of a constant string constructed from repeating patterns of '0123' up to the cardinality of 150. Given that discrete data does yield an exact match, we set the threshold to 0. The size of the Bloom filter was set to 4^10.

4.4.1.1 Distribution of time steps before Cache Hit.

This experiment was designed to get a clear picture on the average number of time steps needed before we get a cache hit. Understanding that the number of time steps is constraint by the probability of getting a '*hopeful*' sequence, we ran the experiment 500 times and plotted the distributions as histograms for both the Bloom based algorithm and the randomized cache maintenance algorithm. As shown in Fig. 4.4, the average number of time steps for the Bloom based algorithm is consistently better and almost remains constant regardless of C, compared to that of the randomized algorithm. This experiment suggests that the number of time steps takes on an exponential distribution. As seen in Fig. 4.4, the Bloom based algorithm maintains this property even when the cache size C is reduced to 10, while the randomized algorithm slowly deviates from this property to a more even distribution. It seems like the general behavior is exponential, with a higher rate and consistent rate in the case of the Bloom filter based algorithm.

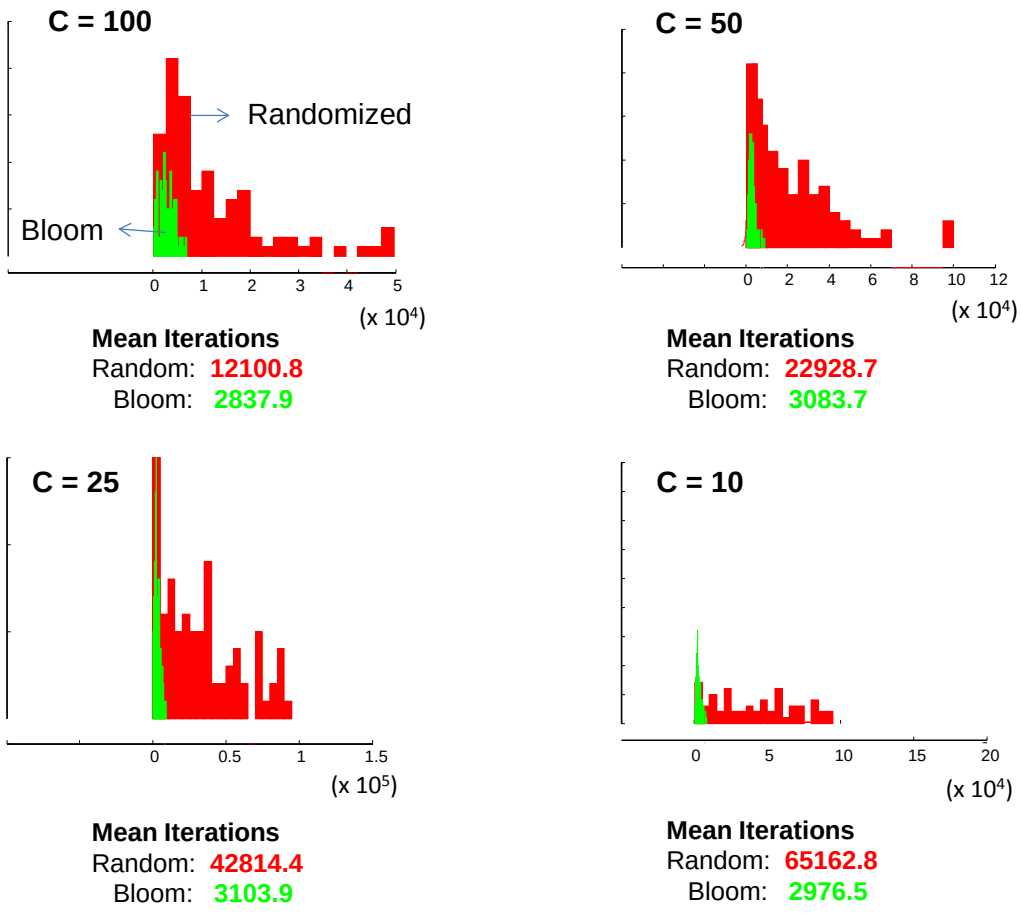

Fig. 4.4: Distribution of time steps required for cache hit over different cache sizes.

4.4.1.2 Number of misses before a Cache Hit

Since it is possible to argue that the results of time steps distribution can be affected by chance, given *f*, we decided to measure performance by looking at the number of times we see a true '*hopeful*' sequence before a cache hit. This metric tells us how effective our cache replacement policy is. As shown in Table 4.1, a Bloom enabled cache replacement algorithm sees a '*hopeful'* at most three times before a cache hit when dealing with discrete data. This is significantly better than the random cache maintenance alternative.

Table 4.1:Cache miss rates between the bloom and randomized algorithms on discrete data. The bloom cache maintenance shows almost a constant miss rate, irrespective of the cache size.

Cache Size	Random Cache Maintenance	Bloom Cache Maintenance
100	12	2.8
50	23.2	3
25	41.5	2.9
10	66.5	3

4.4.1.3 Empirical Cumulative Distribution

This distribution allows us to measure the number of time steps we must wait in order to attain a high probability of finding a '*hopeful*' sequence. As shown in Fig. 4.5, the bloom based algorithm does not vary much with decreasing cache sizes in the discrete data.

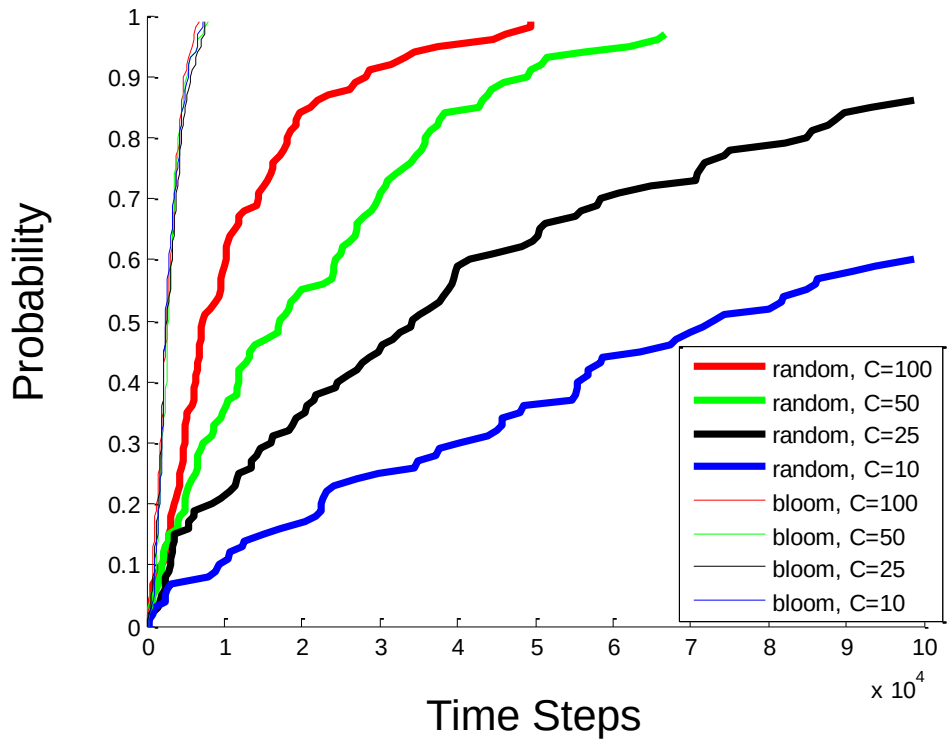

Fig. 4.5: Number of time steps necessary for high cache hit probability. Different colors indicate different cache size, C. Thick plots represent randomized algorithm.

This can be explained with the fact that the Bloom filter allows us to selectively perform our cache replacement task. It kind of allows us to extend the memory of the system, making it possible to remember sequences that appear more than twice. Even with the extraordinary rareness of our '*hopeful*' sequences, the behavior of the Bloom based cache maintenance algorithm is very consistent in the discrete case, even when we reduce cache size to 10. The only thing left to show now is how this cache maintenance policy behaves with real-valued data.

4.4.2 Experiment on Real-Valued Data

For this experiment, the data generator was configured to either produce a sequence from the UCR gun data set or a random walk. Each sequence from this stream had a 0.999 probability of being a random walk. The size of the Bloom filter for these experiments was kept the same as that for the experiment on discrete data. The SAX conversion was done with an alphabet size of 6 and a cardinality of 12.

4.4.2.1 Distribution of time steps before Cache Hit

As shown in Fig. 4.6, the Bloom-SAX cache maintenance algorithm is a significant improvement compared to the randomized approach. The distribution of the number of time steps before a cache hit for the Bloom-SAX algorithm maintains its exponential property, even when C is reduced to 10. This is not true for the time steps distribution from the randomized algorithm (red histograms).

In this experiment, we expected the SAX conversion to have a slightly negative effect on performance, given the generally lossy nature of SAX conversion. The effect of this con-

version, as shown in Fig. 4.6 is such that although Bloom cache maintenance is consistently better than randomized cache maintenance, the algorithm shows increased spread and general increase in mean time steps as cache C reduces to 10. Besides maintaining a much lower mean number of time steps across different C's, the Bloom-SAX cache maintainence generally maintains an exponential distribution. On the otherhand, the randomized cache maintenance algorithm quickly deviates from an exponential distribution as C goes down to 10. However, it is clear that Bloom-SAX integration for streaming sequence cache maintenance significantly improves performance even when $C \ll f$.

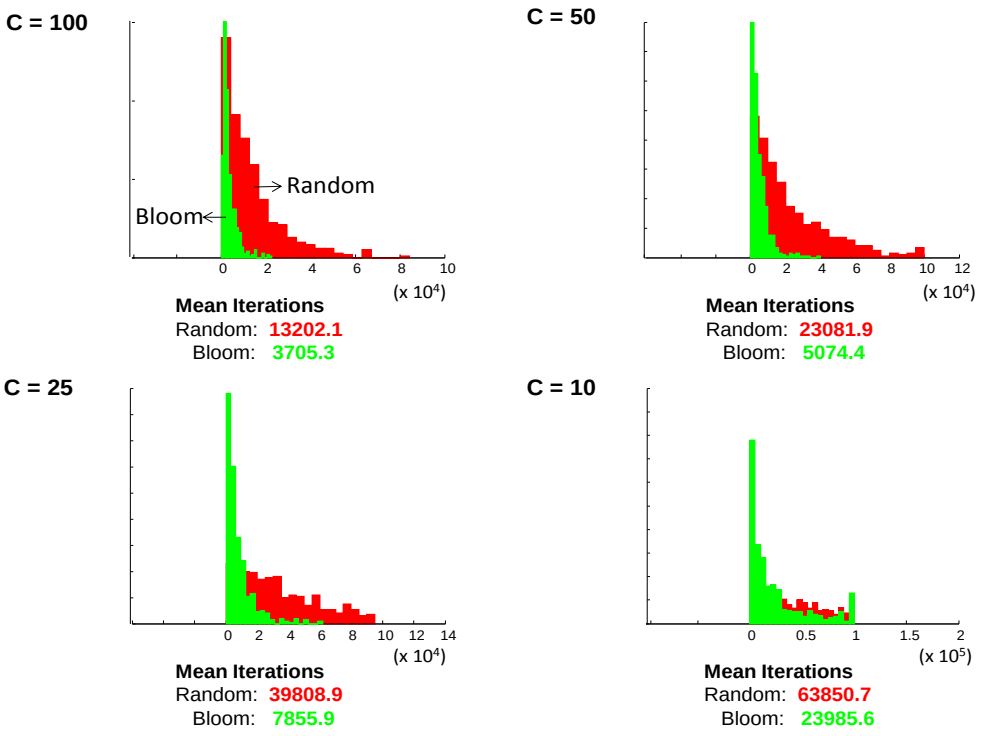

Fig. 4.6: Time steps distributions between bloom and random cache maintenance algorithms on real-valued sequenes. The distributions from Bloom-sax cache maintenance are shown in green. The average numbers of iterations (time steps) are also shown beneath each plot.

4.4.2.2 Number of misses before a Cache Hit

We now present the number of misses before a cache hit both for the randomized algorithm and the Bloom-SAX integrated algorithm. Despite the generally lossy nature of SAX conversion, the Bloom-SAX integration for cache maintenance of streaming real-valued sequences significantly minimizes the number of misses (see

Table 4.2). Note that cache hit comparisons are done with the real (original) time series. At C = 100, 50 or 25, the performance is comparable to that in the experiment on discrete data which is very impressive. At C = 10, the number of misses is slightly higher, but acceptable, given that it is significantly better than the number of misses due to running the randomized algorithm.

Table 4.2:Number of misses before a cache hit for real-valued streams between Bloom based algorithm and randomized algorithm.

Cache Size	Random Cache Maintenance	Bloom-SAX Cache Maintenance
100	13.1	3.8
50	23.2	5.2
25	39.9	8. 2
10	63.6	23.7

4.4.2.3 Empirical Cumulative Distribution at Varying Cache Sizes

In Fig. 4.7, we show the cumulative distribution curves of the time steps distributions between the Bloom-SAX algorithm and the randomized cache maintenance algorithm. As expected, the growth rate of the exponential cumulative distribution curves is much higher with the Bloom-SAX results (thin curves), while the results from the randomized alternative become less exponential as C goes from 100 to 10.

Even with C = 25, Bloom-SAX does better than random discarding with C = 100. This means even with four times less cache, the Bloom-SAX algorithm is still a significant improvement from the randomized alternative.

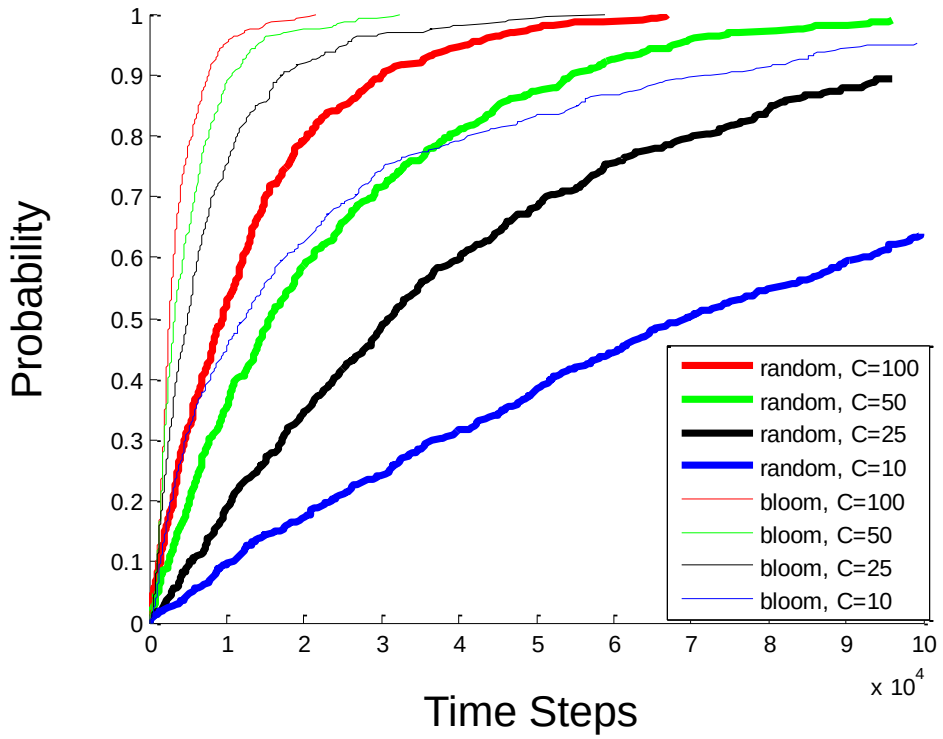

Fig. 4.7: Empirical cumulative distribution from real-valued sequences illustrating the number of time steps required to have a high probability for a cache hit.

4.5 Conclusion

Detecting 'rare' and approximately repeating items from an unbounded stream with finite memory is a difficult problem that was previously unsolved. Most existing solutions are geared towards discrete sequences, where exact solutions are possible. However, most real-world data is real-valued and present a major challenge mostly due to the fact that an

exact solution does not exist. In this work, we have presented a novel integration of the famous SAX time series conversion technique, with the Bloom filter, to build a robust cache maintenance algorithm that overcomes known challenges. Results show a major improvement over the alternative random discarding cache maintenance algorithm. Our contribution also lies in the fact that we solve this problem on real-valued data, whereas only the discrete-valued case has been considered in the literature.

CHAPTER 5

CONCLUSION

Based on the methods/algorithms developed, this dissertation made contributions to interdisciplinary research in developmental plant biology, historical document processing and data streams processing.

This dissertation presented image and time series algorithms with practical implications in developmental plant biological research, historical document processing and mining of unbounded data streams. The body of work presented embodies the spirit and goals of the NSF IGERT program and prepares me for a future in interdisciplinary and translational data mining. Methods for computation of growth features from real-time, live imaging data were presented, including an application of deformation field morphometry for the computation of surface deformation from reconstructed time lapse data. Two temporal multiple time series alignment algorithms for growth data were also presented. Next, a practical application of the MDL principle to the clustering of real world data, like images was also presented, including the basic premise that a clustering algorithm should *NOT* be forced to account for all the data it is given. It should be given the freedom to ignore some data. Finally, this dissertation presented an algorithm to solve a previously unsolved problem in frequent item mining research. It presented a novel algorithm for detecting 'rare', but approximately repeating sequences in real-valued data streams. The significant performance improvement due to the integration of SAX and the Bloom filter is a significant contribution to frequent items mining research.

Overall, the body of work presented here made significant contributions in interdisciplinary research and will continue to promote basic scientific research for years to come. The work also made contributions to and stands to have an impact in the specific discipline of computational sciences.

5.1 Summary of Contributions

This dissertation presented methods and algorithms for the quantitative study of growth dynamics at the Shoot Apex of Arabidopsis *thaliana.* The study of molecular control at the Shoot Apical Meristem (SAM) of model plant Arabidopsis *thaliana* requires a strong understanding of the causal relationship between gene expression, cell division, cell-cell communications, and organ growth. Live imaging has given scientists the ability to capture in real-time, the dynamic context of organ growth at the resolution of the cell. However, given the technical limitations of this technology, only a few genes (usually about 2) can be observed at a given time. As a result of this limitation, studying molecular control requires the spatial mapping and temporal alignment of different functional domains into a single template, development of dynamic models, and eventually a dynamic gene expression atlas. These tasks are not possible without robust temporal alignment algorithms. For this reason, two growth alignment algorithms were presented. The first one extended a well-known Landscape Matching algorithm, giving it the ability to support multiple features. The second algorithm was essentially parameter free and proves to be more robust in the presence of high noise. Applicability to other data sets was tested through the alignment of growth data from the *trinadadian guppy*. Besides these temporal alignment algorithms, this dissertation also presented methods for measuring growth from time

lapse microscopy data collected using live imaging experiments. In this effort, we show (perhaps for the first time) how to apply deformation field morphometry for growth measurement in quantitative plant development studies.

Given the growing importance of algorithms for clustering symbols in almost all document processing systems, this dissertation also presented an algorithm that tackles this problem totally from a domain and language agnostic perspective. In doing so, a practical application of MDL principle on real-world data, like images, is presented. Not only is this clustering algorithm important because individual character clustering is the first and most critical task in any higher level document processing system, it is also important because it overcame a key limitation to the application of MDL to real-world data, like images. For this reason, and the basic premise that clustering algorithms should have the freedom to ignore some data, this algorithm has the possibility of having an impact beyond the task at hand.

Finally, the world is full of growing sources of streaming data. A majority of this streaming data is real-valued. As with anthing else, things change over time, and the events that trigger specific behaviors are sometimes rare, yet repeating. When such events happen, we want to be able to notify an expert to make a decision or take a closer look. This could be in the medical space with patient respiratory data, in the financial space with continuous flow of financial data, etc. Rare, yet significant events could easily be thrown out as noise. Detecting such events is usually a difficult task (generally unsolved), given that most of this streaming data is real-valued and there is no way to accurately check for equality. Secondly, the amount of streaming data is overwhelming, relative to

the size of available memory. Because it is impossible to remember every piece of data point that has ever been seen in a stream, randomized cache management algorithms are the most feasible options. This dissertation presents an algorithm that perhaps for the first time, proposes a solution to cleverly *'extend the memory'* of any streaming data processing system and improve the ability to detect these rare events *'as they happen.'* For the first time, a novel integration of SAX time series representation and the Bloom filter in a robust cache management algorithm is presented. Results show major improvement compared to the randomized alternative.

5.2 Future Work

The body of work presented is compelling and makes significant contributions to the body of knowledge. It also paves the way for many opportunities in future research.

In Chapter 1, we presented two growth alignment algorithms, motivated by the big picture goal of developing a dynamic gene expression atlas for the SAM of Arabidopsis *thaliana.* To get even closer to that goal, a logical follow up to the alignment methods provided is the development of a growth model based on quantitative measures of growth features and temporal alignment. Such a dynamic model could leverage the work in [13] to incorporate the effect of local cell division patterns on organ growth. Development of this tool will give biologists a tool to explore the quantitative relationship between cell division patterns and organ growth. Such a model, even though it does not include gene expression, will be a major step towards the ultimate goal.

In Chapter 4, we presented a powerful new approach to cache maintenance in frequent item mining. In all the experiments presented, we computed parameters for the Bloom

filter and SAX offline, including the threshold for similarity. This was sufficient to demonstrate the power of a new idea. However, the general nature of the type of sequences we target is such that we do not know the characteristics ahead of time. Therefore, a good open problem is that of dynamic parameter generation. In relation to the threshold, this would include learning the possible threshold from the data. SAX and Bloom parameter generation will require a modification of the algorithm to allow for adaptive parameter settings through multiple re-starts. A solution to these challenges will pave the way for translational researchers to incorporate this powerful idea into real-world systems.

REFERENCES

[1] Braude, I., Marker, J., Museth, K., Nissanov, J., and Breen, D. Contour-based surface reconstruction using mpu implicit models. Graphical Models (2007).

[2] Chen, L., McKenna, T., Gribok, A., and Reifman, J. Lam: A landscape matching algorithm for respiratory data alignment. SPIE The International Society for Optical Engineering (2006).

[3] Christin, C., Hoefsloot, H., Smilde, A., Suits, F., Bischoff, R., and Horvatovich, P. Time alignment algorithms based on selected mass traces for complex lc-ms data. Journal of proteome research (2010).

[4] Chung, M., Worsley, K., Robbins, S., Paus, T., Taylor, J., Giedd, J., Rapoport, J., and Evans, A. Deformation-based surface morphometry applied to gray matter deformation. Neuroimage (2003).

[5] Dumais, J., and Kwiatkowska, D. Analysis of surface growth in shoot apices. The Plant Journal (2002).

[6] Eid, A., and Farag, A. On the performance evaluation of 3d reconstruction techniques from a sequence of images. EURASIP J. Appl. Signal Process. (2005).

[7] Fernandez, R., Das, P., Mirabet, V., Moscardi, E., Traas, J., Verdeil, J., Malandain, G., and Godin, C. Imaging plant growth in 4d: robust tissue reconstruction and lineaging at cell resolution. Nature Methods (2010).

[8] Gaser, C., Nenadic, I., Buchsbaum, B., Hazlett, E., and Buchsbaum, M. Deformation-based morphometry and its relation to conventional volumetry of brain lateral ventricles in mri. NeuroImage (2001).

[9] Guilleminault, C., Poyares, D., Rosa, A., and Huang, Y. Heart rate variability, sympathetic and vagal balance and eeg arousals in upper airway resistance and mild obstructive sleep apnea syndromes. Sleep Med (2005).

[10] Keogh, E., and Pazzani, M. Derivative dynamic time warping. First SIAM International Conference on Data Mining (2001).

[11] Khairy, K., and Keller, P. Reconstructing embryonic development. Genesis (2011).

[12] Listgarten, J., Neal, R. M., Roweis, S. T., and Emili, A. Multiple alignment of continuous time series. In Advances in Neural Information Processing Systems (2005).

[13] Liu, M., Yadav, R., Roy-Chowdhury, A., and Reddy, G. Automated tracking of stem cell lineages of arabidopsis shoot apex using local graph matching. The Plant Journal (2010).

[14] [14] Lorensen, W., and Cline, H. Marching cubes: A high resolution 3d surface construction algorithm. SIGGRAPH Comput. Graph. (1987).

[15] P.J. Besl., and N.D. McKay. A method for registration of 3-d shapes. Trans. PAMI (1992).

[16] C. S. Myers and L. R. Rabiner. A Comparative study of several dynamic time-warping algorithms for connected-word recognition. The Bell system technical journal (1981).

[17] Pieperhoff, P., Su¨dmeyer, M., Ho¨mke, L., Zilles, K., Schnitzler, A., and Amunts, K. Detection of structural changes of the human brain in longitudinally acquired mr images by de-

formation field morphometry: Methodological analysis, validation and application. NeuroImage (2008).

[18] Rakthanmanon, T., Keogh, E., Lonardi, S., and Evans, S. Time Series Epenthesis: Clustering Time Series Streams Requires Ignoring Some Data. ICDM (2012).

[19] Reddy, G., and Meyerowitz, E. M. Stem-cell homeostasis and growth dynamics can be uncoupled in the arabidopsis shoot apex. Science (2005).

[20] Reddy, G. V. Real-time Lineage Analysis reveals Oriented Cell Divisions Associated with Morphogenesis at the Shoot Apex of Arabidopsis thaliana. Development 2004, 131: 4225-4237.

[21] Rohlfing, T., Sullivan, E. V., and Pfefferbaum, A. Deformation-based brain morphometry to track the course of alcoholism: Differences between intra-subject and inter-subject analysis. Psychiatry Res (2006).

[22] Soares, J., Kochunov, P., Monkul, E., Nicoletti, M., Brambilla, P., Sassi, R., Mallinger, A., Frank, E., Kupfer, D., Lancaster, J., and Fox, P. Structural brain changes in bipolar disorder using deformation field morphometry. Neuroreport (2005).

[23] Studholme, C., Cardenas, V., Blumenfeld, R., Schuff, N., Rosen, H., Miller, B., and Weiner, M. Deformation tensor morphometry of semantic dementia with quantitative validation. NeuroImage (2004).

[24] Tataw, O., Liu, M., Roy-Chowdhurry, A., Yadav, R., and Reddy, G. Pattern analysis of stem cell growth dynamics in the shoot apex of arabidopsis. ICIP, IEEE International Conference on Image Proccesing (2010).

[25] [Tsai, D.-M., Hou, H.-T., and Su, H. J. Boundary-based corner detection using eigenvalues of covariance matrices. Elsevier - Pattern Recognition Letters 20 (1999), 31–40.

[26] Yadav, R., Girke, T., Pasala, S., Xie, M., and Reddy, G. Gene expression map of the arabidopsis shoot apical meristem stem cell niche. PNAS (2009).

[27] Zhang, Y., and Edgar, T. F. A robust dynamic time warping algorithm for batch trajectory synchronization.

[28] Zwiener, U., Schelenz, C. h., Bramer, S., and Hoyer, D. Short-term dynamics of coherence between respiratory movements, heart rate, and arterial pressure fluctuations in severe acute brain disorders. Physiol Res (2003).

[29] Nafiz Arica, Fatos T. Yarman-Vural. An Overview of Character Recognition Focused on Off-Line Handwriting. IEEE Trans. On Systems,Man, and Cybernetics, 31(2), 2001.

[30] A. Asi, I. Rabaev, K. Kedem, J. El-Sana, User-Assisted Alignment of Arabic Historical Manuscripts, International Workshop on Historical Document Imaging and Processing (HIP2011)

[31] Barron, A., Rissanen, J., & Yu, B. (1998). The minimum description length principle in coding and modeling. IEEE Transactions on Information Theory. 44.

[32] V. Capoyleas. Geometric Clusterings. Journal of Algorithms. 12(2). 1991.

[33] Hui Ding, GoceTrajcevski, Peter Scheuermann, Xiaoyue Wang, Eamonn J. Keogh: Querying and mining of time series data: experimental comparison of representations and distance measures. PVLDB 1(2): 1542-1552 (2008)

[34] Richard C. Dorf (ed.) The Electrical Engineering Handbook, CRC Press, Boca Raton, 1993, ISBN 0-8493-0185-8 page 1770.

[35] Ahmed M. Elgammal, Mohamed A. Ismail. A Graph-Based Segmentation and Feature Extraction Framework for Arabic Text Recognition. ICDAR 2001.

[36] Alicia Fornes, Volkmar Frinken, Andreas Fischer, Jon Almazan, G. Jackson, & Horst Bunke. (2011). A Keyword Spotting Approach Using Blurred Shape Model-Based Descriptors. In *Proceedings of the 2011 Workshop on Historical Document Imaging and Processing* (pp. 83–90).

[37] A. K. Jain, M.N. Murty, P.J. Flynn. Data Clustering: A review. ACM Computing Surveys. 31(3), 1999.

[38] S. Jouili, M. Coustaty, S. Tabbone, J. M. Ogier, NAVIDOMASS: Structural-Based Approaches towards Handling Historical Documents. ICPR, pp. 946-959, 2010.

[39] S. D. Kamvar, D. Klein, and C. D. Manning. Interpreting and Extending Classical Agglomerative Clustering Algorithms Using a Model Based Approach. ICML (2002).

[40] T. Kanungo, D.M. Mount, N. S. Netanyahu, C.D. Piatko, R. Silverman and A.Y. Wu. An Efficient K-Means Clustering Algorithm:Analysis and Implementation. PAMI 24(7) 2002.

[41] Kontkanen, P., Myllymaki, P., Buntine, W., Rissanen, J., &Tirri, H. (2004-6). An MDL Framework for Data Clustering. HIIT Technical Report.

[42] Li, H., & Abe, N. (1996). Clustering Words with the MDL Principle. Conference on Computational Linguistics, 1.

[43] Li, J., Mouchere, H., & Viard-Gaudin, C. (2011). Unsupervised Handwritten Graphical Symbol Learning - Using MDL principle on Relational Graph. KDIR, (pp. 172-178).

[44] Simon X. Liao, MiroslawPawlak. On the Accuracy of Zernike Moments for Image Analysis. PAMI 20(12) 1998.

[45] Cheng-Lin Liu, Hiromich Fujisawa: Classification and Learning for Character Recognition: Comparison of Methods and Remaining Problems. Studies in Computational Intelligence, Machine Learning in Document Analysis and Recognition, 2008.

[46] J. MacQueen. Some methods for classification and analysis of multivariate observations. Proc. Fifth Berkeley Symposium on Mathematics, Statistics and Probability. (1) 1996

[47] Project PaRADIIT (2012) Pattern Redundancy Analysis for Document Image Indexation and Transcription. *https://sites.google.com/site/paradiitproject/home*

[48] I. Rabaev, O. Biller, J. El-Sana, K. Kedem, I. Dinstein, Case Study in Hebrew Character Searching, International Conference on Document Analysis and Recognition, ICDAR2011

[49] Thanawin Rakthanmanon, Qiang Zhu, and Eamonn Keogh (2011). Mining Historical Documents for Near-Duplicate Figures. ICDM 2011

[50] Rakthanmanon, T., Keogh, E., Lonardi, S., & Evans, S. (2011). Time Series Epenthesis:Clustering Time Series Streams Requires Ignoring Some Data. ICDM.

[51] Toni M. Rath, R. Manmatha: Word Image Matching Using Dynamic Time Warping. CVPR (2) 2003: 521-527

[52] P. Roy, F. Rayar and J. Y. Ramel. An efficient coarse-to-fine indexing technique for fast text retrieval in historical documents. DAS (2012).

[53] Imran Siddiqi, Nicole Vincent. A Set of Chain Code Based Features for Writer Recognition. ICDAR 2009.

[54] Ward, Joe H. (1963). "Hierarchical Grouping to Optimize an Objective Function". Journal of the American Statistical Association 58 (301): 236–244.

[55] Y. Xin, M. Pawlak, S. Liao. Accurate Computation of Zernike moments in polar co-ordinates. IEEE TIP 16(2) 2007

[56] Support Website: www.cs.ucr.edu/~tatawm

[57] A. Broder, M. Mitzenmacher. Network Applications of Bloom Filters: A survey. Internet Mathematics, 1(4):485-509 (2005).

[58] A. Kirsch, M. Mitzenmacher. Less hashing, same performance: Building a better Bloom filter. Random Structures and Algorithms, 33(2):187-218 (2008).

[59] B. H. Bloom. Space/time trade-offs in hash coding with allowable errors. Communications of ACM, 13:422-426 (1970).

[60] C. Jin, W. Qian, C. Sha, J. X. Yu, A. Zhou. Dynamically Maintaining Frequent Items Over a Data Stream. CIKM:287-294 (2003)

[61] G. Cormode, M. Hadjieleftheriou. Methods for finding frequent items in data streams. VLDB 19:3-20 (2010).

[62] H. Ding, G. Trajcevski, P. Scheuermann, X. Wang, E. J. Keogh. Querying and mining of time series data. Experimental comparison of representations and distance measures. *PVLDB* 1(2): 1542-1552 (2008).

[63] J. Lin, E. Keogh, S. Lonardi, B. Chiu. A symbolic Representation of Time Series, with Implications for Streaming Algorithms. In proceedings of the 8th ACM SIG-MOD Workshop on Research Issues in Data Mining and Knowledge Discovery (2003).

[64] M. Charikar, K. Chen, M. Farach-Colton. Finding Frequent Items in Data Streams. Theoretical Computer Science 312(1):3-15 (2004).

[65] Y. Kim, K. Cho, J. Yoon, L. Kim, U. Kim. Using Bloom Filters for Mining Top-k Frequent Itemsets in Data Streams. Secure and Trust Computing, Data Management and Applications. CCIS 186:209-216 (2011).

[66] Pranav Patel, Eamonn J. Keogh, Jessica Lin, Stefano Lonardi: Mining Motifs in Massive Time Series Databases. ICDM 2002: 370-377.

[67] S. Y. Wang, X. Hao, H. Xu. Mining Frequent Items based on Bloom Filter. FSKD 4: 679-683 (2007).