APPROVAL SHEET

Title of Thesis: Clustering and Visualization Techniques for Aggregate Trajectory Analysis

Name of Candidate: David Alan Trimm,
Computer Science Doctorate,
2012

Thesis and Abstract Approved: _____
Dr. Penny Rheingans
Professor
Department of Computer Science and
Electrical Engineering

Date Approved: _____

Curriculum Vitae

Name: David Alan Trimm

Degree and date to be conferred: Doctorate, May 2012

Secondary Education: Richmond Hill High School, Richmond Hill, GA

Collegiate institutions attended:

University of Maryland, Baltimore County, Doctorate Computer Science, 2012
Johns Hopkins University, Master of Science in Computer Science, 1998
Georgia Institute of Technology, Bachelor of Science in Computer Science, 1995

Major: Computer Science

ABSTRACT

Title of Thesis: Analyzing Trajectory Populations Through Clustering and Visualization

David A. Trimm, Computer Science, 2012

Thesis directed by: Penny Rheingans, Ph.D.
Department of Computer Science and
Electrical Engineering

Analyzing large trajectory sets enables deeper insights into multiple real-world problems. For example, animal migration data, multi-agent analysis, and virtual entertainment can all benefit from deriving conclusions from large sets of trajectory data. However, the analysis is complicated by several factors when using traditional analytic techniques. For example, directly visualizing the trajectory set results in a multitude of lines that cannot be easily understood. Statistical analysis methods and non-direct visualization techniques (e.g., parallel coordinates) produce conclusions that are non-intuitive and difficult to understand. By using two complementary processes—clustering and visualization—a new approach is developed to analyzing large trajectory sets. First, clustering techniques are developed and refined to group related trajectories together. From these similar sets, a trajectory composition visualization is created and implemented that clearly depicts the cluster characteristics including application-specific attributes. The effectiveness of the approach is demonstrated on two separate and distinct types of data sets resulting in actionable conclusions. The first application, multi-agent analysis, represents a rich, spatial data set. When analyzed using this approach, deficiencies in the underlying artificial intelligence algorithms can be determined leading to improved agent performance. Student course-grade history analysis, the second application, requires adapting the approach for a non-spatial data set. However, the results enable a clear understanding of which courses are most critical in a student's career and which student groups require assistance to succeed.

This research contributes to methods for trajectory clustering, techniques for large-scale visualization of trajectory data, and processes for analyzing student data.

Analyzing Trajectory Populations Through Clustering and Visualization

by

David Alan Trimm

Thesis submitted to the Faculty of the Graduate School
of the University of Maryland in partial fulfillment
of the requirements for the degree of
Doctorate
2012

ProQuest LLC.
789 East Eisenhower Parkway
P.O. Box 1346
Ann Arbor, MI 48106 - 1346

For my mother and father.

ACKNOWLEDGMENTS

I would especially like to thank Penny Rheingans, my advisor, who provided endless support through our weekly meetings. Without Penny's insight into visualization techniques (especially color models and conference submissions), my work would have been incomplete and lacking in fundamental rendering and analytical techniques. Additionally, Penny provided unlimited guidance that focused my energy towards useful (and productive) strategies for my research.

Furthermore, a special thanks to my committee, Dr. Marie desJardins, Dr. Marc Olano, Dr. Anupam Joshi, and Dr. Sreedevi Sampath. Marie's collaborative spirit during conference submissions significantly enhanced my work and made my results understandable to those outside the visualization field. In addition to exposing me to the power of shading languages and parallel computing architectures, Marc showed how efficient stochastic methods can be for making intractable problems solvable. Dr. Joshi provided significant insight into network security and wireless protocols that, while unrelated to my dissertation research, has enriched my professional career. In addition to my committee members, Dr. LeBerge's insight into student performance patterns was absolutely essential to understanding my early results and refining the process into a successful analytic methodology.

I would also like to thank Calvin and Fen Shih, my wife's parents, for caring for my children so that I could spend more time on my research. Finally, I am forever indebted to my wife Kim and my three children, Alex, Ryan, and Jacob, for their encouragement, support, and patience throughout my research. Their love and understanding have enabled me to persist through the longest academic experience of my life.

TABLE OF CONTENTS

LIST OF FIGURES

Chapter 1

INTRODUCTION

Due to the large volume of data involved, trajectory population analysis requires new techniques to organize and present the data to enable sense-making and deeper insight. A successful analytical approach adds significant value across a wide range of applications by automatically partitioning the population and accurately presenting trends and characteristics of the resulting groupings. Such conclusions are not available when trajectories are analyzed individually or by using existing visualization techniques. This research proposes an approach that automatically clusters trajectory populations into similar sets, which are then visually analyzed in aggregate to understand each cluster's structure and trends. The techniques are applied to both spatial and non-spatial datasets and successfully enable users to determine unique and innovative results that are not apparent with existing methods.

For the purposes of this research, trajectories are discrete entity-based, temporal-varying data sets that can be reduced to *two-dimensional space* with a number of application-specific attributes. For example, GPS vehicle plots, player trajectories in virtual environments or gaming events, and animal migratory patterns all fit within this context. While spatial trajectories have a natural representation within the physical world that can be plotted or graphed, non-spatial trajectories do not. For example, stock market trends, sensor readings, and twitter topics represent non-spatial trajectory types. Attributes are

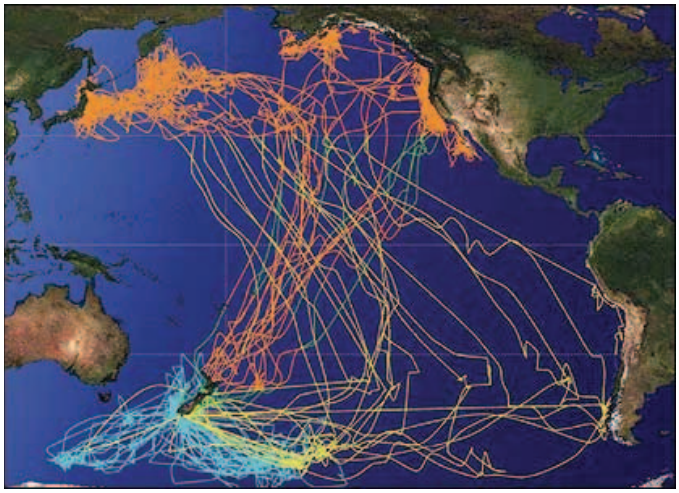

FIG. 1.1. Sooty shearwater migration map. Source:
http://pausetobegin.com/blog/tag/sooty-shearwater-photographs-photographer-migration/

application-specific numerical values that can be directly associated with the trajectory's entity, such as demographic data, and therefore remain constant over the trajectory. Alternatively, attributes can vary over the trajectory's course (e.g., a soccer player's speed while dribbling the ball). In general, attributes provide insight into (1) the entity following the trajectory, (2) changes to the entity over time, and/or (3) the environment through which the entity travels. When trajectories are collected in aggregate over the same domain, the overall data set is referred to as a *trajectory population*. Analytical conclusions about the aggregate trajectories vary based on application. For example, results from automobile GPS data may reveal vehicle-specific characteristics, local road phenomena, or road congestion during specific hours of the day. Analytic findings could then be used to improve vehicle designs, repair road issues, or plan new routes to ease congestion. While this example delves into a specific type of trajectory data, each application area would have similar benefits from aggregate analytical techniques.

Animal migratory patterns provide an excellent case study describing the necessity for trajectory population analysis. Figure 1.1 shows an example of such a domain. With the

proliferation of widespread global positioning technologies and wireless communications, it is now possible to accurately track the location of tagged animals over time. If multiple groups of animals are tagged and tracked over many years, trends in these patterns may show the effects of global warming or human development on migration. For example, global warming may affect the animal's primary food source, leading to the population seeking out a wider feeding area. Such is the case with the polar bear hunting grounds, in which the ice flows have melted and become unstable, forcing the species to modify their hunting activities. Trajectory analysis may also show the effects of human activities on migratory trajectories. New roads or other developments may force modifications to the animal's movement or the development of alternative survival strategies. In both cases, additional information (e.g., the animal's weight, the number of offspring), may reveal aggregate trends when analyzed together.

1.1 Thesis Statement

To effectively analyze large trajectory populations, new techniques and approaches need to be developed for real-world data sets. Leveraging machine analytics to organize large data sets and representing the results in intuitive formats provide a way forward in this problem domain. I claim that clustering and aggregate visualization techniques enable deeper insight into analyzing two-dimensional, entity-based trajectory populations at scale than traditional approaches.

My assertion is plausible for three reasons. First, successful analytic approaches for large data sets require partitioning data into similar groupings. Divide-and-conquer strategies are effective across many domains to efficiently and effectively handle complex problems. As such, clustering is a natural implementation of the divide-and-conquer strategy to partition and group data elements into like sets. By clustering trajectory populations,

the similar elements are grouped together which can then be analyzed to discover trends in each individual grouping. Alternatively, if one attempted to analyze dissimilar trajectories, comparing the corresponding and appropriate trajectory subcomponents would be difficult to impossible. In addition to forming the basis for an analytic strategy, clustering can also facilitate the visual composition process.

Second, visualization techniques intuitively depict two-dimensional spatial data sets. By leveraging human perception, trajectory data can be successfully rendered enabling a comprehensive understanding. Furthermore, by leveraging color to convey statistical properties of the trajectories, aggregate trends in the data become apparent. Furthermore, color is commonly used across a variety of visualization applications to portray data values. Based on existing visual composition techniques combined with trajectory representations, a successful composition can be created that validates the clustering stage and provides insight across multiple facets of the trajectory population. While statistical analysis techniques could be used to both validate clustering and depict trends, results are often unintuitive and difficult to verify without visual aids.

Third, once clustered and composited together, the resulting aggregate trajectory visualizations will enable new methodologies for analyzing this type of data. By providing a new method to combine the data into an intuitive rendering, new methods to look at the data will be developed that lead to novel conclusions. Analytic trade craft evolution is often a result of new methods to visualize and render the data. Once initial results are provided to domain experts, refinements in both the visualization approach as well as the analytical process can occur. Evolution of analytic techniques often result from new approaches, and, by providing intuitive depictions that leverage visual techniques, new inspirations for understanding the data can be created.

I set out to prove this thesis by developing techniques to cluster two-dimensional trajectories and visualize the results in aggregate. To determine the effectiveness of this

approach, I applied the techniques to two separate application domains, found novel discoveries, and reviewed my results with domain experts. Through this process, I develop additional analytic methodologies that leverage the approach to successfully discover and understand trends in student course-grade history data.

1.2 Approach and Results Summary

Before trajectories can be effectively analyzed, they need to be grouped together into common sets that share characteristics such as spatial similarities. Clustering is a natural grouping mechanism for organizing similar trajectories into distinct partitions over the entire population. To successfully cluster trajectories, distance metrics need to be established that accurately determine how similar trajectories are to one another. I propose and apply an effective distance metric for two-dimensional spatial representations and also provide a method for non-spatial data sets (i.e., those that lack a natural two-dimensional representation). The two-dimensional distance metric also has a natural companion method for providing an "average" trajectory for a clustered trajectory set. The average trajectory algorithm is specifically tailored to enable later stages in the analytical pipeline to visualize the trajectories in aggregate.

Once trajectories have been successfully grouped into related clusters, visualization techniques provide a complementary process to depict the cluster's structure and attribute values. This research develops and implements a technique that accumulates thousands of spatially related trajectories to depict trends in the cluster's trajectory attributes. In addition to yielding insight into the attribute trends, the visualization characterizes the overall shape and structure of the cluster and how it varies and fluctuates over time. Consequently, the visualization verifies and validates that the clustering stage successfully grouped similar trajectories from the population.

To demonstrate the effectiveness of this approach, both the clustering and visualization processes are applied to two distinct applications. The first application, simulated robotic soccer competitions, represents a spatial application where two-dimensional trajectories are a common feature of the simulation. Since the simulation takes into account many different variables about the soccer match, the data set provides a rich set of attributes that can be used to derive analytical conclusions about successful artificial intelligence strategies. Additionally, since the goal of the simulated soccer matches is to foster development in multi-agent behavior research, the techniques have applicability to a wider range of artificial intelligence and multi-agent applications (e.g., flocking behavior).

The second application, college-level student course and grade history, represents a non-spatial application. In order to apply the techniques outside of a traditional spatial application, the approach was tailored to create a spatial representation that captures the relevant student and course characteristics. While spatial applications can be clustered using the proposed distance metric, modifications to the metric were necessary to group related students. Once successfully partitioned in their natural groups, each individual student's history was then mapped into a two-dimensional representation that captured the relevant aspects. These aspects enabled a higher level of understanding for each of the resulting clusters to include areas where students deviated from one another in their course work and where course grades significantly fluctuated. In addition to clustering, students were also partitioned by their course-grade bins for gateway, core, and other groupings for comparison using a "small multiples" approach. These natural groupings revealed deeper insight than the proposed clustering approach.

1.3 Contributions

This research provides several substantial contributions including (1) methods to successfully cluster trajectory populations for further analysis, (2) visualization techniques for large sets of related trajectories, and (3) analytical methodologies for student course-grade history data. By developing complementary processes that cover multiple stages of the analytical work flow, unique contributions are delivered that are effective for real-world problems involving large trajectory populations.

Although there are multiple approaches to trajectory clustering, few are used in tandem with aggregate visualization techniques. Furthermore, the implementations for the trajectory clustering algorithms also vary significantly—application dependencies necessitate a careful choice to fully exploit the underlying data. For example, I chose to use parametric representations as the underlying model in both use cases. However, a length-based representation is used for the first application, whereas a time-based version worked better for the second application. In cases where the trajectories are naturally constrained (e.g., GPS vehicle data), correlating the closest spatial location delivers the correct solution. By reviewing the results of the clustering stage for each use case, a decision process for the underlying representation is derived to optimize the follow-on visualization stage.

In addition to the clustering phase contributions, I develop and apply a sophisticated algorithm to visualize aggregate trajectory data. While a few application-specific techniques exist, this approach provides a general solution that works across several real-world problem domains. While it does have limitations and trade offs, it can be used at scale to provide an aggregate view of a clustered trajectory population. Through the complementary clustering process, the cohesive nature of the similar trajectories is exploited—this enables a solution within the visualization phase that would not be possible in isolation. Typically, clustering is applied as a post-process to the visualization stage. For instance,

edge bundling algorithms (Holten & van Wijk 2009) use clustering on the geometric components to achieve a more effective appearance. However, by first organizing the data so that similar elements share common characteristics, successive analytical stages are fed in a complementary fashion to deliver an effective solution.

Lastly, through my analysis of the student course-grade history data, analytical methodologies are developed to identify critical courses for student success. By applying the techniques to a real-world data set and reviewing them with faculty members, an effective approach is developed that groups students across various course-grade combinations. When these natural groupings are rendered using the aggregate visualization algorithm, clear dependencies for student success become apparent—these results can then recommend specific changes to the curriculum to improve the program's success.

1.4 Overview

The rest of the paper is organized as follows. In Chapter 2, previous works devoted to trajectory clustering, visualization, and multi-agent analysis are explored with an emphasis on techniques focused on two-dimensional trajectories. Chapter 3 describes my approach to clustering and visualizing trajectory populations. In order to use existing clustering approaches, a distance metric for two-dimensional trajectories is developed that also complements the visualization process. Chapter 3 also explains the visualization composition process. Since the visualization must verify and validate the clustering process in addition to depicting trends in trajectory attributes, Chapter 3 defines the goals and objectives for the visualization techniques. The visual aggregation method is then described in detail with two separate rendering methods discussed and contrasted. In order to show the applicability of the approach, two case studies are presented to show the effectiveness. The first case study, shown in Chapter 4, examines RoboCup Soccer simulations using both traditional

visualization techniques and the approach developed by this research. The second case study, documented in Chapter 5, applies the approach to student course history data which demonstrates the applicability in non-spatial domains. Both applications fully explore the benefits of the approach as well as shortcomings. These shortcomings are addressed in Chapter 7, which explores future research directions for extending the proposed approach.

Chapter 2

PREVIOUS WORK

Previous work related to trajectory analysis falls into three distinct but interrelated categories: trajectory clustering, entity and agent analysis, and trajectory visualization. Trajectory clustering focuses on grouping related trajectories or assembling multiple trajectory subcomponents together. Entity and agent analysis examines methods to understand and derive conclusions from agents generating trajectories. Finally, trajectory visualization deals with ways to visualize individual or grouped trajectories. The following chapters will review, compare, and contrast each of these areas.

2.1 Trajectory Clustering

Palma et al. (2008) identify interesting locations from mobile device data by clustering subcomponents within the trajectory data for stops and starts. Their method uses density-based clustering to identify and grow these subcomponent trajectories. While the work is applicable to the agent analysis described in Chapter 5, the overall goals and objectives differ from this research. For example, their efforts are geared towards extracting feature vectors that are relevant to interesting spatial locations. Much of their work deals with extracting meaning from trajectories and developing the necessary heuristics and rules to automatically find previously unknown locations of interest in cities. However, Palma's

techniques use a relatively simplistic visualization that could benefit from aggregate visualizations and therefore be complementary to my approach.

Lee et al. (2007) develop a bottom-up approach to find common sub-trajectories in trajectory sets. While the density-based approach yields strongly connected trajectories, the technique is difficult to apply to spatially unconstrained trajectory sets. For example, the heuristic for merging related trajectories is sensitive to both the spatial location and the direction of the subcomponents. With unconstrained data sets, these sensitivities are difficult to separate from one another, causing unassociated subcomponents to be grouped together. I implement Lee's algorithm in Chapter 3, but as with many of the other trajectory clustering approaches, Lee's work could leverage aggregate visualizations to better understand the effectiveness of the clustering algorithm and the associated attributes.

Lee et al. (2008) develop two types of clustering for identifying region-based and trajectory-based elements. Region-based clustering identifies areas where all of the enclosed sub-trajectories move in the same direction while trajectory-based methods construct longer chains of sub-trajectories using density-based clustering. Their approach applies more to machine learning but has wider applicability to both constrained and unconstrained trajectory data sets. For example, the technique effectively separates non-constrained locales from constrained corridors of movement. While the clustering approach is significantly more advanced than my use of clustering, Lee's approach could be aided by aggregate visualizations, especially within the trajectory-based methods. However, the region-based clustering results would need a separate approach for aggregate visualization because the data is not cohesive.

Jeung et al. (2008) discover convoys (entities that move together) in temporal-spatial trajectory datasets. Their methods are complementary to my approach and could significantly benefit from the proposed aggregate visualizations, especially as they relate to showing both the variability and the associated attribute values. Verhein (2009) develops a sep-

arate approach with similar goals. His efforts refine trajectory sets using Spatio-Temporal Association Rules (STARs) to identify group behavior while overcoming noise in the data sets and resolving overlap.

2.2 Entity and Agent Analysis

There are several research areas and commercial activities directly related to path analysis. GeoTime, a commercial visualization application developed by Oculus (Kapler & Wright 2004), is arguably the state of the art for analyzing paths and associated events. In addition to visualizing both the spatial and the temporal aspects of paths within a single comprehensive view, the application provides interactivity to enable an analyst to explore the data. To save analysts time, the application can also automatically correlate paths against the entire data set. However, on the visualization side, GeoTime appears to be unable to display thousands of paths at the same time, and the automated assistance is mostly constrained to repetitive tasks.

Several methods have been developed to analyze movement in virtual worlds. Bowman et al. (1998) developed a way to evaluate how users move within virtual worlds. These researchers sorted movement samples taken from virtual environments into different categories and then evaluated the success of each sample for specific tasks. Statistics were then gathered based on the assigned categories and each individual sample. However, the trajectories themselves were not directly analyzed. Zanbaka et al. (2005) analyzed the same type of movement within virtual worlds as Bowman et al. However, these researchers visualized the aggregation of the trajectories in a manner similar to the straightforward approach described in the introduction. As expected, the view was quite cluttered, leading the researchers to call this a "Spaghetti Plot Visualization" (similar to Figure 2.1). The researchers also created an aggregate view of how long each virtual user remained in a

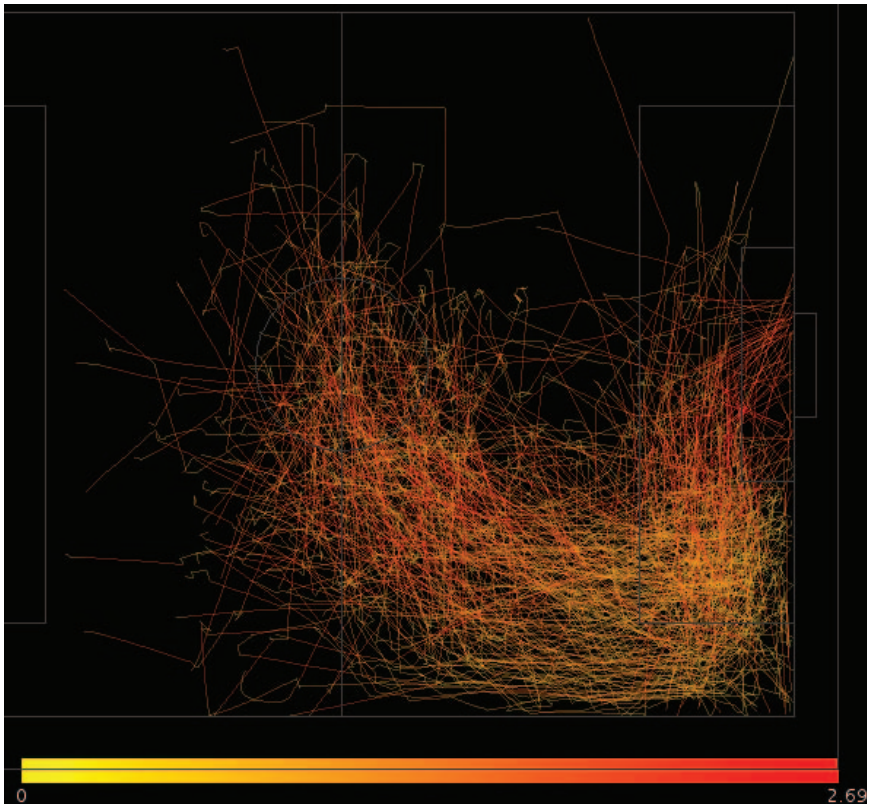

FIG. 2.1. Spaghetti plot. Multiple trajectories can be overlaid on top of a contextual background, but the image is difficult to analyze and understand, especially when colored with trajectory attributes.

location, referred to as dwell time, which better handled the volume of data. Chittaro et al. (2006) developed a tool to visually analyze entity movement within 3D virtual environments. The goals were to design better environments and to study user behavior. Aggregate trajectory data were successfully visualized using vector fields to represent flow and density graphs to show congestion.

Thawonmas et al. (2008) examined trajectories from an online game to find landmarks. They proposed detecting landmarks by clustering the distribution of visiting players and visualized these clusters with multidimensional scaling. While their methods provided unique insight into player behavior as it relates to the online game, they did not visualize the actual trajectory data in aggregate.

Several recent publications have focused on real-world trajectory data. Andrienko and Andrienko (2008) developed a useful ontology for trajectory analysis. They examine large amounts of GPS-derived data to determine traffic congestion and trends in road use. They provide several new visualizations to understand the data across days of the week and the spatial dependencies (e.g., congestion within a particular map grid). Furthermore, they aggregate data through the use of clustering techniques. Willems et al. (2009) investigated marine vessel movement data and created kernel density visualizations. By manipulating the kernel parameters, overview and detail views were generated, providing insight into shipping lane activity. In addition to the vessel density, velocity was also rendered, depicting potentially dangerous situations involving slow-moving ships.

2.3 Trajectory Visualization

In addition to the visualization techniques already described, flow visualization is the most similar technique to this research. Primarily because of fluid dynamic simulation, the most common resultant data are particle trajectories that need to be aggregated for analy-

sis. Laramee et al. (2004) provide an excellent summary of the various flow visualization techniques. In particular, the closest approach is the *dense, texture-based flow visualization* type that Laramee et al. describe. Several important differences exist between flow visualization and this approach. Flow visualization primarily works with datasets where local phenomena exhibit similar tendencies. For example, in the case of air tunnels and currents, the particles of interest are perturbed in similar ways. My approach is to separate out via clustering each unique flow pattern for further composition. Furthermore, because clustering is applied prior to visualization, the resulting composition consists of flows that are fairly uniform throughout the visualization.

Research into more effective use of parallel coordinate visualizations is also similar to this approach. For example, visualization techniques that provide aggregate depictions of trends in multi-variate data. Yang et al. (2003) develop multi-resolution views using hierarchical clustering to group similar parallel coordinate data sets. Similarly, Johansson et al. (2005) cluster large parallel coordinate datasets in order to show the underlying structure of the data as well as statistical properties. My approach differs in several important aspects. First, the trajectory data sets do not fit well into the base structure for parallel coordinations. For example, in both the spatial and non-spatial cases, the temporal aspects play a crucial role but do not equate to different dimensions in the data. In the first use case, parallel coordinate aggregation (and other traditional visualization techniques) are applied to the extracted properties of the spatial data. Second, varying trajectory lengths occur often in the data—both in time/duration and in actual spatial length. While parallel coordinates could be modified to handle different spatial lengths, the fit would be imperfect. Furthermore, parallel coordinate visualizations are specifically tailored to multiple one-dimensional attributes. When applied to naturally spatial datasets (i.e., two- or three-dimensional data), the intuitive nature of the data is lost. In this research, I was able to successfully depict the second use case, student-course data, in a parallel coordinates-like

view. In fact, the visual composition approach can be viewed as a more generic parallel coordinates view for two-dimensional datasets.

Chapter 3

APPROACH

The approach to trajectory population analysis is to first cluster the population into similar partitions or sets. Once trajectories with similar spatial properties are grouped together, the composition visualization process can then be used to capture the overall spatial representation of the clustered trajectories. In addition to characterizing the average trajectory for the group, the visualization also shows deviations and variability across each trajectory set. Furthermore, through the use of both color blending and weaving, two approaches to coloring the visualization, the composition depicts an additional attribute related to the cluster and commonalities across the clustered partition. The remainder of the Approach chapter is broken into three distinct sections. Figure 3.1 provides a graphical depiction of

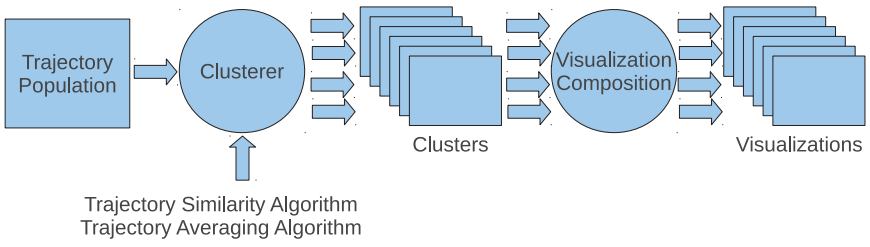

FIG. 3.1. Approach overview

the overall steps associated with the implementation. In the figure, the trajectory population at the left is fed into the clustering algorithm, which groups related cluster trajectories together. Each cluster is then fed into the visualization composition process to be rendered by the visual composition process. A "small multiples" approach (Tufte 1990) is then used to compare and contrast the resultant visualizations as represented by the multiple images (i.e., stacks in the diagram). Section 3.1 discusses trajectory representation and defines the similarity metric and averaging algorithm that were used by the clustering operation. Section 3.2 defines the approach to clustering the trajectory population and various design decisions and algorithms chosen to be most effective for feeding the visualization composition process. Finally, Section 3.3 delves into the steps needed to composite the trajectory clusters into an aggregated composition, which depicts the average trajectory, the variability in individual trajectories, and the attributes associated with each entity.

3.1 Trajectory Representation, Comparison, and Averaging

A trajectory is defined as a discrete entity-based, temporally varying data sequence that can be reduced to *two-dimensional space* with a number of application-specific attributes. In order to use the trajectory, a trajectory is defined in the following parametric representation:

Definition 1. Trajectory: A trajectory is a list of space-time-attribute points $\{ p_0 = (x_0, y_0, t_0, a_{0,0}, a_{0,1}, ...)$, $p_1 = (x_1, y_1, t_1, a_{1,0}, a_{1,1}, ...)$, ... , $p_n = (x_n, y_n, t_n, a_{n,0}, a_{n,1}, ...)$ $\}$ where $x_i, y_i, a_{i,j} \in \Re$ and $t_i \in [0...1]$ and $t_0 = 0.0$ and $t_n = 1.0$ and $t_0 < t_1 < ... < t_n$. In addition to the data points, a trajectory has an associated entity (e.g., a person, player, actor). Attributes can be fixed for a trajectory, fixed for the entity, or change over the course of the trajectory.

In practice, distance along the trajectory is used for the parametric representation.

However, duration along the trajectory may also be used and will produce equally valid results that may be more appropriate when entities frequently start or pause at spatial locations (e.g., GPS data where vehicles are delayed by congestion or traffic lights). Duration is also used for non-spatial applications in which all of the entity activity can be binned into discrete time intervals—the second use case, student course history data, falls into this category. Lastly, for spatial applications with constrained corridors (e.g., vehicle GPS data), trajectory points are best compared and correlated with their closest neighboring trajectory points.

In order to compare and average trajectories, intermediate values that do not exactly align with the discrete trajectory parameterization need to be calculated. In order to interpolate these values, the following equation is used:

Definition 2. To calculate an arbitrary point, t, along the trajectory, i is chosen such that $t_i < t < t_{i+1}$. The proportional time value is calculated, $d = (t - t_i)/(t_{i+1} - t_i)$. The remaining values for the arbitrary point can then be calculated as follows, $x = (d * (x_{i+1} - x_i)) + x_i$ and $y = (d * (y_{i+1} - y_i)) + y_i$.

Clustering algorithms require a similarity metric for determining how individual elements (i.e., trajectories) compare with one another. In addition to the actual comparison or distance metric, several clustering algorithms also require an averaging function for already grouped elements to be iteratively compared with non-grouped elements. Both the similarity metric and average-trajectory calculation use the same fundamental algorithm.

The trajectory comparison algorithm needs to satisfy several high-level goals. First, it must correctly model high-frequency sections of each trajectory. High-frequency sections are defined as having significant changes in trajectory curvature or direction over a short section of the trajectory. The comparison algorithm must also efficiently process low-frequency trajectories (i.e., trajectories that have few curves or little variability in segment orientation). Lastly, the algorithm should provide the same similarity score for two trajec-

tories that have the same spatial characteristics but vary in the number of points or segments that define the trajectory. These two straightforward goals ensure that similar trajectories will be efficiently compared and will take into consideration all of the points along the trajectory.

The algorithm satisfies the first two principles (high- and low-frequency trajectories) and provides similar but not exact scores for the last design criterion (same spatial shape). The last design criterion can be accomplished if redundant points (i.e., those that are co-linear to the previous and subsequent trajectory points) are removed prior to the comparison algorithm.

Since the discrete parametric values determine the frequency of the trajectory, the parametric values form the basis for the comparison algorithm. First, the algorithm gathers all of the parametric values from the two trajectories. Duplicate parametric values are then removed, leaving only unique values. Next, for the remaining parametric values, the corresponding spatial locations from both trajectories are determined (interpolated as necessary), and their spatial, Euclidean distances are fed into a root mean square (RMS) calculation. Since a trajectory can only change direction at a parametric value, the proposed distance comparison correctly captures high-frequency sections. Furthermore, low-frequency trajectories are efficiently processed since they contribute few parametric values to the algorithm. The algorithm can be summarized as follows:

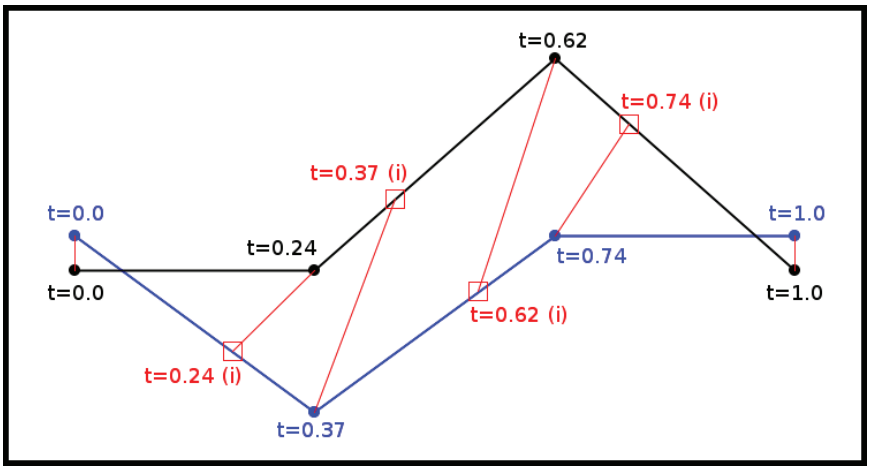

FIG. 3.2. Trajectory distance comparison. The red squares indicate locations where interpolation is required to align all of the parametric values across both trajectories with one another.

Algorithm 3.1.1: TRAJECTORYSIMILARITY($Trajectory1, Trajectory2$)

$Tset \leftarrow$empty

for each $t \in Trajectory1$

  do $Tset \leftarrow t$

for each $t \in Trajectory2$

  do $Tset \leftarrow t$

$sum \leftarrow 0$

for each $t \in Tset$

$$\textbf{do} \begin{cases} Distance = \text{EUCLIDEANDISTANCE}(\text{POINT}(Trajectory1, t), \text{POINT}(Trajectory2, t)) \\ sum \leftarrow sum + Distance * Distance \end{cases}$$

$return$SQRT$(sum/|Tset|)$

The *Point* function returns the interpolated spatial location on the specified *trajectory* for the specified parametric value *t*.

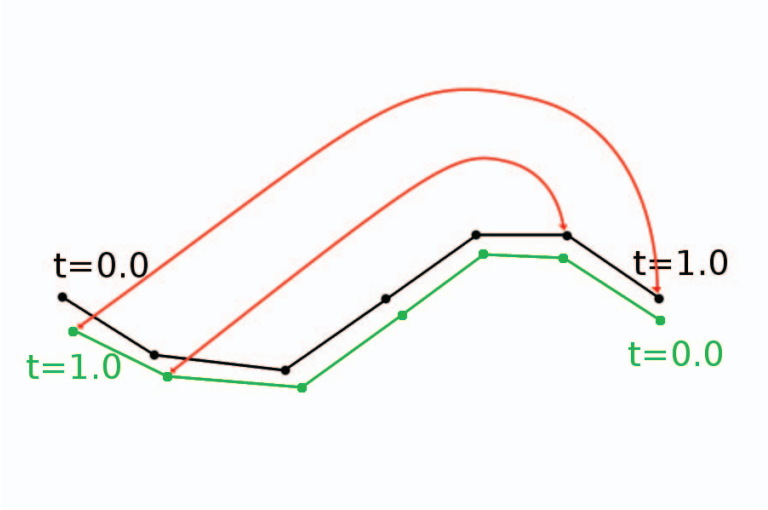

FIG. 3.3. Closest point comparison. This method fails if the trajectories have the same point value but in reverse chronological ordering. The two red line show the initial set of parametric values that are compared (incorrectly).

Figure 3.2 shows a graphical depiction for comparing two trajectories. In the figure, the parametric values for each trajectory are listed as the "t" value. The red lines show which points would be compared, and the red text with the "(i)" labels show where trajectory points would need to be interpolated to provide an accurate comparison. In the figure, every red line is added to the root mean square calculation.

An alternative to using the parametric representation for comparison is to use the closest point without regard to its trajectory location (Andrienko & Andrienko 2008). In general, this technique works well if the trajectory population is constrained to fixed corridors. For example, vehicle GPS data can be compared via closest-point comparison because the data has already been constrained (Rogers, Langley, & Wilson 1999). However, for the generic case where the trajectory data is not constrained, the closest point algorithm fails to correctly model the trajectories for comparison. Instead, trajectories that may have coinciding points but vary in their temporal ordering may be incorrectly classified as similar (for example, the reverse trajectory would have a high similarity score but would not be

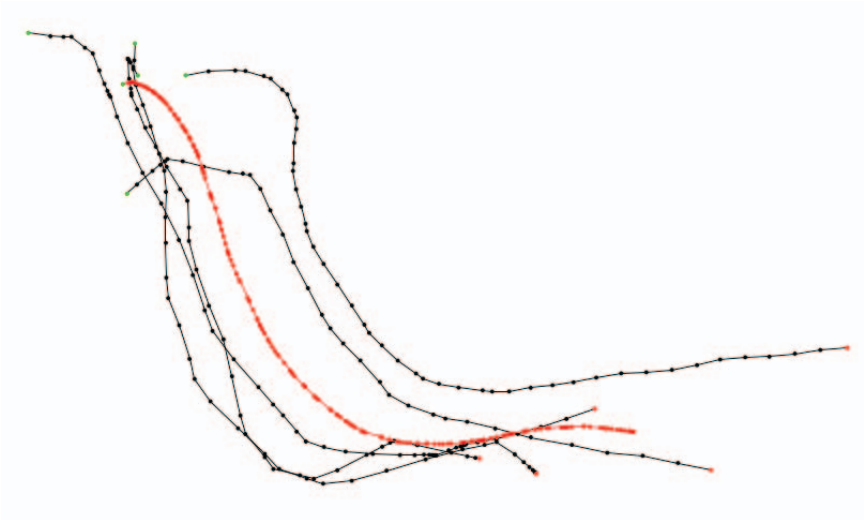

FIG. 3.4. Trajectory average example. The average trajectory, in red, is created by assembling all of the parametric points from the contributing trajectories, in black.

viable for composition—Figure 3.3 shows this example).

The requirements for the trajectory averaging algorithm are the same as those for the distance comparison—namely, that high-frequency sections are correctly captured, low-frequency trajectories are efficiently processed, and spatially identical trajectories produce the same results. Since the trajectory averaging algorithm is modeled after the distance algorithm and based on the parametric values, the first two requirements are satisfied. Unlike with the distance algorithm, the third requirement is also satisfied. Since many trajectories may need to be averaged at once, the averaging algorithm first gathers the discrete parametric values from all of the trajectories to be averaged. As with the distance comparison, the parametric values are de-duped and used to determine the appropriate points along the trajectories to average. The parametric values are also sorted to correctly order the values for the average trajectory. To calculate one of the average points, the corresponding parametric value is used to determine the spatial location on all of the contributing trajectories. These locations are then averaged together to determine the point on the average trajectory. The

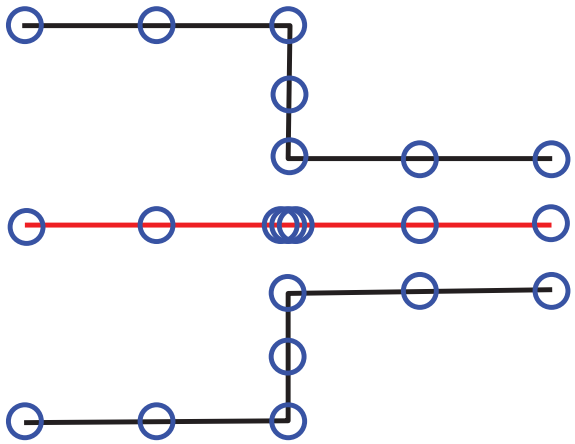

FIG. 3.5. Parametric values for average trajectory. If the parametric values for the average trajectory are re-calculated based on the average trajectory's length, errors will occur in future comparisons and averaging operations.

final average trajectory is then formed by linking these averaged points together into a set of discrete line segments. An example of the trajectory averaging results is shown in Figure 3.4. In the figure, the red trajectory depicts the average of the clustered trajectories (shown in black). The discrete values are also shown as small circles along each trajectory—as one can see, the average trajectory has many more discrete locations since all parametric values from the clustered trajectories are used.

In order for future comparison algorithms to work correctly, the parametric values assigned to the points on the average trajectory are not based on the distance (or duration/fixed time intervals) along the average trajectory. Instead, the parametric values retain the values calculated during the averaging algorithm. This ensures that both future comparisons and averaging operations faithfully model the original data points. Figure 3.5 depicts an extreme example where a re-calculation of the parametric values would cause errors in future comparison operations. In the figure, the middle section of the average trajectory (red) shows the problematic parametric values. Since the average trajectories segment is orthogonal to both of the contributing trajectories, the average trajectory becomes distorted

(i.e., shortened even though both contributing trajectories have a lengthy section at that location).

The proposed averaging algorithm produces trajectories whose number of points are the sum of the number of points of all of the contributing trajectories. Since the number of distinct parametric values from all of the trajectories is used and identical occurrences of parametric values are rare, the number of points in the average trajectory grows multiplicatively with the number of contributing trajectories, creating a performance bottleneck for the clustering operations. Line-simplification algorithms (Hershberger & Snoeyink 1992) can be applied to significantly reduce the computational requirements for successive clustering iterations but are not needed for the data sets in the case studies. As with the averaging algorithm, the line-simplification approach would need to maintain the original parametric values and not be re-parameterized on the simplified trajectory. Once simplified, the trajectory can then be used for future distance comparisons. It is recommended that the original average trajectory be used for successive averaging operations to decrease the accumulated error.

3.2 Clustering

Before the trajectories can be visualized, the trajectory population needs to be organized into related sets or groupings. By only compositing similar trajectories in a single visualization, the cohesive nature of the spatial representations can be exploited to composite the trajectories together. This cohesive nature enables the visualization to maintain the average shape of all of the trajectories and to bring out the variations in the trajectory attributes for analysis. If, on the other hand, randomly chosen trajectories were composited together, the average shape would have no meaningful representation and fail to provide a context for comparing the trajectory-related attributes.

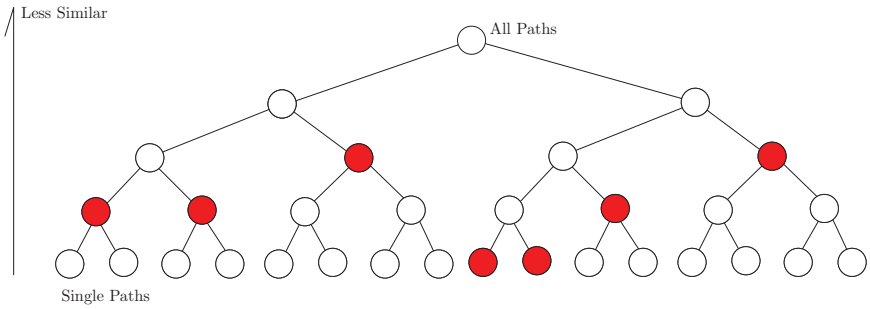

FIG. 3.6. Hierarchical clustering. To determine a distinct, non-overlapping set of clusters, a cut of the tree must be taken. The cut is depicted as the red elements in the tree.

In order to organize the trajectory population into related sets, cluster analysis is used on the trajectories to determine which are related. There are many clustering approaches that can be used. For the use case datasets, hierarchical clustering is used. Hierarchical clustering is a bottom-up approach that iteratively merges the two most similar elements within the data set, forming a new element until only a single grouping remains. For the first use case, similarity is determined by the distance metric, as defined in the last section. The new element is represented as the average of the merged elements and is used for further comparisons. In order to maintain the relationship between the merged elements and the new element, the hierarchical clustering algorithm represents the merger as a parent-child relationship in a binary tree. The algorithm continues to merge similar elements until only one remains that forms the root of the binary tree. Once finished, all of the original data elements exist as leaves of the tree while the interior nodes consist of averaged representations. The root element represents the average of all of the trajectories within the dataset.

In order to find a distinct set of non-overlapping clusters, a cut of the tree is taken, as shown in Figure 3.6 (Ferral 2004). The cut is accomplished by a recursive pre-order binary tree traversal that examines the root of the subtree before traversing the left or right subtree. At each subtree's root, the cut algorithm compares the maximum distance for any

FIG. 3.7. Cluster centers. Once clustered, the following trajectories represent the cluster results for the RoboCup data set.

of the subtree's leaf elements (i.e., the original trajectories) against the average trajectory represented by the current subtree root. If the maximum distance is less than an empirically derived threshold, the entire subtree is considered a complete cluster and the traversal is halted for this subtree's children. If the maximum distance is greater, then each child is recursively examined until the maximum distance falls below the threshold. The threshold was empirically determined to maximize the spatial cohesion (i.e., spatial similarity of the trajectories) while grouping as many trajectories together as possible. Figure 3.7 shows the resultant average trajectories for the first application, RoboCup, after clustering. The figure shows the average ball trajectory for the clustered plays within the tournament, and the width of each line corresponds to the number of trajectories within the cluster. In summary, the algorithm can be represented with the following pseudocode:

Algorithm 3.2.1: RECURSIVETREECUT($Root, Threshold$)

if $RootisLeaf$

then ADDCLUSTER($RootTrajectorySet$)

$AverageTrajectory \leftarrow$ TRAJECTORYAVERAGINGALGORITHM($RootTrajectorySet$)

for each $trajectory \in RootTrajectorySet$

do $MaxDistance \leftarrow$ TRAJECTORYSIMILIARITY($AverageTrajectory, trajectory$)

if $MaxDistance < Threshold$

then ADDCLUSTER($RootTrajectorySet$)

$$\textbf{else} \begin{cases} \text{RECURSIVETREECUT}(\text{CHILD}(Root, Right), Threshold) \\ \text{RECURSIVETREECUT}(\text{CHILD}(Root, Left), Threshold) \end{cases}$$

Hierarchical clustering was chosen for several reasons. First, common clustering tech-

niques (e.g., *K-means* (Berkhin 2002)) require a pre-established notion for the number of clusters. Second, by dynamically re-cutting the resultant tree, the optimum number of clusters can be derived through empirical analysis, which yielded spatially cohesive clusters for the visualization process. Third, hierarchical clustering was tractable with the number of trajectories in both use cases. *Density-based* clustering algorithms were explored, since previous works had successfully processed spatial trajectory sets (Lee, Han, & Whang 2007). However, these results separated the trajectories into piecewise data elements for re-assembly during the clustering process. For my application, I wanted to maintain the connectivity between the trajectory and the entity creating the trajectory for follow-on analysis. The density-based approach does yield motivating results for spatially constrained applications such as GPS or other fixed-corridor applications. For example, Figure 3.8 shows the results from applying density-based clustering against GPS data. Since the data is directionally cohesive, especially in constrained local areas, the density-based approach yields excellent results since there is little ambiguity in which sub-trajectory components fit together. However, when applied to the RoboCup data set, the algorithm produced mixed results due to the unconstrained nature of the player activities. Figure 3.9 shows the initial results when the density-based method was applied. The algorithm was unable to construct cohesive trajectories due to the many different directions taken within the small locales.

3.3 Visualization

The trajectory composition structure needs to show the overall spatial characteristics of the cluster and the trend of the trajectory-related attribute. The spatial characteristics provide the viewer with an overall feel for the cluster's compactness, the density and general shape of the clustered trajectories, and where spatial variance occurs along the trajectories. The trajectory-related attribute requirement captures how an application-specific

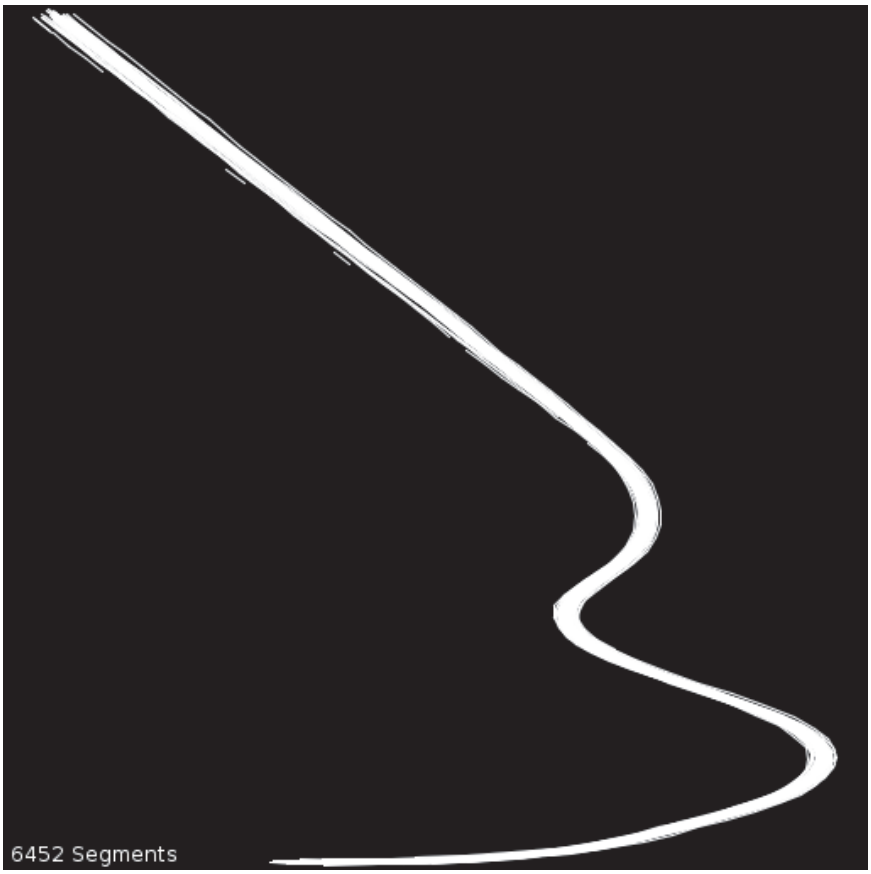

FIG. 3.8. Density-based clustering (constrained data). For GPS and other spatially constrained applications, density-based clustering works extremely well to create continuous trajectories.

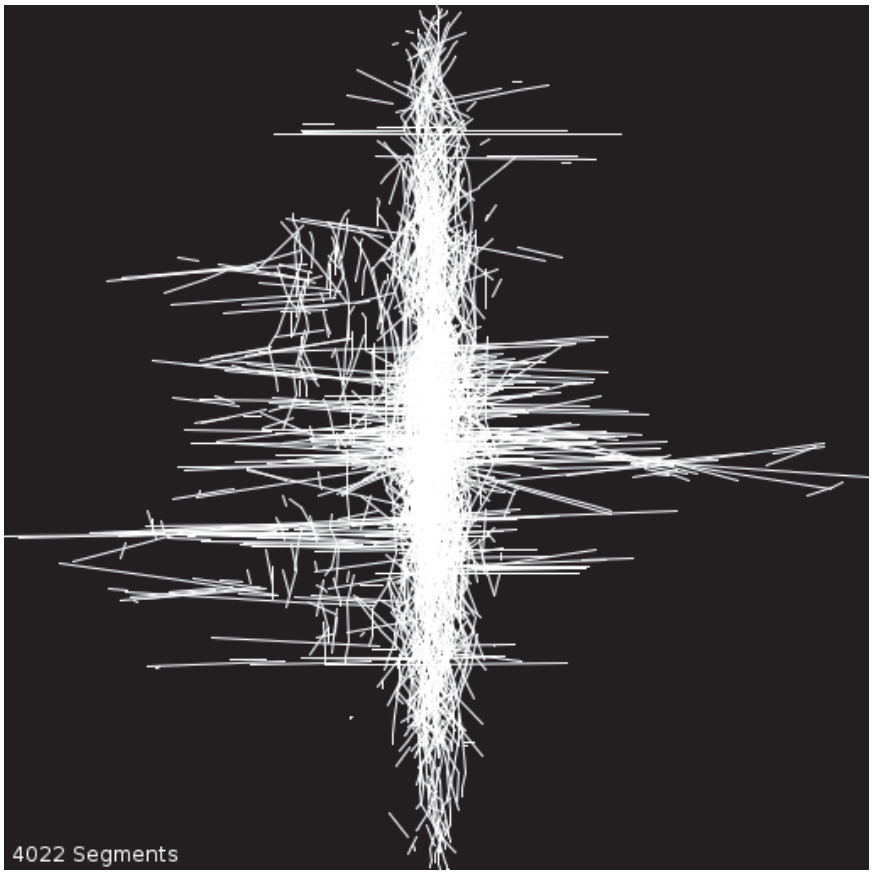

FIG. 3.9. Density-based clustering (unconstrained data). For unconstrained data sets, the density-based approach had difficulty growing a cohesive trajectory through the data set.

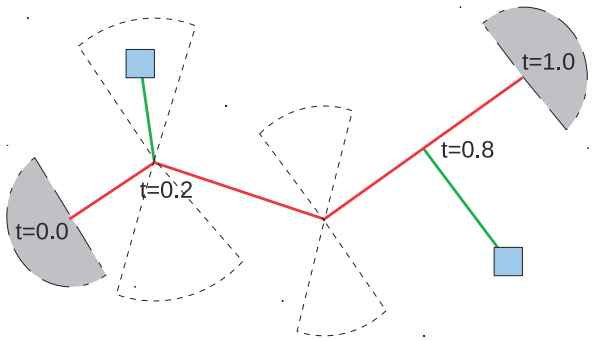

FIG. 3.10. Over- and underrepresentation diagram. Curves in the average trajectory (red line) cause areas of over- and underrepresentation (dotted pie slices) for contributing trajectories on the curve's outside and inside, respectively.

attribute changes over the course of the clustered trajectories and the variance among them. Together, these features enable a viewer to gauge the success of the clustering phase, determine the dependencies of the attribute over the trajectories, and understand the variability in the attribute.

To satisfy these composition goals, the cluster's average trajectory was chosen as a common frame of reference. Each clustered trajectory, called a "contributing trajectory" throughout this section, is transformed to the reference frame for composition. The transformation then ensures that the contributing trajectory accurately influences the appropriate pixels in the common reference frame. The challenge for the transformation stage is to overcome areas where the average trajectory has high curvature. In these locations, two problems occur: over- and underrepresentation of the transformed trajectories. For example, trajectories on the outside of the curve become over-represented because they sweep across a wider radius. Trajectories on the inside suffer from the opposite problem—underrepresentation—because they are "pinched" on the inside track. This means that over-

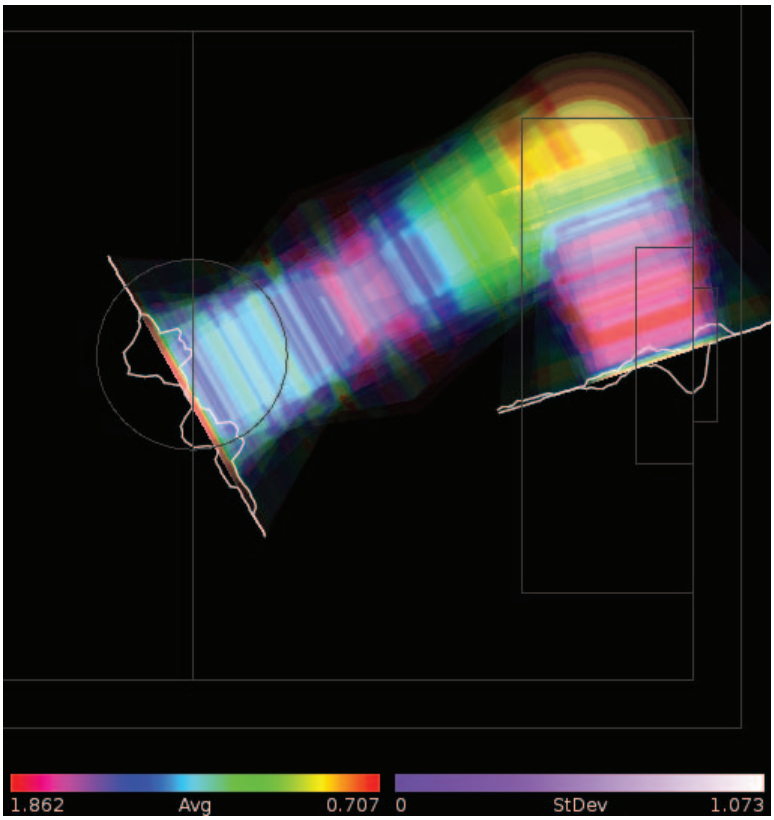

FIG. 3.11. Over- and underrepresentation example. Note the overrepresented region towards the top of the trajectory (yellow) and the underrepresented section opposite. The underrepresented area causes legitimate values to be overwritten by the incorrect values.

represented trajectories will occupy a larger area than their under-represented equivalents in the final composition. Figure 3.10 shows a notional diagram of the areas in contention as dotted pie slices around the average trajectory's points. For the first set (from the left), the top slice indicates overrepresentation will occur—overrepresentation does not impact other valid areas within the visualization but instead causes the same value to be replicated over the width of the slice. However, underrepresentation occurs on the lower half of the first slice. Unlike overrepresentation, underrepresentation does interfere with legitimate parts of the visualization. Because the slice covers the area for two trajectory segments, a decision has to be made for what data will be represented. Figure 3.11 depicts a real-world example of this problem from the robotic soccer data. In the figure, the upper bend over-represents a single set of data points as they sweep around the edge of the diagram. The under-represented parts are "crimped" in the lower part of the same curve. This figure also depicts a limitation in the naive approach to compositing the trajectories: if over- and underrepresentation areas are not resolved prior to composition, then legitimate data is overwritten by the simple approach. To consistently transform the contributing trajectories and ease the tension from over- and underrepresentation, a modified Level Set approach was used to construct a transformation algorithm.

3.4 Average Trajectory Transformation

Level Sets (Sethian 1999) model the expansion of a two-dimensional boundary over time from an initial starting condition. The starting condition represents the boundary at the initial time (i.e., t_0). By iteratively expanding the boundary, the Level Set algorithm determines how the boundary propagates and when it will reach a particular location. For a boundary that expands at a constant rate, the time to reach a point is equivalent to the distance to the initial boundary. Figure 3.12 depicts two examples of the Level Set algorithm.

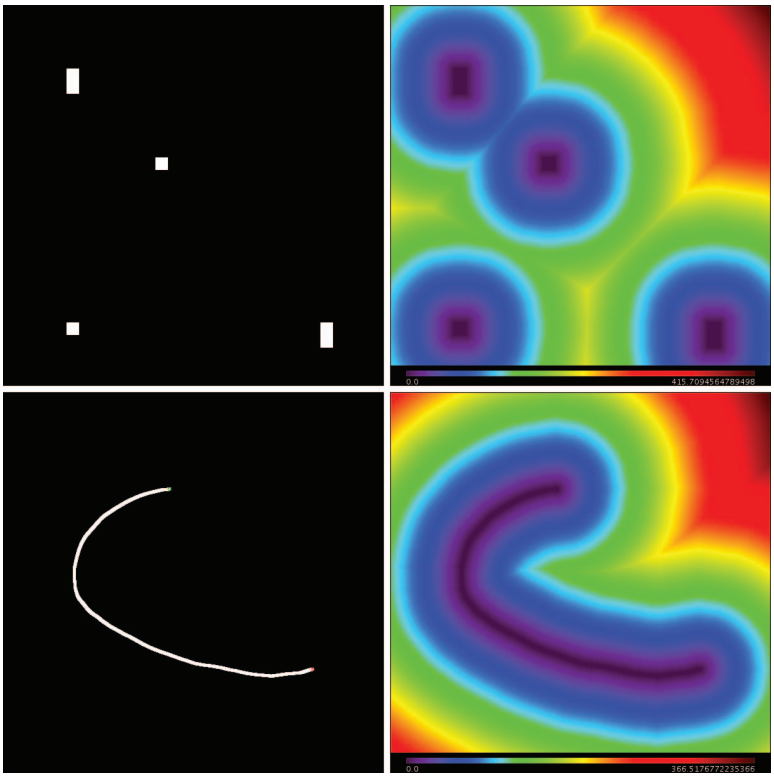

FIG. 3.12. Level set examples. Level Sets are initialized with a starting condition (left) and record the expansion time from that starting condition (right).

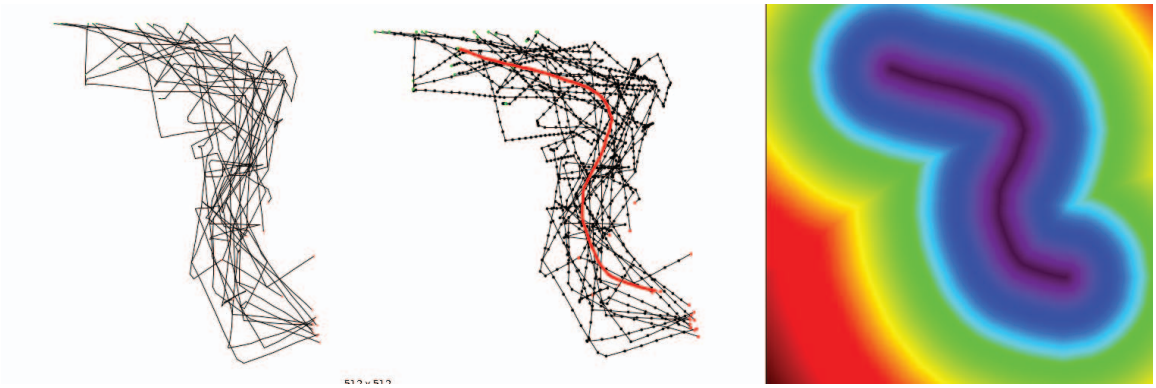

FIG. 3.13. Level set construction for trajectory cluster.

The initial starting conditions are shown on the left side of the figure, and the results of the Level Set algorithm are on the right. The coloring of the results represents the boundary expansion time (and distance from initial boundary starting condition). Purple represents t_0. The colors then transition from purple to blue, cyan, green, yellow, and finally, red, which are the points last reached by the boundary and are the furthest from the starting condition.

In order to construct the transformation, Level Sets are used to capture the boundary of the average trajectory (the initial starting condition) as it expands across the rendering surface. This expansion captures the distance of each pixel on the rendering surface from the average trajectory—recall that time and distance are equivalent for boundaries that expand at a constant rate. Due to the Level Set properties, areas of underrepresentation are resolved inline with the stated goals—i.e., areas of contention are correctly annotated with the closest distance and the corresponding parametric value. Overrepresentation still occurs and is an area of future work. The lower half of Figure 3.12 shows an example of a Level Set for an average trajectory. The "pinching" occurs on the inside of the "C", but the Level Set gives a smooth seam across this area. As an example, Figure 3.13 shows the original trajectories (left), the average trajectory (in red, middle tile), and the resulting

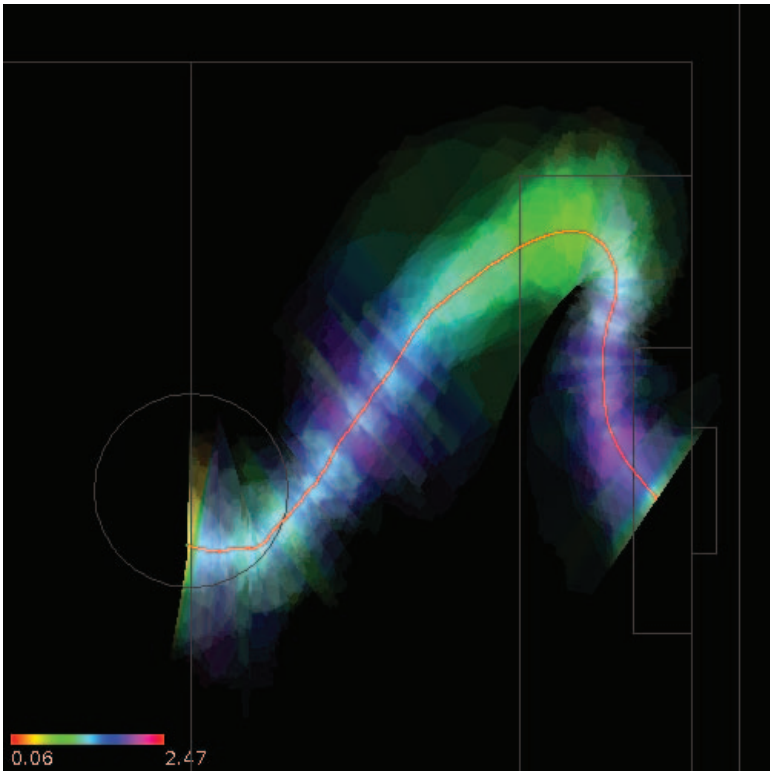

FIG. 3.14. Over- and underrepresentation correction.

Level Set for the trajectory (right).

The Level Set algorithm is further modified to propagate, and capture, the parametric values from the average trajectory. The end result for the modified Level Set algorithm is that, for each pixel, the distance to the closest point on the average trajectory and the parametric value at that point are captured. Figure 3.10 depicts the information stored for two such pixels. In the figure, the closest distance is shown in green and the parametric value at that closest point is enumerated (t value). In the first pixel's case ($t = 0.2$), its associated parametric value is the same as the discrete average trajectory point forming the first bend and occurs in an area of overrepresentation. The second pixel's parametric value ($t = 0.8$) is interpolated along the average trajectory's last segment. Since the value was interpolated and not from a discrete point, no over- or underrepresentation will occur.

Figure 3.14 shows a trajectory composition using the Level Set method to resolve under- and overrepresentation. Note that in the figure, abrupt changes in attributes are no longer apparent as with the earlier examples (i.e., Figure 3.11).

The average trajectory's end points are handled as special cases during the Level Set algorithm. Contributing trajectory points that are closest to either end point are ignored—spatially, these ignored points are located in the two gray semi-circles in Figure 3.10. If they were included, the final composition would contain a large "bubble" on each end that replicates the end point contribution. These two bubbles would overpower the image without providing additional, relevant information to the viewer.

Once the modified Level Set algorithm determines the parametric and distance values for each pixel, the contributing trajectories are transformed and composited together for the final rendering. The final objective for this stage is to accumulate the individual trajectory contributions for each pixel in the composition; the contributions are captured as a list of trajectory attribute values for the corresponding trajectories that influence a particular pixel. The per-pixel list of attribute values is then used to select the pixel's color and is described in more detail in the next chapter.

Individual trajectory contributions can be handled in one of three ways: localized, symmetric, or asymmetric (Figure 3.15). When the accumulation is localized, the individual trajectory only contributes to a few pixels at each corresponding parametric (t) and distance value (d). Recall that the distance value is calculated with respect to the trajectory's distance from the average trajectory. The localized application most closely resembles the contributing trajectories and provides the most granular composition. Both symmetric and asymmetric extend the contribution from the distance down to the average trajectory, referred to as the core. The key difference is that the symmetric version accounts for the signed distance from the average trajectory. Test data examples enumerating each type are shown in Figures 3.20, 3.21, and 3.22 and will be discussed later.

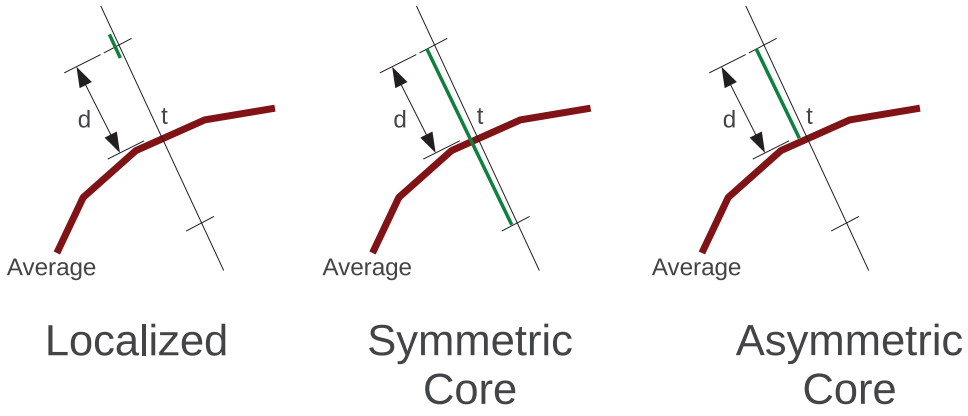

FIG. 3.15. Accumulation options. From left to right: localized accumulation, symmetric core accumulation, and asymmetric core accumulation. Accumulated locations are represented by the thick green line. The t-value represents the parametric value for the corresponding average trajectory, and the d-value is the contributing trajectory's distance from the average trajectory.

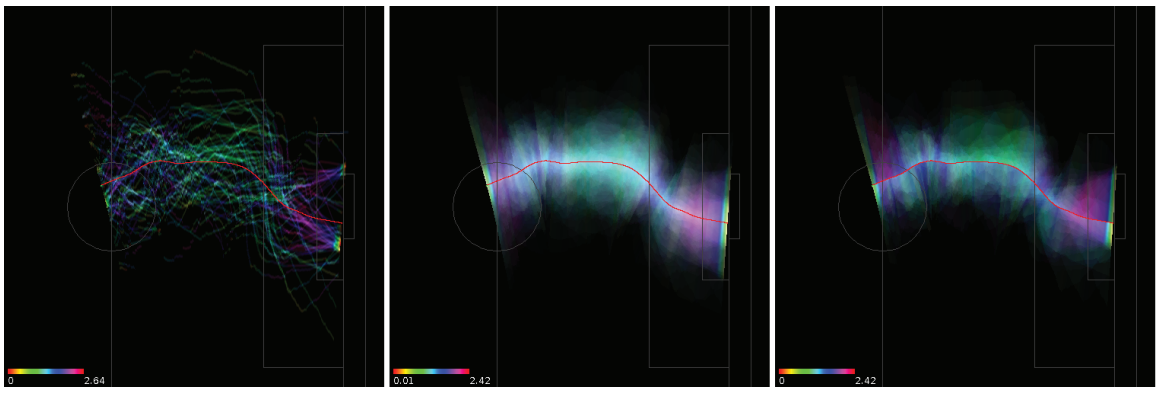

FIG. 3.16. Accumulation option examples. From left to right, localized accumulation, symmetric core accumulation, and asymmetric core accumulation.

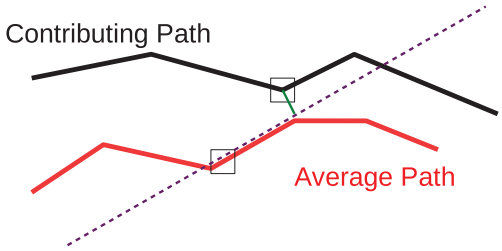

FIG. 3.17. Orthogonal distance. During the composition phase, the contributing trajectory's closest distance to the tangent of the corresponding average trajectory point is used.

3.5 Visualization Composition Process

For each pixel, the contributing trajectories that influence its appearance need to be determined. To accomplish this, the corresponding spatial location is determined on each contributing trajectory on a pixel-by-pixel basis—recall that each pixel has a parametric value associated with it as a result of the Level Set. The parametric value is then used to interpolate along the contributing trajectory to find its corresponding spatial location. This spatial location is then compared with the average trajectory to determine the distance and relative position (above or below the average trajectory) in world space. If the distance is within a specified delta of the pixel's distance value (i.e., the Level Sets expansion time), then this trajectory contributes to the pixel. The delta is derived empirically and ensures that each trajectory contributes to a perceivable area. To store the actual contribution, the trajectory's attribute value is also interpolated at this parametric value and added to this pixel's list of attribute values. The implementation must guarantee that a single trajectory can only contribute to each pixel at most once.

During the development of the composition process, a consistent "valley" in the final

rendering along the average trajectory was observed. Examining how the average trajectory was formed revealed that the Euclidean distance measurement was the cause. Since each average point is centered within the corresponding points on the contributing trajectories, the point-to-point distances are larger than would be expected, causing the "valley." However, using the orthogonal distance from each contributing point to the average trajectory's tangent produces more intuitive results. Because the trajectories are composited in reference to the average trajectory, the closest orthogonal distance produces a more appealing transformation—i.e., orthogonal measurements are preferred in the common reference frame. Figure 3.17 illustrates this calculation. In the figure, the corresponding points on both the average trajectory and the contributing trajectory are highlighted by the small squares. The tangent along the average is denoted by a dotted purple line. The tangential distance used by the composition is the closest point on the tangent, shown as a green line.

3.6 Color Composition

The final phase of the composition process is to assign color values to each pixel in the final rendering. Several goals need to be satisfied by the chosen scheme. First, the spatial density of the contributing trajectories needs to be effectively depicted, providing the viewer with a clear understanding of the cluster's spatial structure. Second, the trajectory attribute values should be discernible by the viewer in their corresponding spatial location—by properly showing this value, the attribute trends along the trajectory can be determined. Finally, if the contributing trajectory attributes have a wide variance, that should be depicted as well. In general, pixel brightness provides an intuitive representation for areas of high density, and pixel hue is a common choice for attribute values.

From the structure composition step described in the last section, each pixel has a list of attribute values from the trajectories that spatially contribute at this location. The

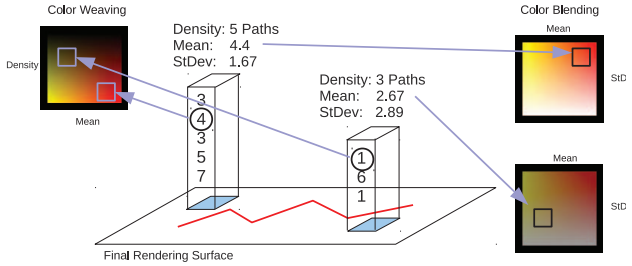

FIG. 3.18. Color composition. The attribute lists created earlier in the composition process can be used to color the image through blending or weaving.

list of values can be used to directly derive parameters for the color model. The length of the list is directly proportional to the composition density and the mean and standard deviation can be readily calculated. Figure 3.18 depicts the list of attribute values for two example pixels and the resulting statistics from these lists. In the figure, the common reference frame is represented on the bottom as the *final rendering surface*. The two pixels are shaded and the list attribute values are shown as a column of values. From each of the lists, the density, mean, and standard deviation are calculated (as written at the top of each column). These statistics are used to select an appropriate color, as described in the next section. Additionally, Figure 5.3 provides a similar example from the second use case.

3.7 Blending Method

Two separate color compositing schemes, blending and weaving, were chosen to effectively portray the attribute data. The first method uses a color blending approach (Chlan & Rheingans 2005) to account for the density, mean, and standard deviation, and the other method weaves colors (Hagh-Shenas *et al.* 2007) together to show the attribute value from a randomly selected contributing trajectory. To render the color-blended composition, the mean, standard deviation, and density value feed into the hue, saturation, and brightness

(HSB) of the color value, respectively. In order to satisfy the specific principles described by Rheingans (2000), the hue component is constrained to only cover two or three colors from the entire color spectrum. By modulating the saturation by the standard deviation, areas where most of the contributing trajectories have similar attribute values deepen as saturated colors. Areas where the contributing trajectories disagree or have a wide variance have low saturation and appear washed out (i.e., white or gray). Finally, density corresponds directly with the brightness component, and, against a black background, provides an opaque representation where the maximum density is reached.

In Figure 3.18, the color blending step is shown on the right side of the figure. The first example pixel has more trajectories contributing to the composition and therefore has a brighter color, since the density is higher. The first pixel also has a higher mean value and a lower variance, resulting in a higher saturation on the upper end of the color scale. On the other hand, the second example pixel has fewer trajectories (less density) and therefore uses a darker color scale that, when placed against a black background, provides the intuitive appearance of less density. The second pixel has a lower mean value and a higher variance, resulting in a choice from the lower end of the color scale that is significantly more washed out.

Blending's primary advantage is that every data point contributes to a pixel and all three aspects (i.e. attribute value, attribute deviation, and trajectory density) are represented by the visualization. However, to achieve desirable results, it is often necessary to manually manipulate the color range for the mean values (hue) and the standard deviation (saturation). This can be complicated if multiple visualizations need to be compared—adjusting each visualization separately results in visualizations that cannot be directly compared against one another. Depending on the color scale chosen, it may also be difficult for the viewer to determine the mean value in low density areas of the image. The difficulty lies in human perception correctly understanding the hue of the pixel in darker regions.

3.8 Weaving Method

The weaving method, the second color compositing scheme, also uses the same hue, saturation, and brightness color model, but does not modulate the saturation by the standard deviation. Instead, the algorithm randomly chooses a contributing trajectory's attribute value to select the appropriate hue component for the woven cell. The density is still used with the brightness component to correctly convey the density and therefore the cluster's overall structure. Weaving provides the viewer with an exact value of a single data point in that region; deviation must be inferred by looking at neighboring point samples. On the left side of Figure 3.18, a simple depiction of how the woven color is chosen. First, a random value is chosen from each pixel list. This value chooses the woven color and the density at this location determines the brightness.

Weaving's primary advantage is that every pixel represents an actual data point from the trajectory set. By picking a random trajectory for each pixel (or local area), actual data points more accurately depict the underlying trajectories. Standard deviation is more difficult to determine, especially on a qualitative basis. While visual perception can pick out areas of high deviation due to the mix of high variances in the woven colors, determining the qualitative deviation is a challenge.

To illustrate the strengths and weaknesses of each approach, Figure 3.19 depicts each method applied to the different accumulation models. Blending produces intuitive results across all three accumulation models (localized, symmetric core, and asymmetric core). However, due to the subtle variances in the saturation, the attribute trending information is difficult to determine except at a very granular level especially within both core visualizations (middle and bottom left). It should be noted that the asymmetric version (bottom left) has more saturated colors because less variability occurs on each side of the average. The blending approach applied locally (top left) produces better results for interpreting

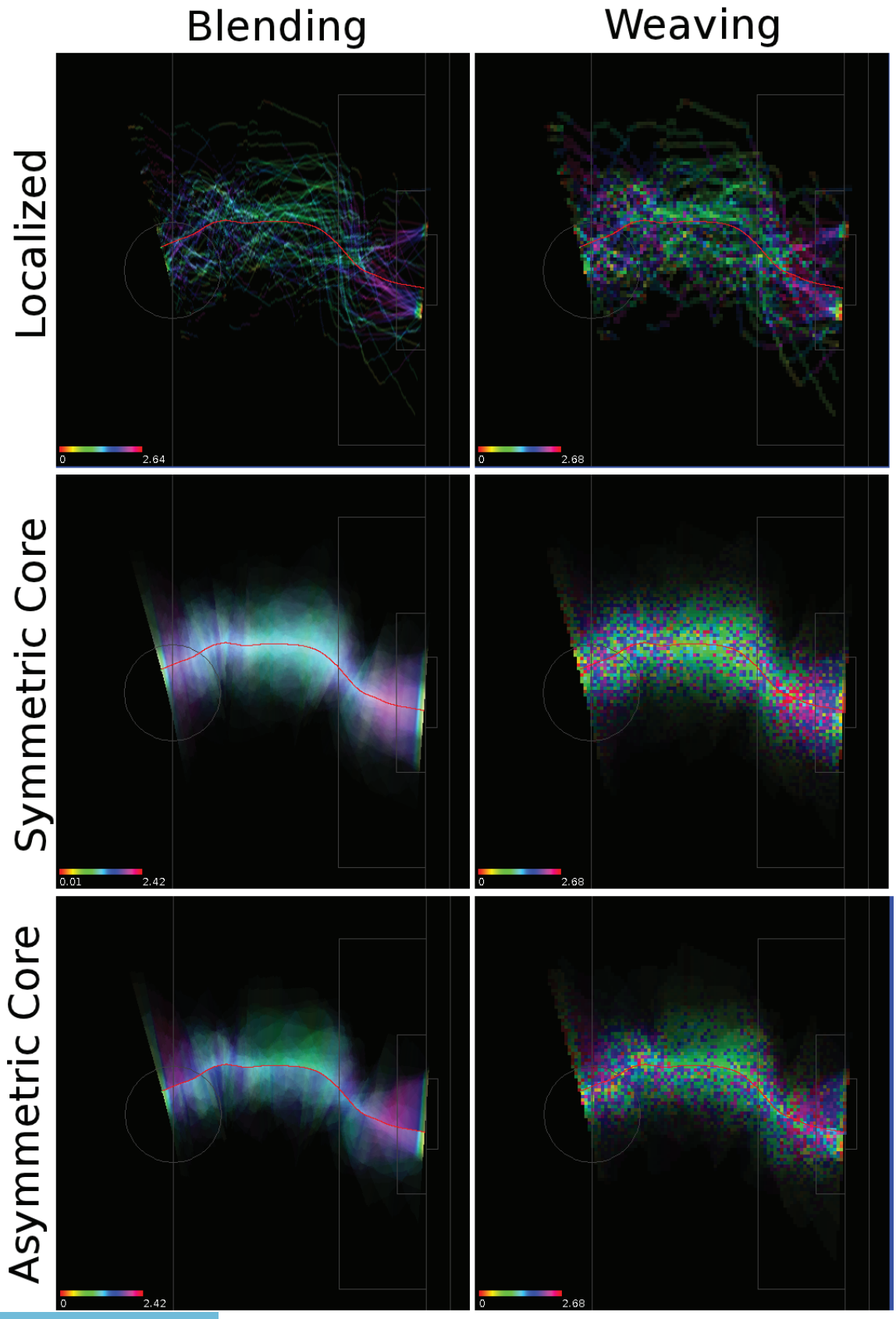

FIG. 3.19. Blending versus weaving. The tiles show the distinct differences between using blending (left) versus weaving (right) color models. Vertically, the tiles show localized (top), symmetric core (middle), and asymmetric core (bottom) comparisons.

the mean value (i.e. hue). Weaving, on the other hand, produces stronger results in both core visualizations (middle and bottom right). For example, the middle section in the symmetric core (middle right) is especially strong for seeing the constant attribute value. The localized, weaving results (top right) do not maintain the cohesive nature of the blended counterpart (top left).

3.9 Test Data

Figures 3.20, 3.21, and 3.22 show sample test data for 10, 100, and 1000 synthetic trajectories, respectively. Except for the top row (which shows the original, non-composited data), each figure provides examples, from left to right, of symmetric core, asymmetric core, and localized compositions, respectively. Furthermore, the second and third row use the closest point calculation for determining trajectory contributions while the fourth and fifth row use the tangential distance. Lastly, the second and fourth row use a blended approach while the third and fifth row show the woven techniques.

As one can see, the composition becomes smoother and less disjoint as more trajectories are added. In fact, the algorithm produces better results with significantly more data, achieving the stated goal of compositing thousands of similar trajectories at once. However, since this is a synthetic data set, it should be noted that actual trajectory data would need to be highly cohesive to achieve similar results.

Although early testing led to the use of the tangent method (fourth and fifth rows) for accumulating values, the results show that the closest point method (second and third rows) produces stronger results. For example, in the localized versions (third column), the bifurcation of the trajectories is clean while the tangent method produces an overlapping result. Since the test data was derived by both uniformly offsetting the individual trajectories from the base curve *and* uniformly offsetting each point by a smaller amount, the correct result

should be the bifurcated representation. This bifurcation becomes much more apparent as the number of accumulated trajectories grow.

One notable difference becomes apparent between the localized (third column) and the core types (first and second columns). Since a uniform distribution was used to perturb the synthetic data away from the average trajectory, the two uniform curves become apparent in the localized version. This is not directly apparent in the core version, although it can be inferred from the uniform distribution of density (i.e. very little dimming of the color).

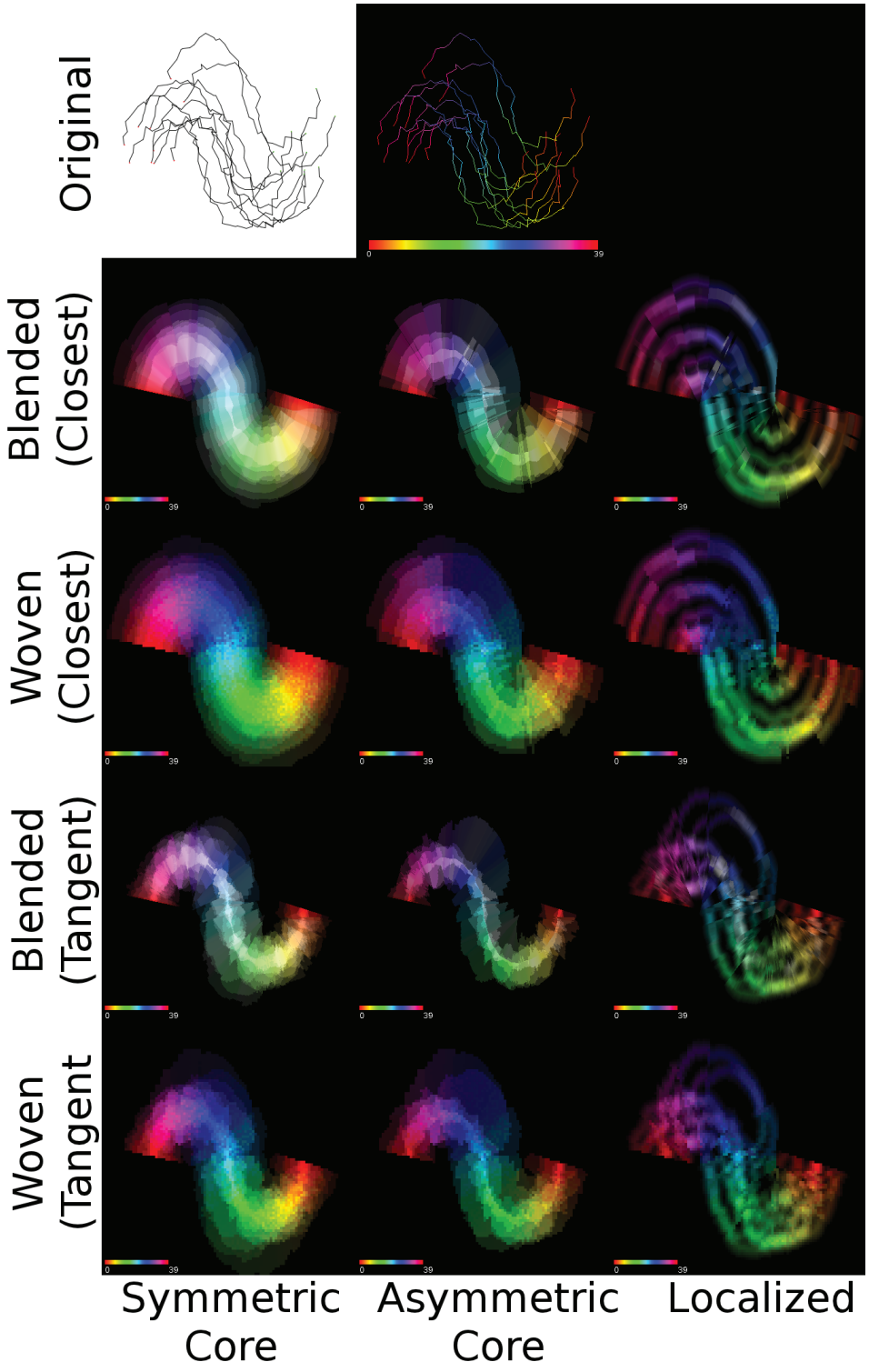

FIG. 3.20. Test data, 10 trajectories

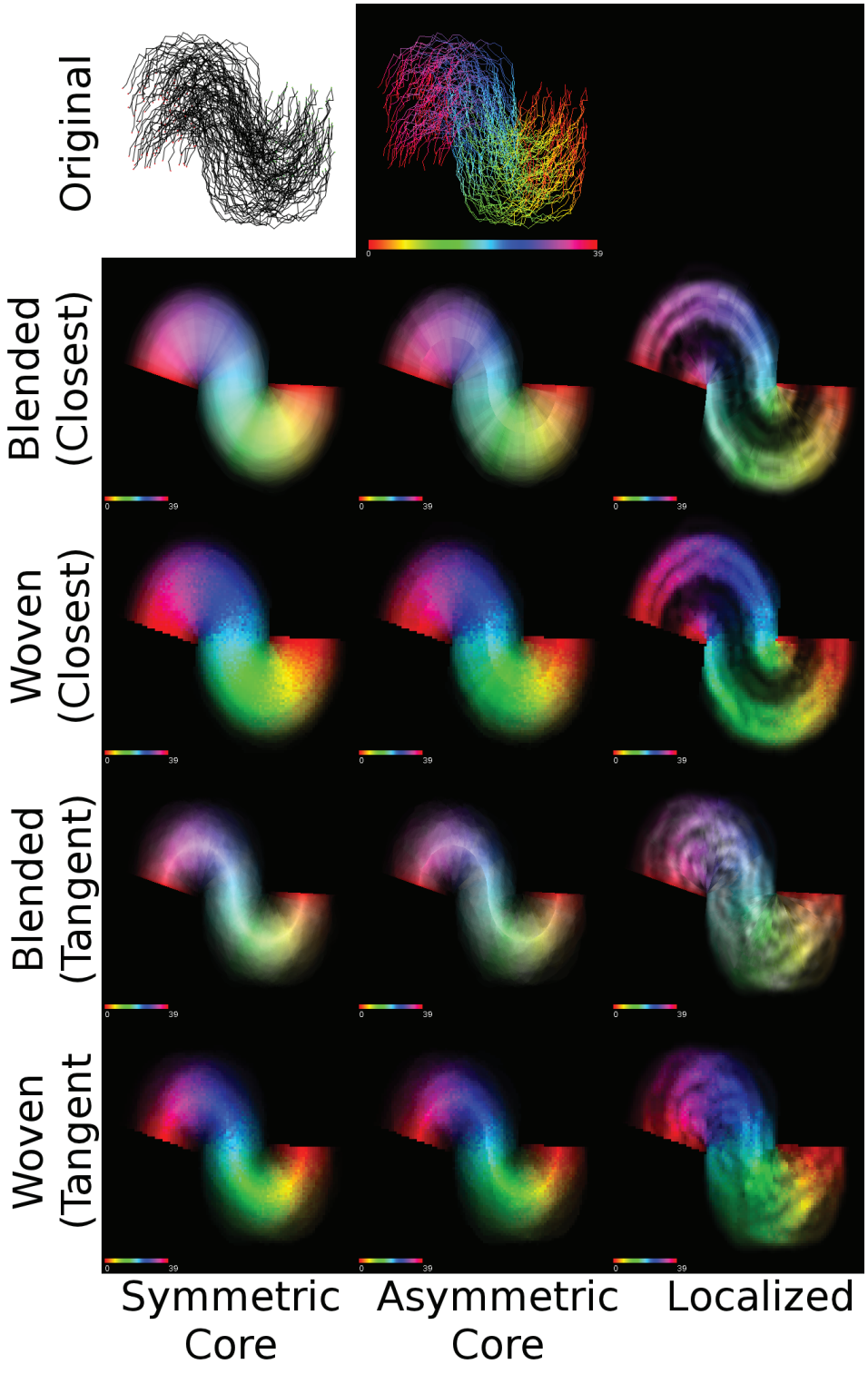

FIG. 3.21. Test data, 100 trajectories

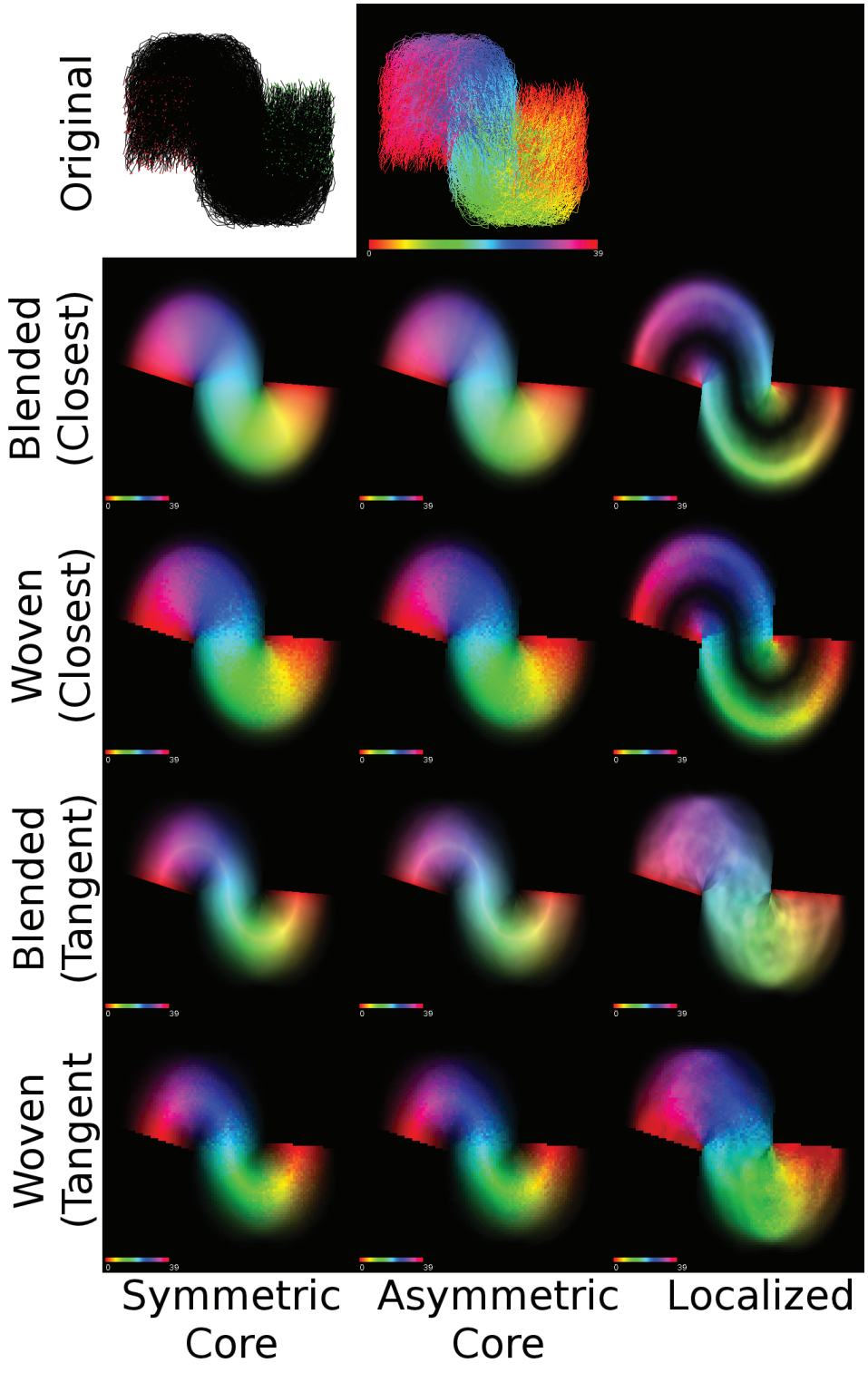

FIG. 3.22. Test data, 1000 trajectories

Chapter 4

APPLICATION: ROBOCUP SOCCER

To demonstrate the effectiveness of the approach, trajectories derived from artificial intelligence multi-agent systems were examined using both traditional visualization techniques and the novel approach of trajectory clustering and composition. The following case study provides real-world examples of how trajectory composition can aid in understanding large trajectory populations. From the analyst's perspective, several requirements need to be satisfied by the visualization process. First, the analyst needs to investigate thousands of trajectories produced from multiple, simulated multi-agent systems. These trajectories need to be grouped with other, similar trajectories for further analysis. The resultant groupings need to be validated by the analyst to ensure that they do indeed belong with each other. Finally, when analyzing each individual group of trajectories, the analyst needs to understand how specific, application-dependent attributes vary both along the trajectory and across the group. The goal for analyzing these attributes is to improve the overall algorithm behind the multi-agent system.

4.1 RoboCup Soccer Background and Previous Work

RoboCup is a soccer-based framework for research in multiple artificial intelligence disciplines, including strategy and planning, play mechanics, and collaboration among in-

dependent entities. Teams regularly meet to compete in tournaments and there is a wealth of research into different methods for solving each aspect of RoboCup soccer. For example, several papers (Salmani, Fard, & Naghibzadeh 2006)(Zafarani & Yazdchi 2007)(Whiteson *et al.* 2005) have proposed methodologies for how each AI player can optimize its behavior to achieve the best results for the team. There are multiple leagues within RoboCup, exploring multiple AI challenges, from robotics to team-based collaboration. The focus of the analysis within this paper is the two-dimensional simulation league, which provides ample data for examining most RoboCup multi-agent problems, except robotics.

Several methods are currently used to analyze the performance of RoboCup teams. These methods include looking at the overall scores from matches, gathering statistics from plays or other game mechanics, and simplified simulations such as keep-away. As within other areas of artificial intelligence, many of the training algorithms used have their own intrinsic methods for determining a team or player's performance. These methods are used for either unstructured or directed learning. Another, more straightforward, method used by researchers is to just watch the matches as you would a regular soccer match and identify deficient heuristics or other suboptimal strategies.

Several methodologies have been used to analyze RoboCup soccer matches. Raines et al. (1999) developed automated techniques to analyze the behavior of the agents independent of the particular domain. Data mining and inductive logic are used to determine success and failure cases on derived state variables directly related to success. Riley and Veloso (2000) focused on adapting the strategy of multi-agent teams, as well as the individual agents, to more effectively overcome an adversary. To accomplish this, behavior is broken down into distinct classes. Visser el al. (2002) also developed an adaptive strategy to maximize the success of player passing using decision tree induction. While these techniques have focused on automated ways to analyze performance, the need for information visualization of RoboCup data to understand and improve team performance has not been

addressed.

Data visualization is a natural choice for analyzing the data generated from simulated RoboCup matches. The logs produced by the simulation contain an exact representation of almost all of the events that occurred within the match. However, several problems are apparent. The first is that the logs do not contain information about the internal state of the artificial intelligence algorithm running within each team member. Therefore, it is difficult to gauge the overall intent of the player. Because the internal agent state is not accessible, a second problem with the log data is exposed: it only contains very low-level mechanics, so one must extensively process the logs to generate information about ball possession, whether the intent behind a kick was to dribble or pass, which team members were seen by a player, etc. For the visualizations in this chapter, the trajectory followed by the soccer ball during a team's *possession* was extracted from the log data. A *possession* is defined as a contiguous set of game states during which one team controls the ball. To provide more density for the clustering algorithm, the field is normalized with respect to the possessing team.

4.2 Traditional Visualization Techniques

Traditional visualization techniques are applied to the RoboCup dataset first. These techniques have general applicability across a wide array of domains and are universally acknowledged for their intuitive nature, acceptance across users, and ability to provide scalable insight into data. By first studying how traditional visualization can provide insight, the proposed trajectory composition techniques can be compared and contrasted to determine if value is added. The goal with respect to the traditional techniques is to determine what additional insights can be derived from spatial trajectory data.

4.3 Analytical Visualizations

My work uses several familiar, as well as novel, visualizations for analyzing multi-agent systems. Visualization is a natural choice for analyzing the large amounts of data that is generated from multi-agent systems. To be effective, the data needs to be portrayed in an intuitive manner. While this may be straightforward for spatial aspects, creative ways to depict non-spatial information within their spatial context were developed. This combination enables multi-agent researchers to fully understand an agent's behavior and overall system coordination.

More specifically, visualizations that summarize simulated multi-agent environments were developed. These summarization views are useful for quickly reviewing large amounts of data to determine points of interest. These points of interest may represent either successful behaviors that need to be reinforced or sub-optimal strategies that require algorithmic improvements. The summarizations capture both spatial and non-spatial aspects and allow analysts to drill down into simulation data for further examination. The summarizations clearly portray relevant parts of the data produced by multi-agent simulations and key attributes for each agent for further analysis.

Tailored versions of parallel coordinates are also provided to analyze specific attributes for each agent within the multi-agent simulation. This methodology is applied to the strategy behind teams of multi-agents to cover both shared common views and global strategies. The methods rely on extracting domain-specific attributes that were identified as important to the success of the multi-agent teams. These attributes are then assembled into multivariate constructs for use within a parallel coordinate visualization. Furthermore, enhanced parallel coordinate views that overlay histograms in addition to the attribute relationships are also created. To overcome some of the limitations of parallel coordinates for this application, a filtering mechanism was added to separate competing multi-agent teams, color to

separate effective strategies from ineffective ones, and clustering to partition similar samples.

4.4 Visualization Goals

The research focuses on several key goals for analyzing multi-agent systems. The first provides effective summarizations of both spatial and non-spatial aspects of simulated multi-agent systems. A key requirement is to extract relevant spatial aspects related to the goal for the multi-agent system while providing specific spatial and non-spatial information that determines each agent's choices and actions. As a further complication, the analytic must also capture the agents' collaborative relationships. To satisfy this requirement, both types of data are combined: the spatial data provides the overall goal-oriented information and the non-spatial information is mapped within the spatial context so that the relationship is clear. The goal is to enable quick review of the simulation to determine interesting events that can then be further analyzed.

A second goal of the visualizations is to depict domain-specific behavior of individual agents and the overall multi-agent strategy. Key requirements include representing attributes critical to the success of an agent's behavior and the interrelationships with other attributes. This methodology is also applied to multiple agents including their collaboration, shared common view, and overall strategy. This is accomplished by extracting domain-specific attributes related to the goals and objectives of the multi-agent team and using a modified parallel coordinates view for comprehension. By providing this visualization, analysts can understand the complex interactions among key features of the multi-agent system, enabling performance improvements in their design.

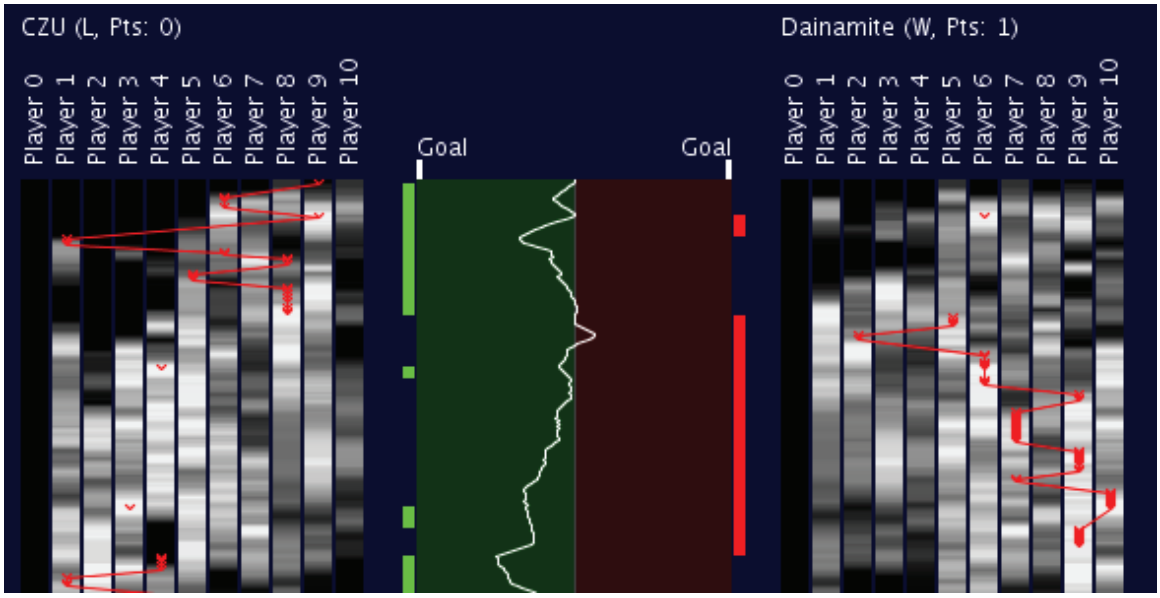

FIG. 4.1. Match summary visualization. A RoboCup match is succinctly depicted to track ball possession, ball field position, collaborative play mechanics, and specific player attributes. The displayed attribute, represented as a gray scale value, is the minimum distance to an adversarial player.

4.5 Match Summarization

Analyzing a complete RoboCup match is a tedious process when viewed as the straightforward animation produced by the RoboCup simulator. Match summarization is a natural choice for visualization because the critical match components can be extracted and depicted within a single view. These critical components include the ball position relative to the goals, ball possession, coordination among team members to conduct plays, and specific attributes about each player throughout the match. Most of these are not directly provided by the RoboCup logs; they have to be determined from the low-level events. To analyze other multi-agent systems using this type of technique, application-specific components related to the goals of the system would need to be derived.

To satisfy these goals, a match trace visualization was constructed that can present

several RoboCup match elements at once. Figure 4.1 provides an example of this summary view. The match trace consists of a single pixel row for every time increment within the game. For RoboCup, this equates to each state within a match. The state is a single simulation increment where all simulation variables get updated. Each pixel row in the visualization depicts the ball position, ball possession, and a single attribute for all players within the match shown as a gray-scale value. To capture the spatial relationship, a simplified field is represented in the center of the visualizations to show the relative position of the ball between the goals; one team's side is tinted green, while the opposing side is red. The ball position itself is the white pixel on the simplified field in the middle. To provide a sense of time, these white pixels are connected with lines across each of the match states. Ball possession is denoted by the red and green bars on each side of the field representation. By providing this cue, one can quickly determine if the team that has possession is moving the ball down the field or not. As one can see, the components of this visualization capture key characteristics of the soccer match and depict critical changes over time to determine where interesting events occur. For example, suboptimal teams that can never push the ball past the center line are easy to identify because the center line is never crossed in the visualization.

To characterize players in the match, a column is devoted to each player showing a single user-chosen attribute as a gray-scale value. The choice of attributes include distance to the ball, stamina level, distance to the closest team member, and distance to the closest adversary. These attributes are clearly relevant to understanding how the match is progressing, the effectiveness of each side, and, on a larger level, the strategy employed by each team. Lighter shades represent a closer distance or more stamina, while darker regions represent a longer distance / less stamina. Long distances are capped at a distance of 10.0 to provide a higher degree of resolution for small values. Stamina basically equates to how much power a player can expend to move: the more stamina the player has, the faster they

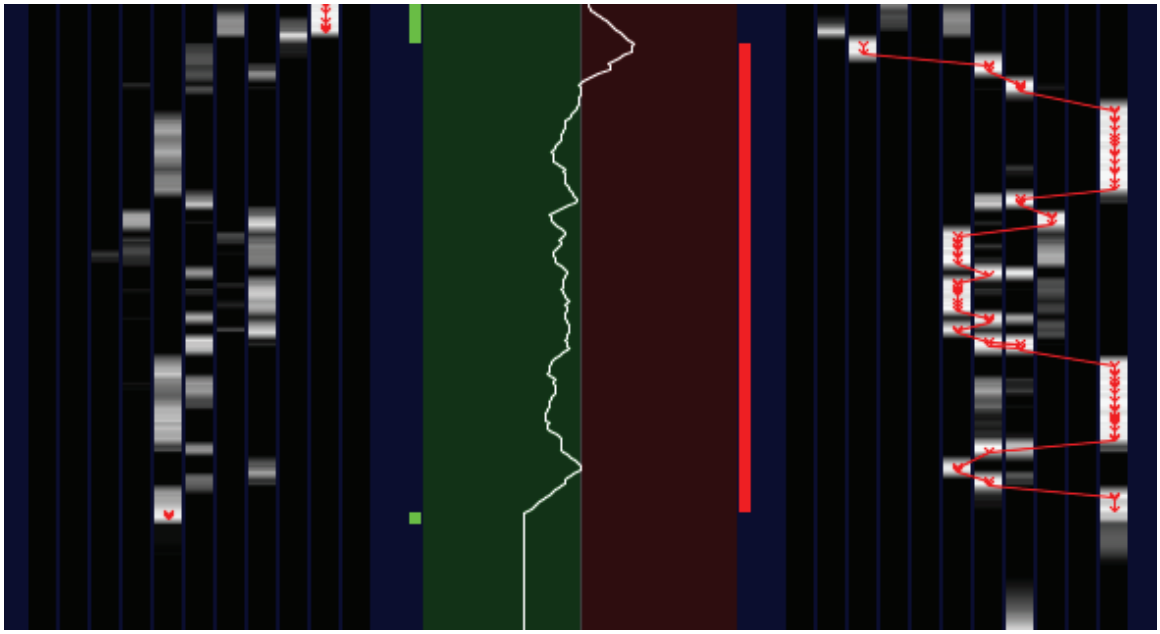

FIG. 4.2. Summary of complex passing. The team on the right executed a 16-pass play, resulting in only a small gain of field position. The displayed player attribute is the distance from the player to the ball.

can dash for the ball.

To provide information about how team members are collaborating, player kicks and individual ball possession are shown across player columns with a red line. Each player kick is represented as a tiny "v" or arrow symbol–a series of these symbols indicate dribbling activity. Changes of player possession can be determined because the line will transition from one player column to another player column within the same team. Each pass begins with a passing kick and ends with a receiving kick to "catch" the ball for further dribbling, passing, or scoring. This method of capturing player interactions provides a natural and intuitive reflection on how collaborative the team is and the complexity of its strategy for coordinating the agents. Figure 4.2 shows a relatively complex play pattern. As one can see, the team on the right has possession of the ball. Over the course of the play, six players are involved with multiple dribbles and passes. The ball possession portion in-

dicates that there was little forward progress until the pass towards the end of the play, at which point the team lost possession. Within the diagram, the player attribute has been mapped to the player's distance from the ball. Potential turnover points can be identified for each of the defensive players as their attribute intensity approaches white, indicating that they are almost on top of the ball. In the example discussed previously, two defensive players have several chances to intercept the ball, as shown by the diagram. This potentially provides context for why the offensive side continued to pass the ball between players. As one can see, highly collaborative plays become readily apparent. Scoring drives are easy to spot, and one can easily understand the events that led up to a goal shot, or, conversely, that resulted in a loss of possession.

Watching the traditional animated version of this part of the RoboCup match would require several minutes. However, the summarization can be viewed in less than a minute to determine points of interest and complex collaborative plays. In addition to decreasing the amount of time required, the summarization provides key indicators for how the team coordinates their individual movements. The traditional animation loses this information because each view of the match only shows a single instance in time–there is no context for past or future information. Historical information about who had possession of the ball or which player performed the original pass is lost. However, the summarization visualization clearly and intuitively depicts this complexity. The summarization view provided here shows all of the time steps within the game, enabling the viewer to visually see patterns of performance over time.

4.6 Analyzing Dribbles, Passes, and Plays

To effectively analyze agent mechanics, multi-agent collaboration, and multi-agent strategies, parallel coordinate views were developed that show derived parameters from

multi-agent simulations. Parallel coordinates were chosen because there is an expectation that the relationships between extracted values and task performance would have a strong correlation. Initial limitations of parallel coordinates led to tailoring the visualization to distinguish the result of the task performance for each multi-variate group, provide filtering for competing multi-agent systems, cluster similar multi-variate groups to partition data sets, and eventually derive a new type of parallel coordinate visualization for understanding how the different team strategies affect overall team performance.

To apply this methodology to RoboCup, multiple analytic passes for a RoboCup match were created that would automatically derive player possession, team possession, basic play mechanics (e.g., dribbles and passes), and composite plays (i.e., combinations of dribbles and passes while a team maintains possession). From each layer of the analysis, parameters and attributes based on a player's observable behavior are derived that could then be used to determine why one team's strategy or mechanics were better than another team's. Figure 4.3 depicts some of the attributes extracted for passes. *Ball velocity* and *kicker velocity* represent the velocity of the ball and the kicker's velocity, respectively, at the beginning of the pass. The *intercept distance* and *perpendicular distance* are calculated for the nearest team and adversary players to the pass center line. The *intercept distance* is the distance from the initial pass location to the perpendicular intersection on the center line, while the *perpendicular distance* calculates the players distance to the center line's position. The *pass angle* and *view angle* represent the angle between the kicker's direction and view, respectively, and the pass direction.

Figures 4.4 and 4.5 show the attributes extracted from players performing passing and dribbling, respectively, within the RoboCup tournament. In addition to the standard parallel coordinates view, the percentage of soccer passes and dribble mechanics that go through each vertical attribute axis were also displayed. The percentage is represented as a text value denoting percent and a gray scale value with white equivalent to 100%.

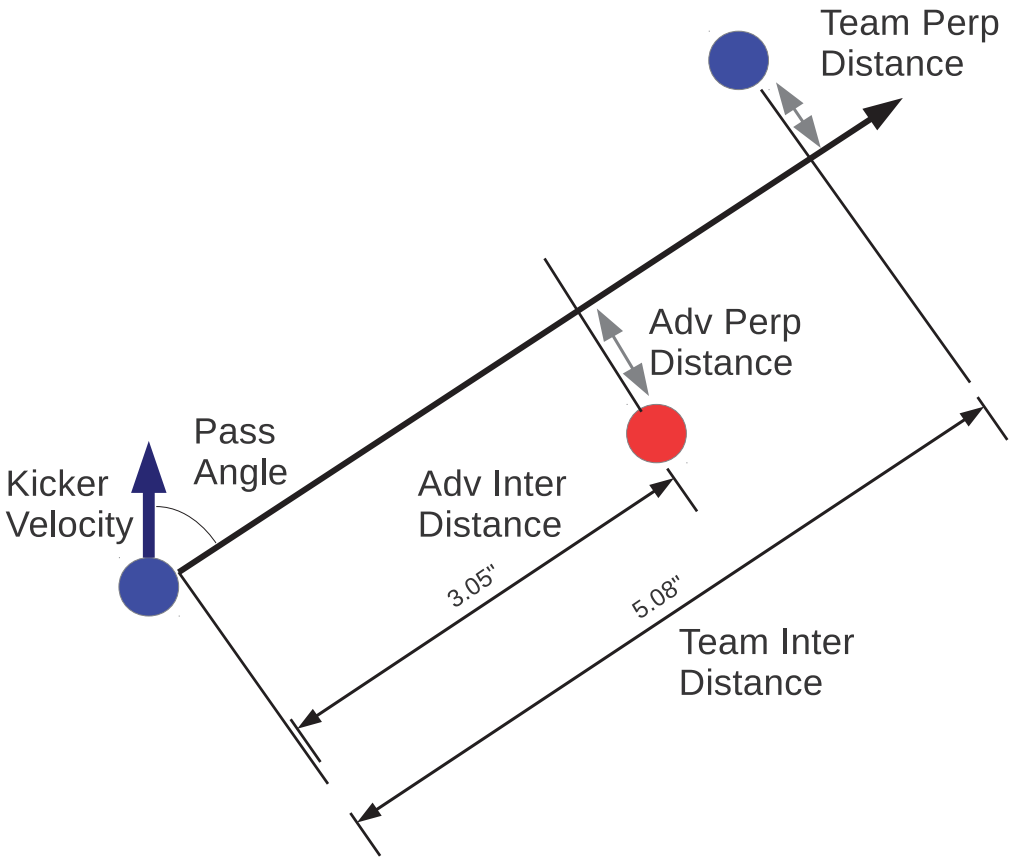

FIG. 4.3. Extracted attributes for RoboCup passes.

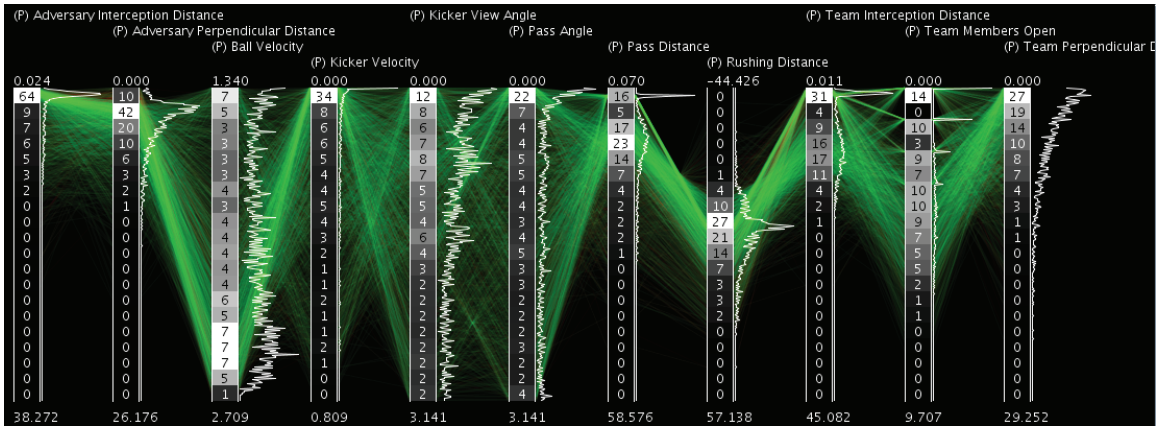

FIG. 4.4. Pass attributes for #1 ranked team. Twelve different attributes for all passes performed by the winning team throughout the tournament are displayed as a parallel coordinate view.

Without these additions, it is difficult to assess how many parallel coordinate lines traverse the attribute value. The range for each variable is placed on the top and bottom of the parallel coordinate axis to further gauge the actual value. However, because of the number of parallel coordinate lines, it is difficult to determine which attributes indicate success or whether the extracted attributes are even relevant. These limitations led to tailoring and modifying the parallel coordinate visualization to answer fundamental questions about multi-agent performance.

First, the overall distribution for each extracted attribute needs to be understood. To accomplish this, histogram information is overlaid to displayed multi-variate values next to the parallel coordinate axis for that attribute. Each histogram provided a quick reference for the overall trend within that attribute (e.g., whether the attribute had a normal distribution, uniform distribution, or bathtub-like curve) via a standard two-dimensional graph. The second limitation was the need to disambiguate successful plays from ones that were either ineffective against, or intercepted by, the opposing team. This limitation also applied to separating one RoboCup team from another. While applying color for different play

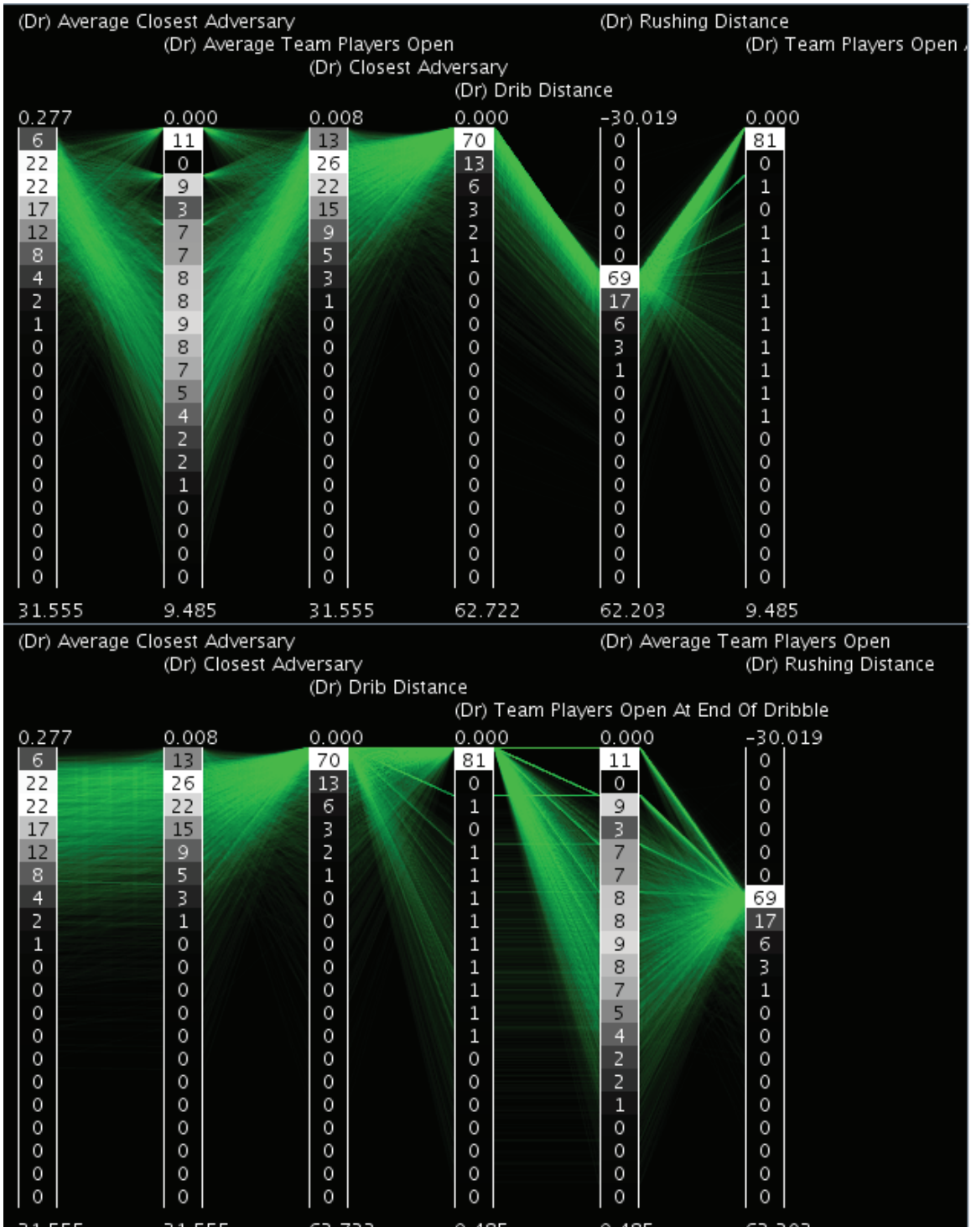

FIG. 4.5. Optimized parallel coordinates for successful dribbles for top three teams. The results from before (top) and after (bottom) the optimization algorithm are shown.

results helped with the understanding the success of the play, coloring each attribute line by its corresponding team was infeasible. The number of teams would quickly overload the number of distinguishable colors, due to the magnitude of parallel coordinate lines displayed and would obscure the result of the play. Interactive filtering was also provided so that a user could select which RoboCup team's attributes were displayed. This interactive capability allowed teams to be analyzed on an individual basis or across all of matches.

Figures 4.6, 4.7, and 4.8 use parallel coordinates to compare passing characteristics of the teams in the tournament. In each figure, the number one ranked team is shown on the top, while the lowest ranked one is on the bottom. The first figure, Figure 4.6, includes the successful, failed, and intercepted passes colored in green, orange, and red, respectively. Intercepted passes resulted in the opposing team gaining possession while failed passes accounted for neither team obtaining the ball (e.g., ball going out of bounds). A quick glance at the distribution of these values (far right column) shows that 81% of the best team's passes completed successfully, compared to only 47% of the worst team's. Both teams had a preference for a low kicker velocity but the best team's distribution was larger as shown by both the histogram on the first column (far left) and the percentile markers. The worst team had a preference for ninety-degree kicks, as both the pass angle and view angle show in the bottom portion of Figure 4.6—these were largely successful, as shown by the green coloring. However, while the best team preferred zero-angle kicks, the distribution was more uniform indicating better adaptability to situations in the tournament.

By filtering out all but the successful passes, a better understanding of team preferences in passing becomes clear (Figure 4.7). In particular, the worst team's deficiencies in passing are clearly evident in the bottom of the diagram. Both the pass angle and view angle (noted in the last paragraph) had tendencies for ninety-degree passes. Furthermore, the worst team also had a preference to pass the ball when the adversary perpendicular distance was approximately 12, as shown by the spike in the second to the right-most column in the

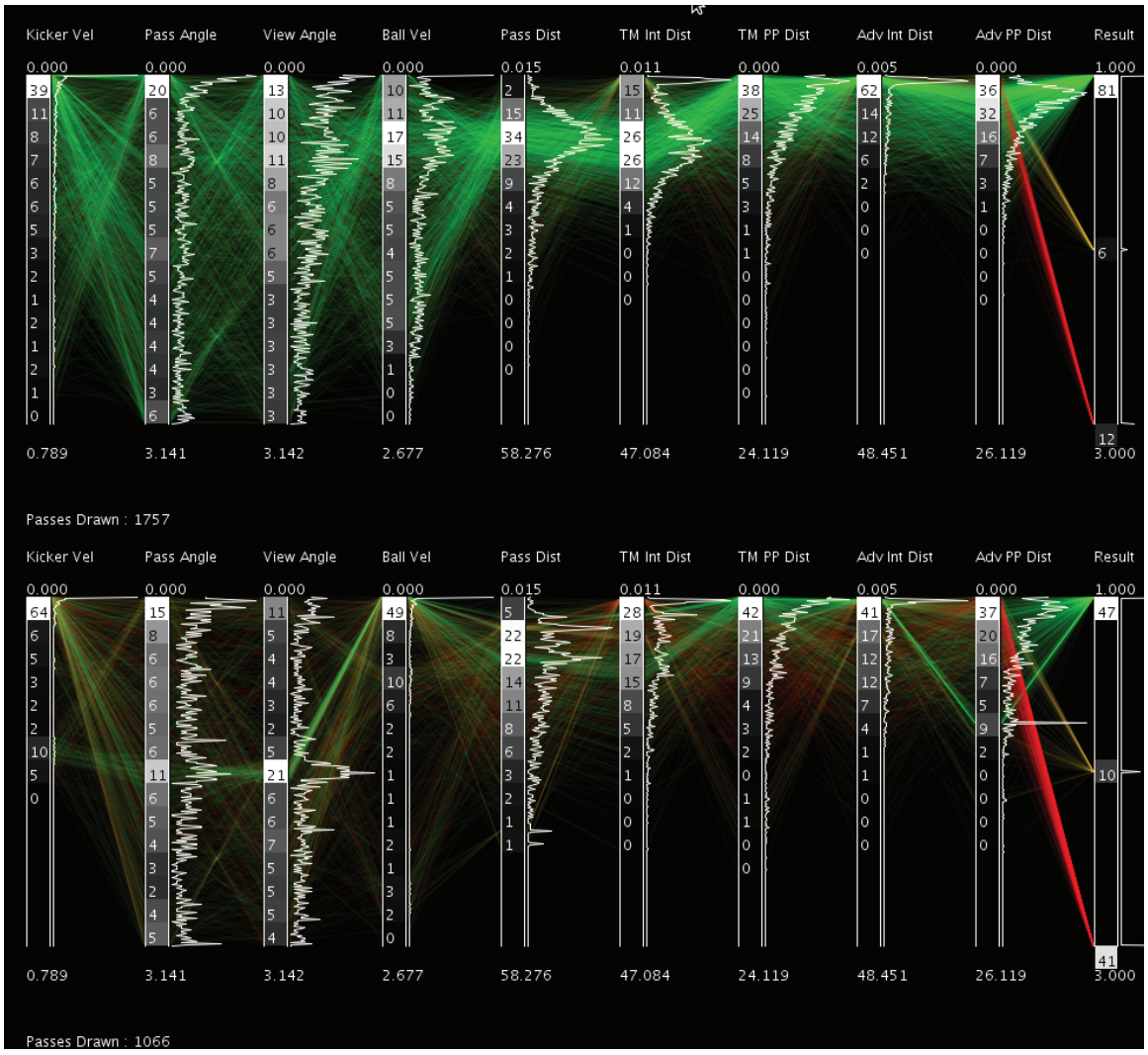

FIG. 4.6. Parallel coordinate view of best and worst team pass characteristics.

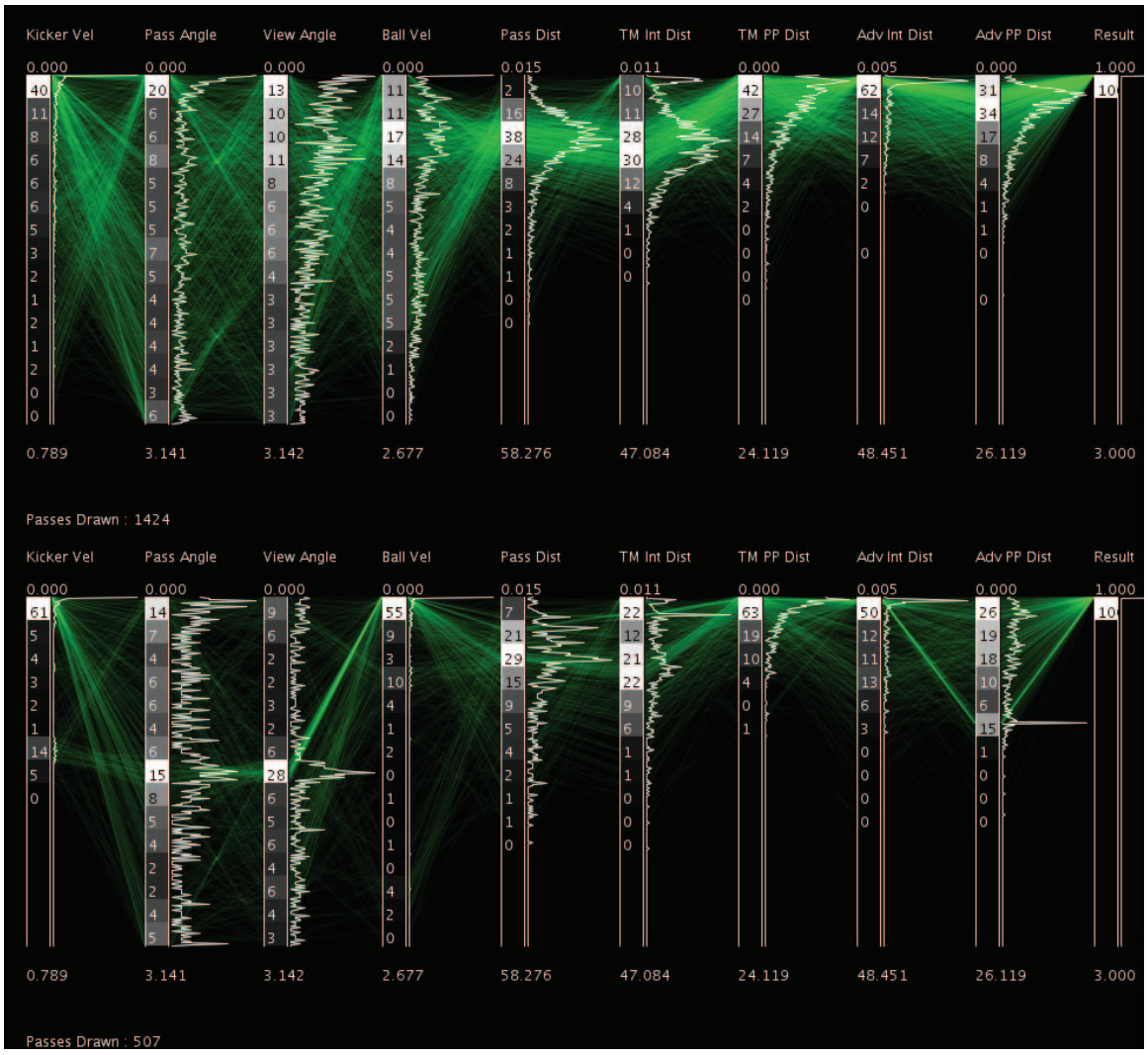

FIG. 4.7. Successful pass characteristics for best and worst team.

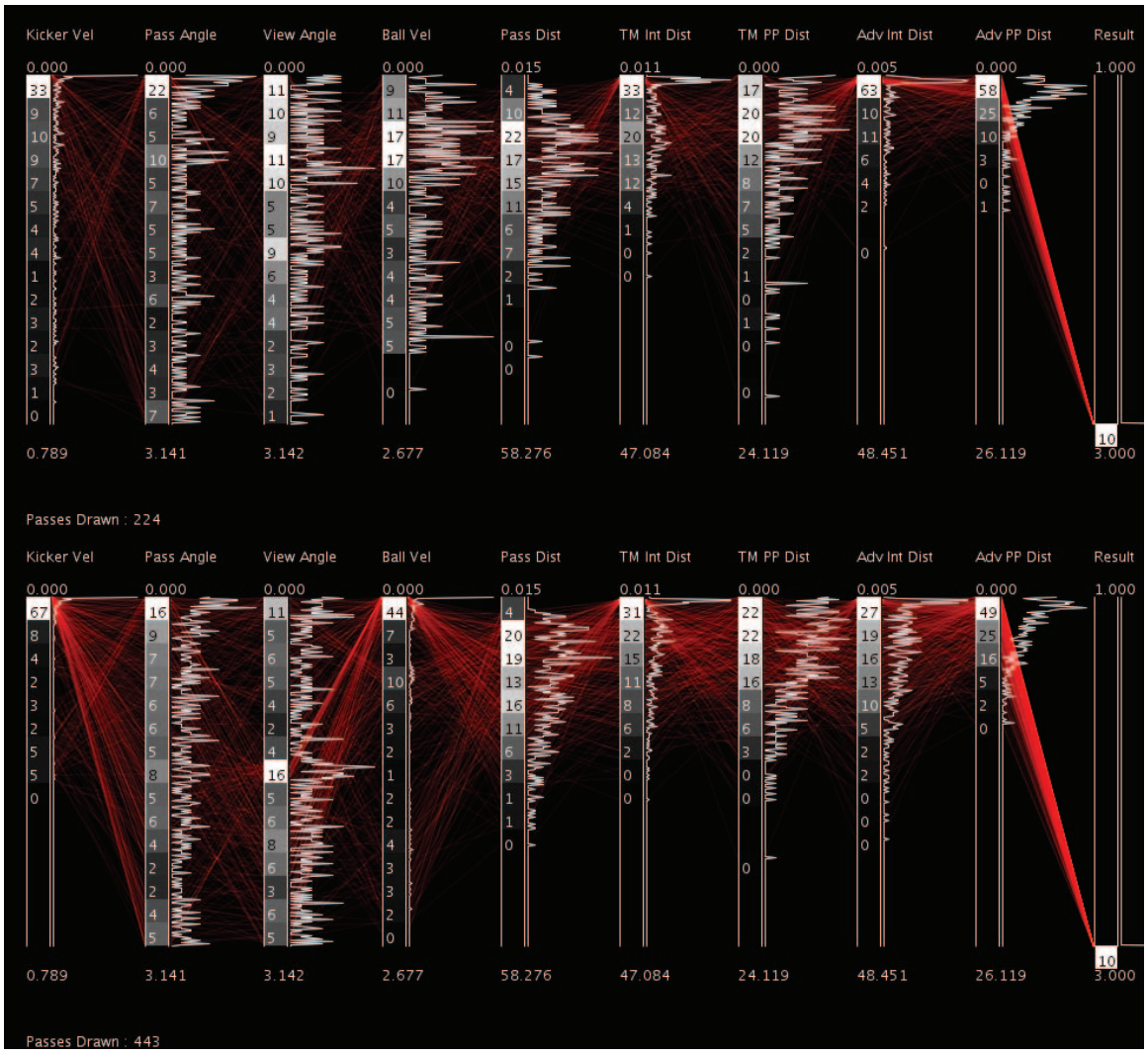

FIG. 4.8. Intercepted pass characteristics for best and worst team.

bottom of the figure. This may indicate a hard-coded value or a point where the heuristic A.I. algorithm needs to be examined in more detail. When these attributes are compared to the best team (top of figure), the superior game mechanics become obvious. For instance, the best team member's interception distance had a better distribution (i.e., longer passes) and the perpendicular distance was a little more drawn out indicating successful recovery of the ball even when not directly on the pass center line.

In the last figure, Figure 4.8, the lowest ranked team shows higher interception rates when the ball velocity was low (close or equal to zero) and the teammate's distance and adversary distance were very close to the pass center line. These attributes indicate that the team's heuristics allowed adversaries to close in on players and steal the ball and that the team's members were often bunched up. A more successful approach is to use successful passes to keep the adversary away from the player who has possession. This also requires that the teammates maintain distance between each other to keep the adversary on the defense. The highest-ranked team also exhibited this characteristic to some degree, with the exception that, unless the adversary was on top of the kicker, then the probability of an interception was significantly lower.

In terms of improving a team's performance, the worst ranked team needs to focus on improving the basic play mechanics for soccer. Higher-level strategies and goals are impossible to implement effectively without a solid game play implementation. In particular, the limitations on kicker velocity and the low ball velocity should both be examined in more detail. Adjusting the team's adaptability for these attributes would increase situations where passes can be successful, and, in turn, enable the team to more effectively move the ball down the field and to increase collaborative strategies among players. This is also true for the ninety-degree preference in passing. While these passes are successful, a team cannot be limited to one type of play especially when situations demand a multitude of options for success.

The suboptimal ordering of the parallel coordinate axes was another limitation that needed to be addressed. Ma and Hellerstein (1999) also addressed this limitation for parallel coordinates. When the axes are not optimized, a viewer cannot clearly distinguish which lines form trends. For example, the average closest adversary and closest adversary are difficult to correlate in the top of Figure 4.5 because they are separated by a single axis. However, when aligned together (as shown in the bottom of the figure), the correlation becomes more obvious—i.e., the two values are approximately the same across all of the samples. To optimize the ordering, a greedy strategy is implemented to place closely related axes together. First, the sum of the differences between every possible pair of coordinate axes is calculated for the displayed multi-variate samples. The higher the sum, the more likely that displayed lines would criss-cross the display, potentially obscuring similarly related values. Next, the axes are ordered to minimize the sums of these differences. Figure 4.5 depicts the before and after results for the filtered version of successful dribbles. While this provides an optimum solution based on attributes that have roughly equivalent values, another option would be to optimize based on principal component analysis or to minimize the number of edge crossings across the parallel coordinates display. Principal component analysis is an automated technique to determine which variables are related or dependent on one another. By applying this principle, portions of the parallel coordinate view that are confusing, unclear or that have too many crisscrossing lines would better lend themselves to analysis because related variables could be placed next to one another. Also, in place of the greedy strategy used, a dynamic programming approach would work exceedingly well at the expense of more computational time.

While these modifications provided additional analytical insights, it was still difficult to distinguish why one team performed better than another. Some of the suboptimal mechanics did stand out in the analysis, but identifying the overall deficiency was difficult. Additional fidelity was needed to separate each team so that they were visible at the same

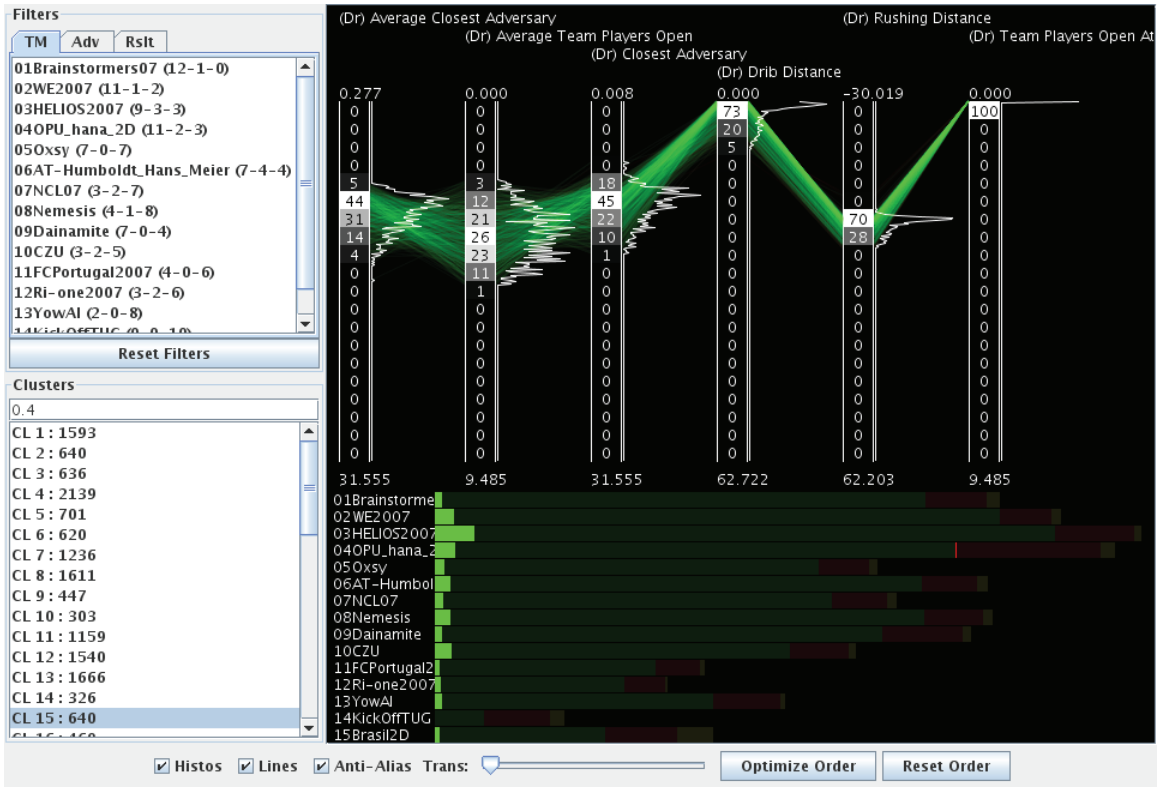

FIG. 4.9. Clustered samples within parallel coordinates visualization. To enable a better understanding of how the dribble attributes are partitioned, the parallel coordinates implementation automatically clusters them and shows each grouping interactively.

time. A complementary approach was chosen that mixed attribute clustering (Johansson *et al.* 2005) and attribute percentages among teams. The clustering approach was straightforward and based on the sum of the root mean square difference between normalized multivariate values. A resulting cluster was a group of attributes whose values were closely related and most likely to be represented similarly on the parallel coordinate display. To capture how each team related to the currently displayed attributes, a bar-graph based visualization below the parallel coordinates that shows the total number of attributes per team (broken down into the three color groups based on the success of the pass, dribble, or play, as described previously) was provided. To indicate which attributes are actually shown, a brighter color is dynamically calculated and displayed as the overall totals bar is dimmed. As with the interactive filter, going through the clusters provided a quick method for scanning for team-biased mechanics, successful mechanics, and failed implementations. Figure 4.9 shows an example of a dribble attribute cluster that was relatively successful across all of the teams, but especially for the third-ranked team. The highlighted bar charts denoting the number of team dribbles are in the lower half of the figure. For this cluster, the dribble distance was relatively low while the closest adversary distance was far from the ball. These types of dribbles were most likely to connect passes that kept the adversary off balance.

The advantage of this approach is to use machine automation to overcome visualization limitations. In this case, visualization limitations preclude a user from being able to extract the relevant features because there are too many lines that obscure potential patterns of interest. However, by applying machine automation to automatically group related attributes together combined with an interface that enables each grouping to be examined individually, a more tractable method to analyze the data is provided.

4.7 Traditional Visualization Results

The following paragraphs show how my methods enable a deeper understanding of success and strategy within the RoboCup tournament. Results from the three sections are given below (dribble analysis provided very few conclusions about team performance and has subsequently been omitted).

Match Summarization Results Match summarization provided a key method for quickly determining how a particular team played. From the visualization, the viewer can quickly determine the average complexity of collaborative plays, whether the game was one-sided or not, and how attributes such as stamina and distance to other players affected a player's decision making.

From the match summarization view, the overall flow of the match and specific plays that were of interest to the analysis were quickly apparent. More specifically, many of the lower-ranked teams had suboptimal implementations. This was apparent from the ball remaining on one side of the field throughout the match or the overall time of ball possession. Other indications of suboptimal performance became apparent when looking at the stamina attribute for each player. Teams with an effective strategy used stamina to accomplish specific plays whereas suboptimal teams tended to have a greedy strategy which burned all team member's stamina at once.

Figure 4.10 provides an example of this suboptimal strategy. The team on the right uses almost every player's stamina when they're on defense (stamina is recorded in the RoboCup tournament logs). This is shown by the white line that fades into gray as the stamina is consumed. However, the team on the left continues to score with a more efficient use of individual players' stamina. Also, the team on the right continues to keep the ball on the opposing team's side with only a few players. The top and bottom plays indicate that

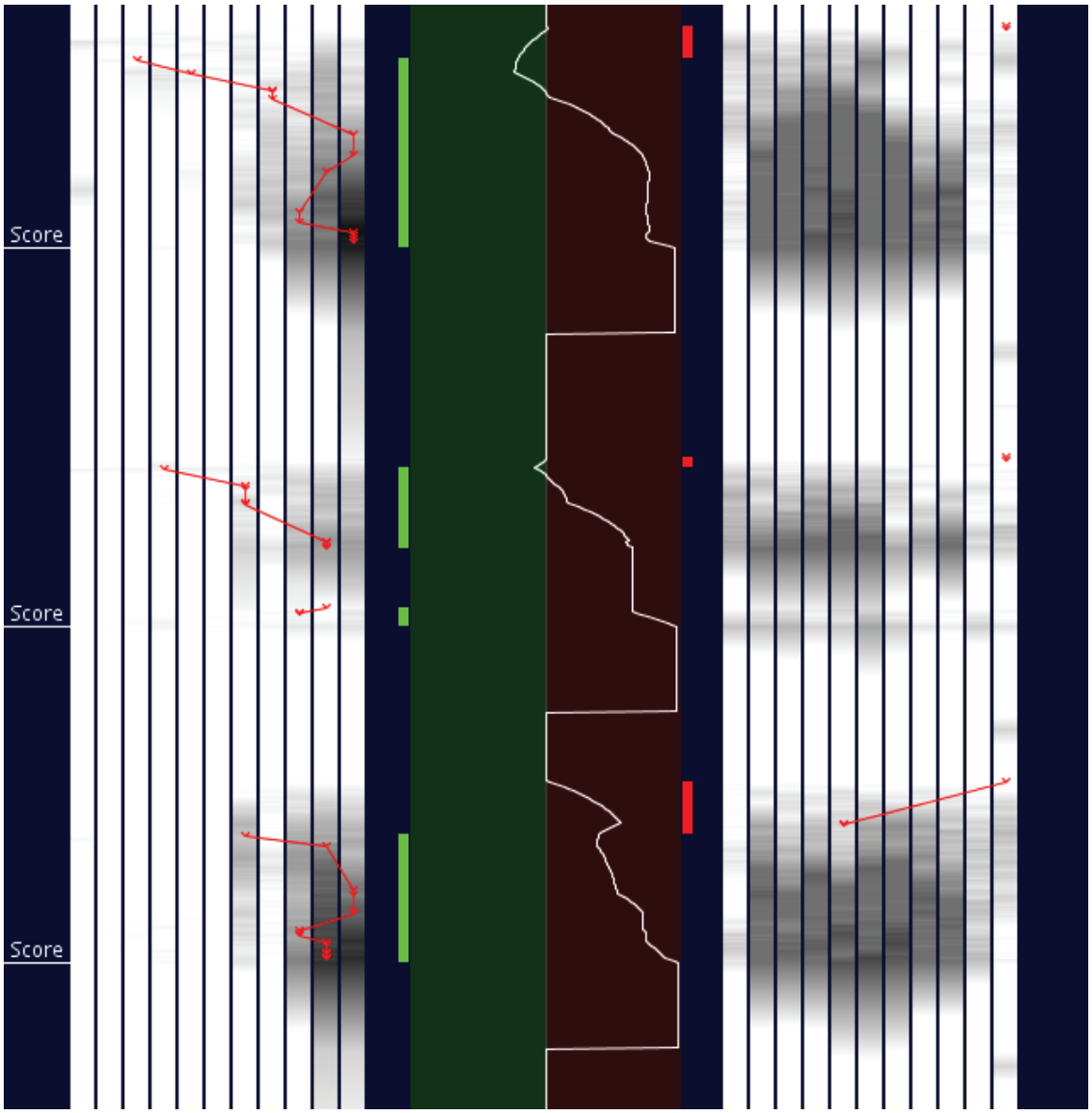

FIG. 4.10. Greedy stamina use. The team on the right side shows a stamina consumption pattern for almost all team members.

four players are used in the player's offense.

In addition to suboptimal strategies, other interesting patterns become apparent from the summarization view. Figure 4.11 depicts a team successfully maintaining control of the ball while trying to penetrate the opposing team's defenses. As one can see, the defenders are successfully blocking the team's attempts repeatedly. This is shown by the ball remaining in roughly the same position of the field. The offense is only using two players (i.e., only two players are active on the left-hand side of the diagram), and, while the overall drive lasts for many time increments, there are multiple out-of-bounds or penalties, breaking the drive into discrete parts (i.e., straight vertical ball lines that are not in the possession of either team). The success of the defense can be inferred by the defensive players (right-hand side) coming close to the ball (i.e., light gray and white bars), forcing the offensive to pass the ball, as shown by the passing lines and ball movement information.

When roughly equivalent teams are compared, the summarization agrees with the viewer's intuition. Figure 4.12 represents a match between the second- and third-ranked team in the competition. As expected, suboptimal strategies are avoided and, in general, each team has a combination of successes and failures over the course of the match. In addition to these quick turn-arounds between similar teams, long stints between scores were also observed when teams were similarly ranked. While the data to understand a team's learning capabilities was not available, a few of the teams did score more towards the end of the match potentially indicating adaptation throughout the match. The inverse was also true—some defenses appeared weak at the beginning of the game but improved as the team learned how to counter the opposing team.

While summarization provided a quick overview and enabled drill down to simulation events, it couldn't be used to analyze specific attributes of a team's performance. To look more at these values, a tailored parallel coordinate visualization was used to produce the following results.

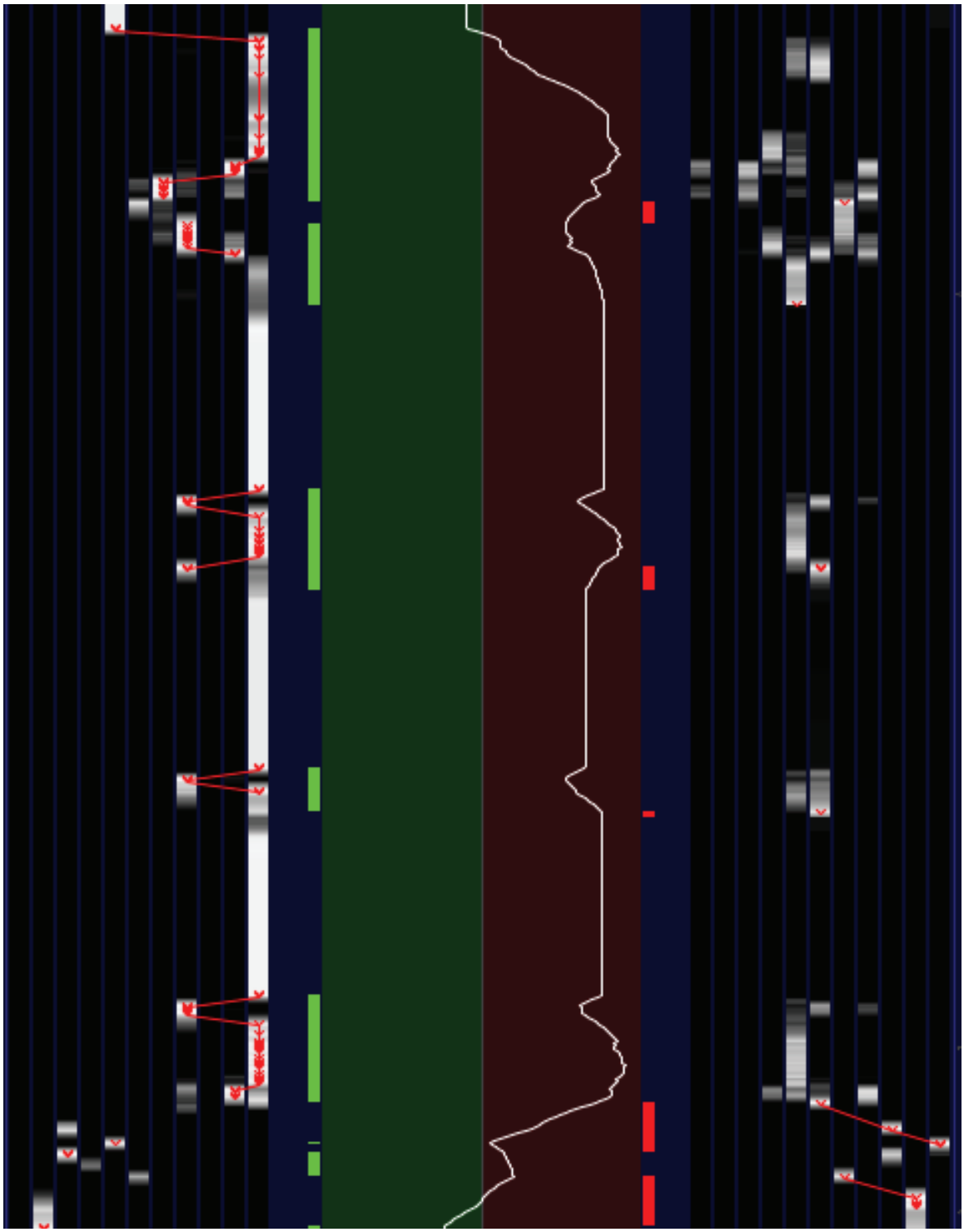

FIG. 4.11. Team attempting to penetrate defense.

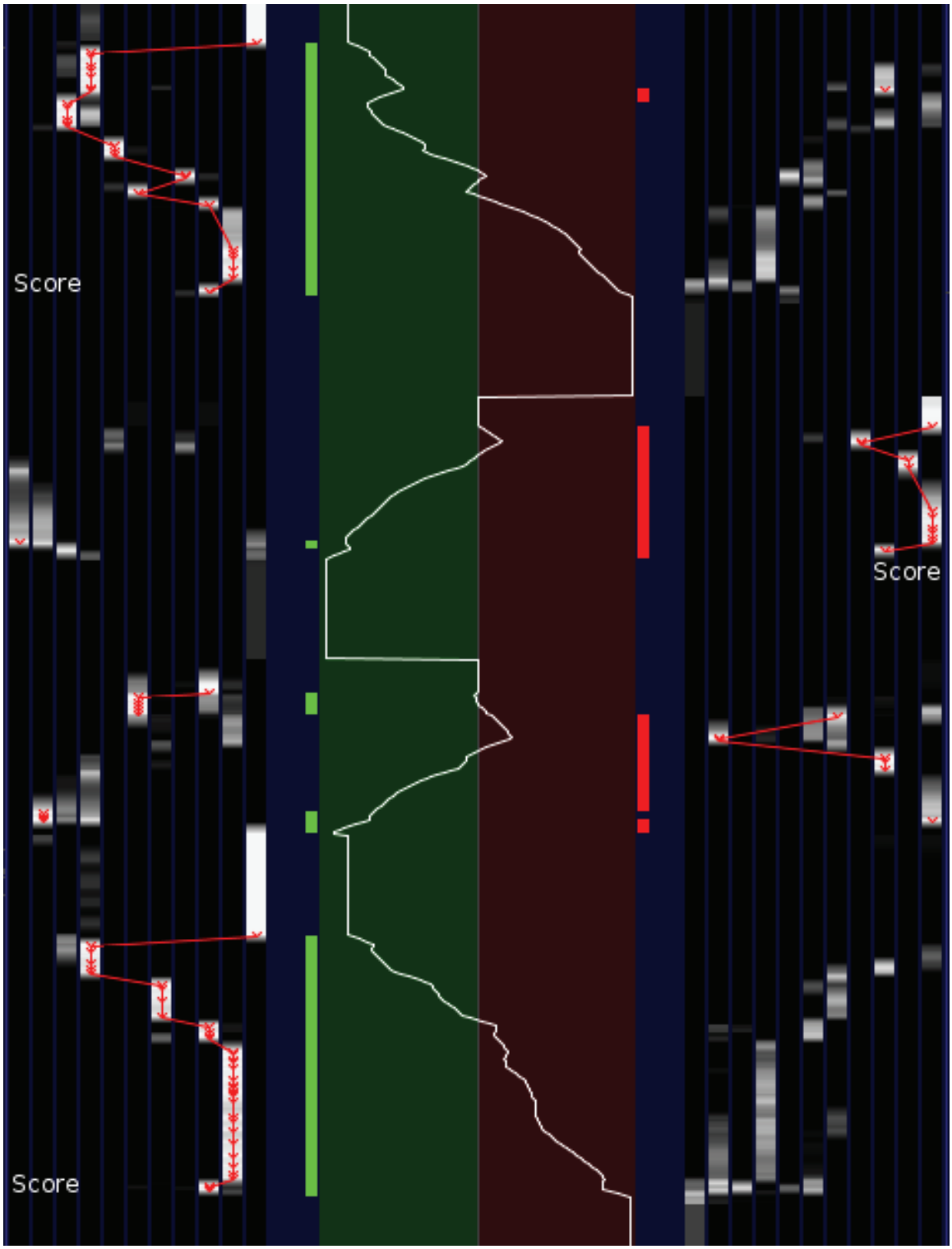

FIG. 4.12. Equivalent teams trading offensive drives.

Pass Clustering Results Passes were easier to analyze by examining the histograms than the relationships between each attribute. For example, there was a strong predisposition for lower-ranked teams to pass when the adversary perpendicular distance was at a distance of 9.4 units; this spike is visible across the sums of all pass attributes for the entire tournament. A spike also occurs within the dribble parameter for adversary closest distance at the same value so there may be a bias towards passing at that distance. There is a slight tendency across all of the teams to use the highest possible velocity for a pass, but the distribution itself is mostly uniform. Kicker velocity at the time of the pass kick is sharply at zero with an exponential fall-off. Teams preferred a straight pass, but some teams had a slight peak at the 90 degree mark. Within the RoboCup simulator, straight kicks (in the same direction as the player is facing) have no reduction in force, while off angle kicks receive less and less power. This explains why most teams favor the straight passes. Passes perpendicular to the player's direction (at 90 degrees) are probably lateral passes to other team members. Pass distance was in general quite short, but had a small normal distribution later.

There were loosely correlated pass attributes once clustering was performed that became apparent within the parallel coordinate display. However, the correlations for successful passes were relatively weak, indicating that the internal decision making and play mechanics must be relatively complex. In one clustered view of pass attributes (Figure 4.13), the top three teams had a relatively strong showing with a normal pass angle centered around 90 degrees (column sixth from the left). This was accompanied by a relatively small adversary interception distance (column one) and mostly centered on the kicker's viewing angle (column five), which most likely indicated lateral kicks from the player's motion. Other teams had the same attribute clusters but at roughly a third or less of the volume across the tournament (green bars at bottom of figure). Based on the increased use of these particular parameters by the top three teams, several conclusions can be inferred.

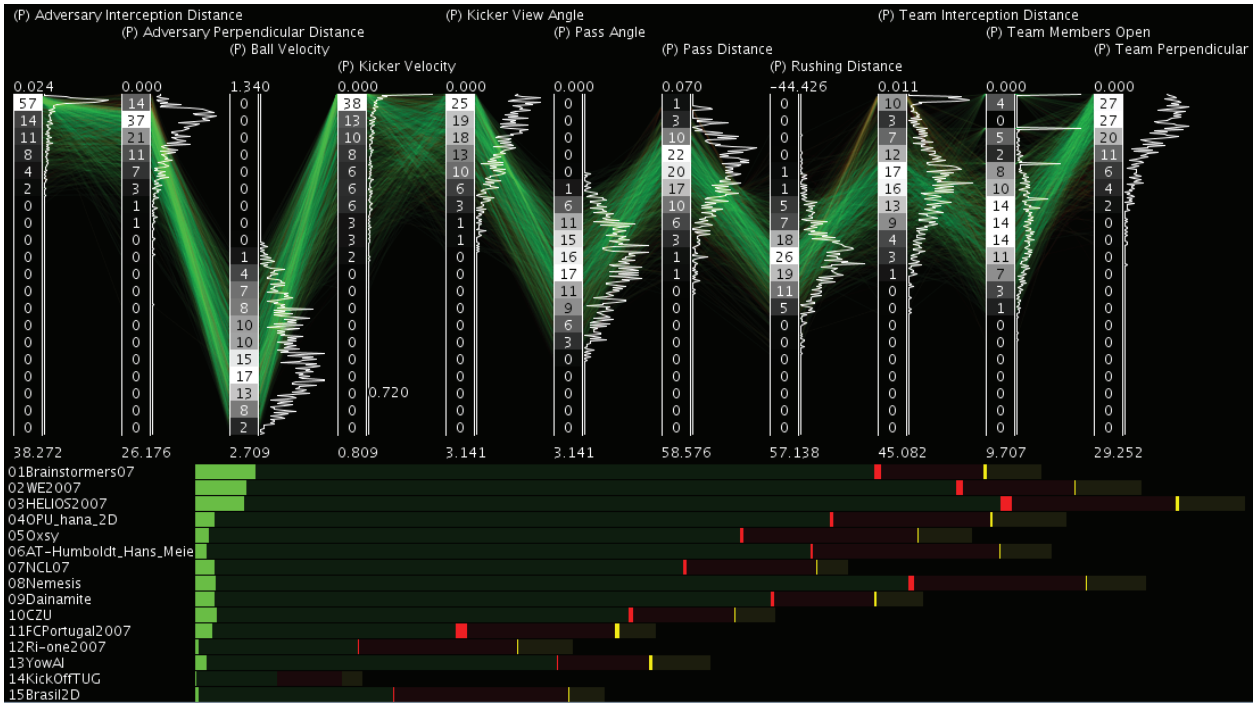

FIG. 4.13. Top three team successful passing strategy. This particular cluster of passes was favored by the top three ranked teams and represents lateral passes due to the angle from the player's direction.

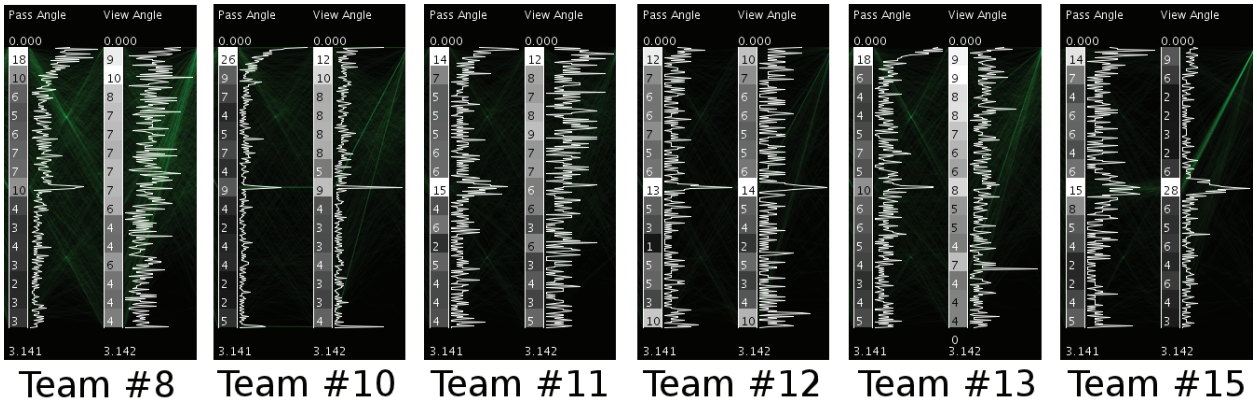

FIG. 4.14. Ninety-degree pass preferences.

First, these particular passing mechanics were mastered by the top teams. Second, the defensive strategy to prevent these passes was not implemented by the teams in the tournament. However, without additional details (either extracted parameters or internal team states), coming to a complete conclusion is difficult. The visualization techniques provide a method to go through the data and identify interesting trends—coming to conclusions for the particular anomaly requires additional analysis.

Other interesting trends become apparent when similar mechanics were found. For instance, the teams depicted in Figure 4.14 all showed a preference for ninety-degree angle kicks (first column in each of the tiled images in the figure). The teams ranked eighth, tenth, eleventh, twelfth, and fifteenth all had a spike at the ninety-degree angle from the kickers current direction. As all of these teams are in the bottom of the league, this passing preference indicates a weakness in the team's mechanics. For most of the higher ranked teams, the distribution was uniformly distributed after the initial normal distribution around zero degrees (recall that zero degree kicks have more power under RoboCup rules). It is distinctly possible that the lower ranked teams hard coded some of the behavioral mechanics into their artificially intelligence implementation—in this case, ninety-degree passes may appear to be more viable for lateral passes between team members moving

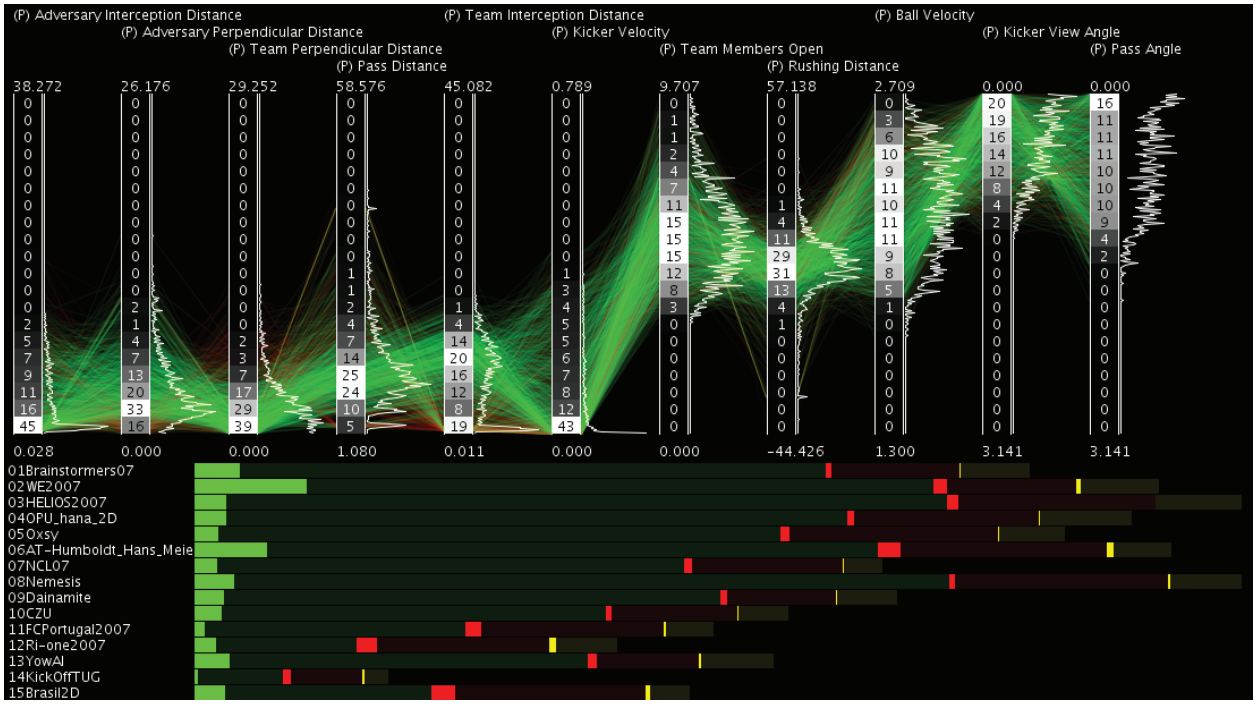

FIG. 4.15. Team #2 and #6 play mechanics. Note the higher percentage of this cluster's play mechanics for the second and sixth ranked teams.

in parallel to one another. However, as discussed earlier, flexibility across all of the pass angles open additional possibilities enabling more successful strategies.

Figure 4.15 shows that Teams #2 and #6 had an increased percentage of successful kicks with a relatively high pass velocity. As with other attributes, all of the teams shared this cluster except some of the lower-ranked teams; however, this wasn't necessarily consistent. The number of team members open for this cluster was also distinct in that it had a normal distribution. The open team member calculation is a gradient function measuring a player's openness to receiving a pass based on whether they were blocked by an adversary opponent and pro-rated by the distance from the kicker. The typical distribution for this attribute was very distinct (usually discrete bins at the 0, 1, and 2 player increments), indicating that these particular plays had a more complex field position than usual. Addi-

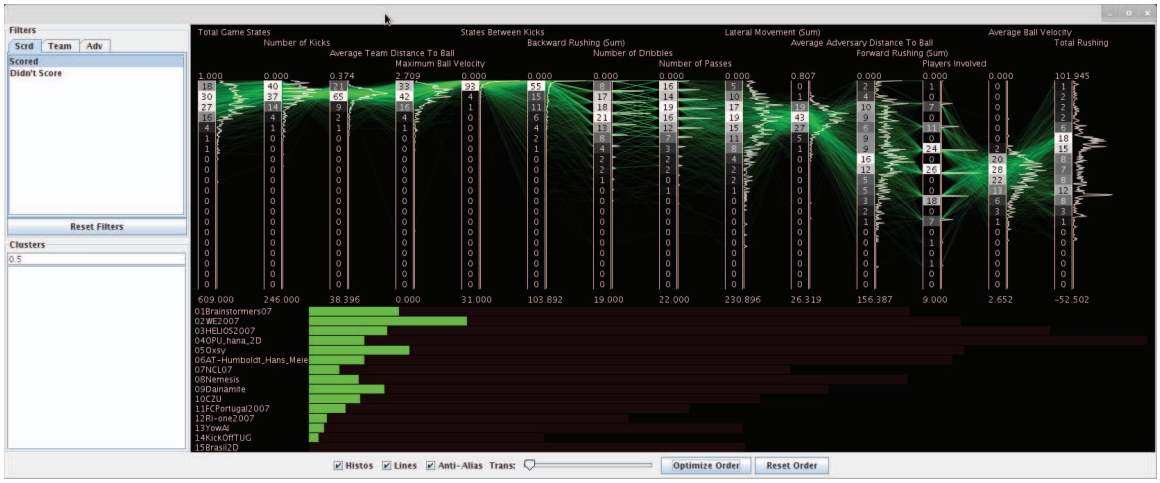

FIG. 4.16. Overall characteristics of successful plays.

tionally, these two teams may share a similar algorithmic implementation approach.

Play Clustering Results Analyzing plays proved to be a challenge due to the relatively small number of successful plays within the tournament data set. However, some general conclusions about successful plays can be made from the parallel coordinate view, as shown in Figure 4.16. The total game states (column one), number of dribbles (column seven), number of passes (column eight), average ball velocity (column 13), players involved (column 12), and average adversary distance to ball (column 10) had normal distributions. Average team distance to the ball was much sharper, with a peak at 2.65 units. Players involved peaked at three and four players, which corresponded well with the three passes as a peak. Total rushing distance had an initial spike at 17 units and another surge between 52 and 60 units. However, when comparing these relationships with failed plays (Figure 4.17), the trends are very similar. Many of the distributions were roughly the same but shifted. Players involved, number of kicks, and number of passes were in this category. Lateral movement had a large difference between successful and unsuccessful plays; unsuccessful plays had a sharp spike at zero for lateral movements while successful plays had

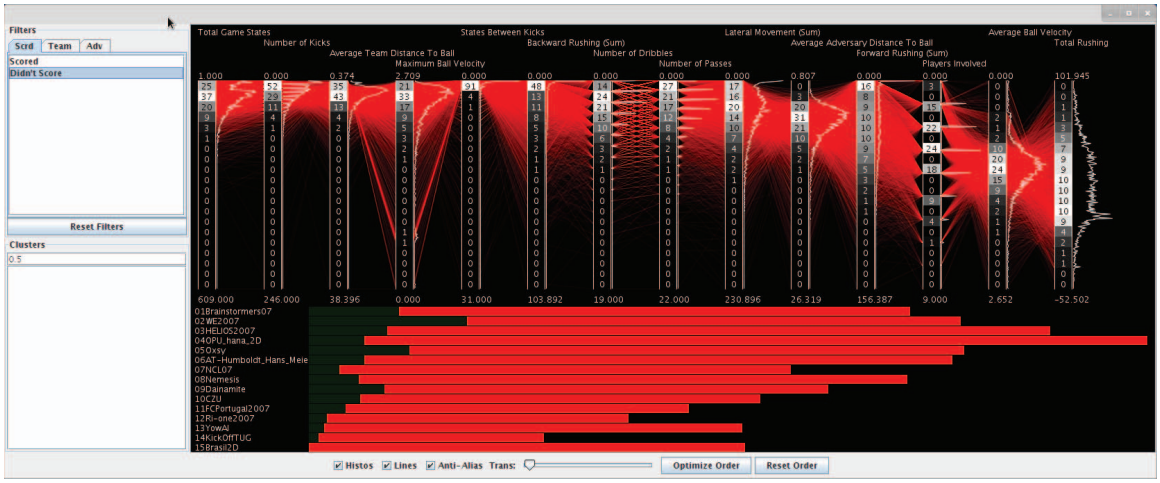

FIG. 4.17. Overall characteristics of failed plays.

a normal distribution. This indicates that straight approaches to the goal were unlikely to succeed; scoring approaches used lateral movements to bypass the opponent's defense. Curiously enough, average adversary distance to ball seemed to have little difference between successful and unsuccessful plays.

Clustering of the parallel coordinate attributes provided unique insight into differences between teams. Several clusters favored one or two particular teams, while other clusters coincided with the final rankings of the tournament. However, this wasn't always consistent, most likely indicating that different strategies or play styles weren't necessarily deficient, just different. For example, the higher ranked teams used more players, kicks, game time steps, etc. for plays, as would be expected. One unexpected anomaly occurred with the second-ranked team: they had a spike within a cluster that used fewer players during the play than the overall successful plays. Figure 4.18 shows this anomaly (red lines) along with a comparison of all of the successful plays (right-hand side of figure, green lines). Note the differences in the players-involved attribute among the spikes in the histogram and the larger bar at the bottom of the figure for Team #2. Team #1 had a spike within a cluster where backward rushing was significantly higher. This may indicate that

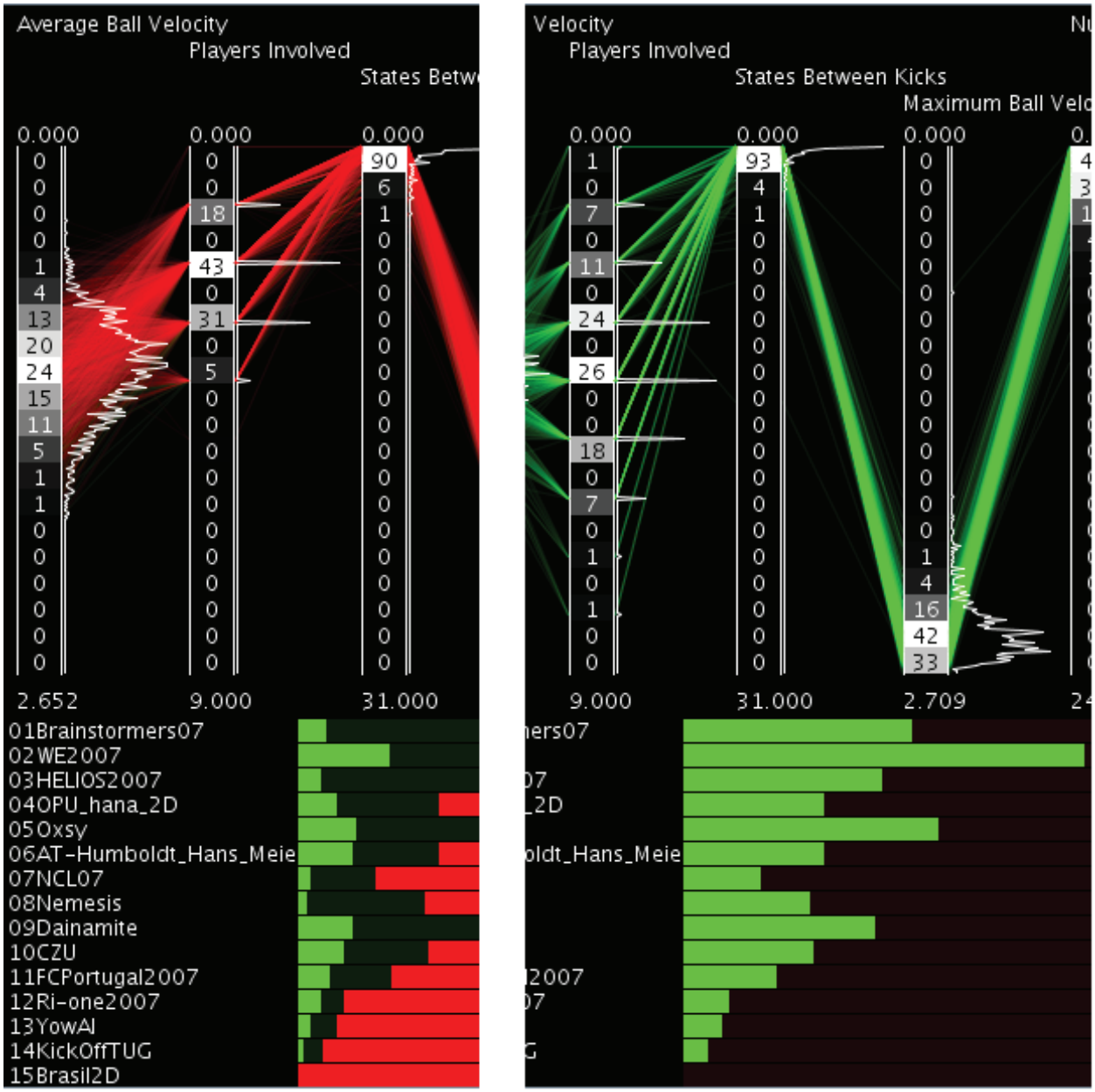

FIG. 4.18. Play cluster with fewer players involved. For reference, all of the successful plays are shown as a blow-out on the right-hand side of the figure. Note the increased percentage of these plays with the #2 ranked team (bottom of figure, left-hand side).

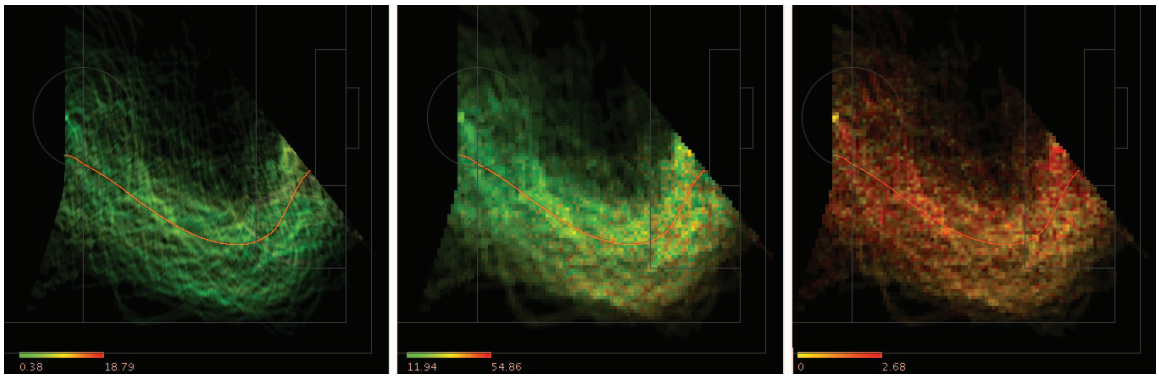

FIG. 4.19. RoboCup example #1. From left to right, the trajectory attributes rendered are minimum distance to offensive player (blend), average distance to defensive players (weave), and ball velocity (weave).

greedy or hill climbing strategies were not employed and that a different optimization was done.

4.8 Play Visualizations

While traditional visualization techniques derived new discoveries about the RoboCup dataset, combining the spatial ball trajectories with the application attributes can provide additional, and more intuitive, insight. Due to the number of trajectories in the tournament, automated clustering techniques are needed to group the data and prioritize analysis for the most common trajectories. In addition to prioritizing the dataset, the approach benefits when similar trajectories are visualized in composite.

The first RoboCup example in Figure 4.19 is the composition of 228 clustered trajectories showing an offensive drive around the lower half of the field. The three tiles represent, from left to right, the minimum distance to an offensive player, the average distance to the defensive players, and the ball velocity. For two of these renderings, three distinct colors were used (green, yellow, and red); the velocity image uses just two colors (yellow and red). The bright spot towards the center of the field shows the typical starting

point for most games (i.e., higher density trajectory data). The minimum distance to an offensive player (left-most tile) uses the blending color scheme. Note that for almost all of the play, the distance was relatively low across all of the trajectories. This is expected since the offensive player is continuously kicking and controlling the ball on its way to the goal. The only place this varies is towards the end of the trajectory, when the ball is kicked toward the goal. In this spatial location, the rendering shows discoloration towards both white (indicating variance) and yellow (showing increased distance).

The middle tile for the first example represents average distance to the defensive players. Towards the lower portion of the curve and the goal shot area, the colors transition to yellow, indicating less defensive coverage. This type of play is commonly chosen to circumvent the defenders in order to line up a goal shot, as shown. Recall that weaving the colors together shows actual sampled data values—interpreting the variance is up to the viewer. The right-most tile represents the ball velocity and is also color woven. Both the beginning and the goal shot area have higher overall velocities. This is expected of the goal shot, and, for the beginning of the trajectory, indicates a faster move as the offense pushes the ball into the other team's side of the field. The lower portion of the curve has a lower velocity, most likely indicating deliberation as the offense prepares to push towards the goal.

From the overall brightness of the composition, one can quickly determine the overall spatial structure of the cluster. In this case, the distance of the clustered trajectory from the average trajectory is relatively uniform. The cluster itself is also reasonably compact—i.e., the rendered trajectories do indeed belong in the same grouping. The beginning shows a slightly wider distribution, due to the variance in the starting position. The end, of course, is narrower because the goal area is smaller. The overall variance in the trajectory attribute values is also easy to gauge—in the blending example, the viewer can quickly pick out washed-out areas. In the woven examples, the differences between the adjacent tiles are

easily determined.

To compare the final composition with the straightforward, or raw data, approach described in the introduction, Figure 4.20 represents the same data (i.e., a specific cluster's ball velocity) as the right-most tile in Figure 4.19. The most striking difference is the starting ball location—a bright spot in the composition but relatively muted in the raw data. Similarly, the overall density, or cluster structure, is also difficult to gauge in the raw data approach—the viewer is not able to successfully assimilate the differences in the density since the high-frequency data (i.e., all the distinct lines) distracts the eye. The overall trends in the attribute value also tend to get lost in the raw data approach. While the viewer may gauge that the lower right side has a lower velocity (i.e., more yellow) in the raw data, the random, stray lines tend to overemphasize the attribute value around the edges of the "hair ball." However, the straightforward approach does provide a better representation for the final goal shots—this limitation in the approach will be discussed at the end of this section.

The second RoboCup example, Figure 4.21, highlights the composition's ability to show the structure of the cluster. In the rendering, the end of the trajectory has a distinct split. This split occurs as the offensive player shoots towards one of the corners of the goal, since the goalie is usually situated dead center to cover as much of the goal as possible. In addition to the cluster's structure, the visualization provides insight into the ball velocity attribute. At a glance, one can quickly determine variances in the ball velocity across the trajectories for most of the play (i.e., the somewhat washed-out appearance). Upon closer inspection, the initial velocity is relatively low and consistent (bright yellow) in the spatial area near the center of the field. As with the previous example, the final shot towards the goal has a higher velocity, at least in the lower shot direction. The variance in the upper shot direction would warrant more analysis to determine other contributing factors. Since this rendering only has 84 trajectories, the composition is somewhat coarse. Greater numbers of trajectories produce smoother images, since individual trajectory contributions

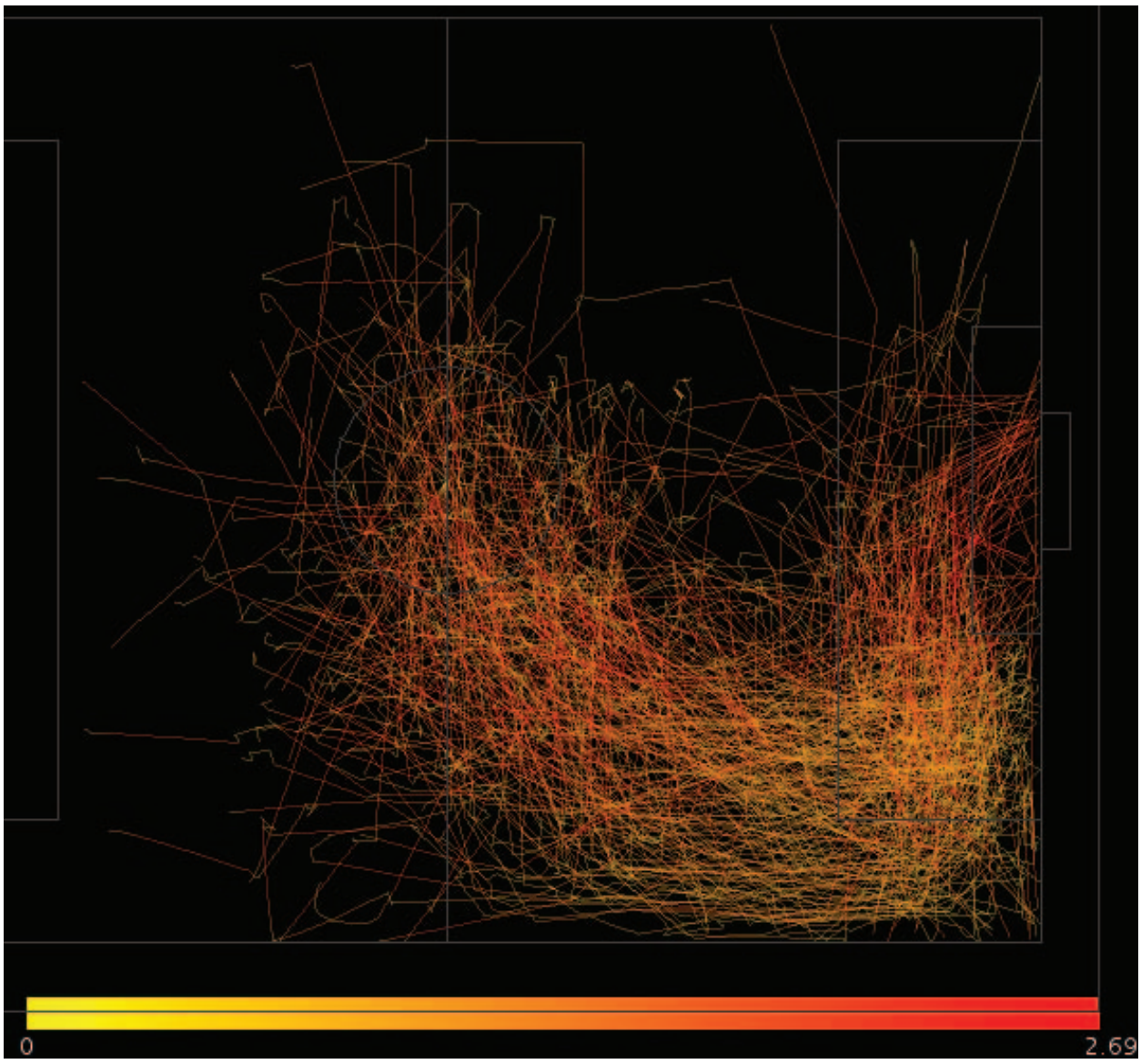

FIG. 4.20. Spaghetti plot. Multiple trajectories can be overlaid on top of a contextual background, but the image is difficult to analyze and understand.

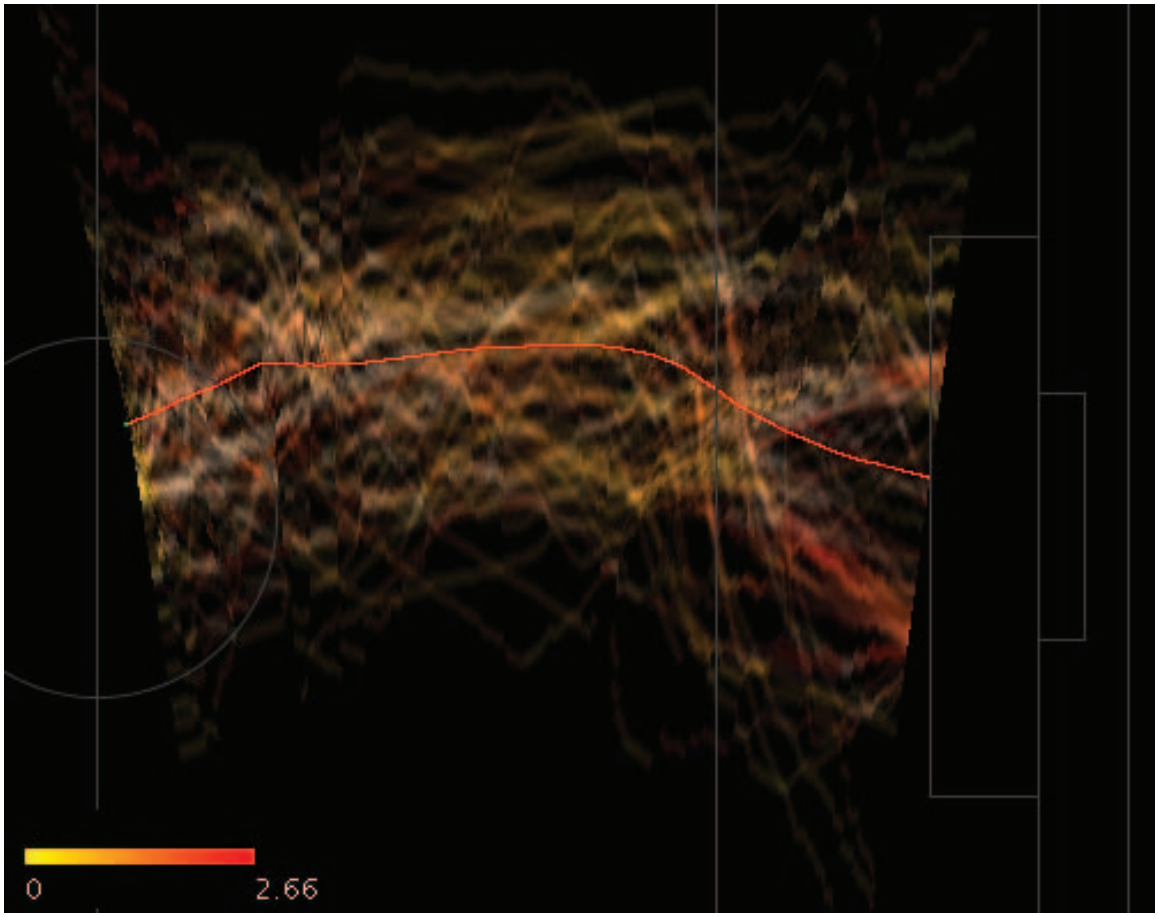

FIG. 4.21. RoboCup example #2 showing overall structure in cluster for the ball velocity (blend).

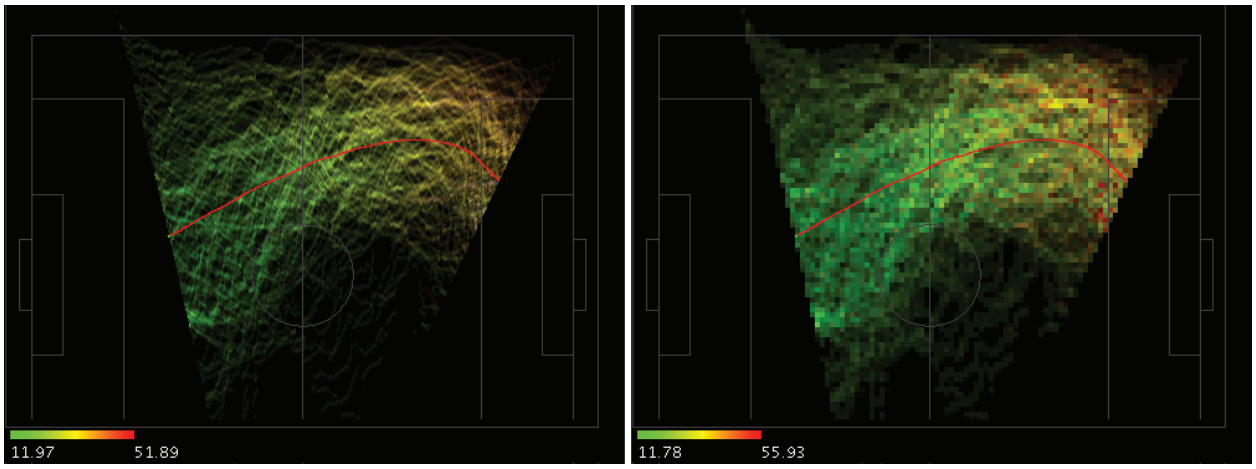

FIG. 4.22. RoboCup example #3 showing the difference between color weaving and blending for average distance to the offensive team members.

are diminished.

The final RoboCup example, Figure 4.22, shows the difference between using color weaving and color blending. Both tiles show the same attribute: average distance to the offensive players. The overall trend, as the ball moves from the offense side to the defense, is further and further away (i.e., green to yellow to red) from the offensive players. Most of the players on the offensive team remain in the back field in case the ball changes possession. The color weaving approach provides a stronger indication of variance, especially when the overall effect is in a large area, such as in the end of the trajectory. The color blending method forces the viewer to interpret the variance by identifying areas where the random sampling shows a variety of colors. However, the weaving model portrays exact attribute values from the data.

In general, weaving is more appropriate when the viewer needs to understand the exact data values of the underlying trajectory population. It is also more effective when there are fewer trajectories because sampling will not excessively over-represent the data. Weaving is somewhat easier to interpret because it does not require the viewer to discern

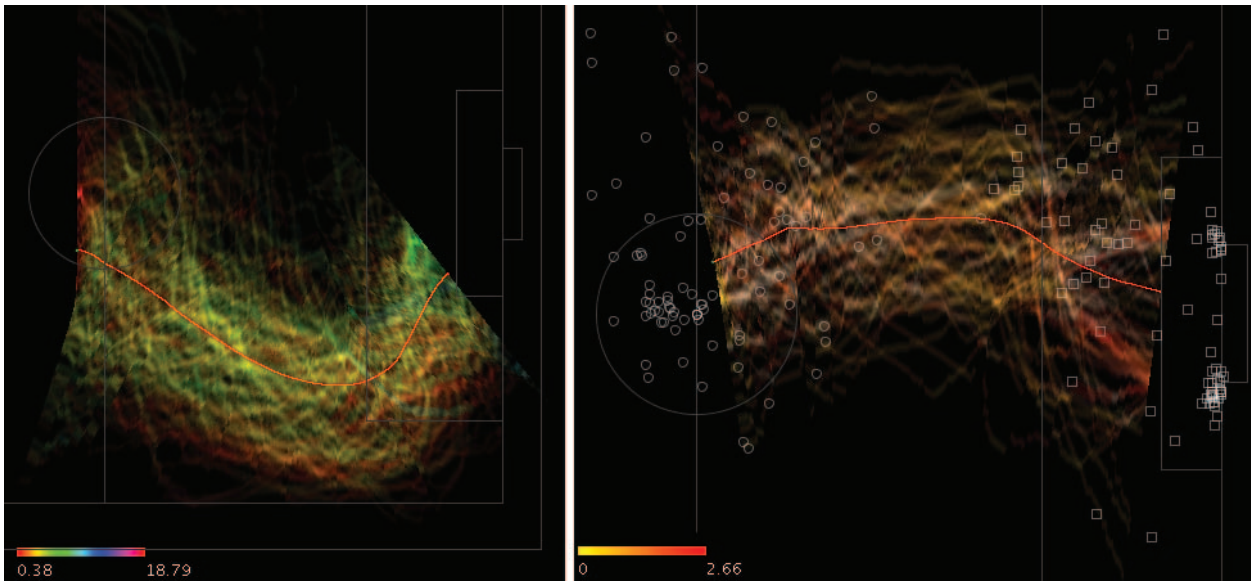

FIG. 4.23. Composition variations. The left tile expands the spectrum range to provide more fidelity. The right tile overlays the original trajectory end points to show their true locations.

the washed-out appearance produced by color blending. Weaving may also be used with a wider variety of color scales, since it is not constrained to just the hue component. However, larger groups of trajectories, which produce better statistical properties, benefit more from the blending approach, especially when paired with a larger color scale, as described at the end of this section. The blending approach can also use a higher resolution display, since the weaving is dependent on discernible tiles.

Several limitations are exposed by the final composition renderings. The first is that, since the average trajectory is used for the composition, the actual trajectory end points are obscured. For example, in the original data, many of the contributing trajectories extended all the way to the goal area in Figures 4.19, 4.21, and 4.22. However, the composition shows them as ending sooner. Similarly, the actual starting position in soccer, located in the dead center of the field, is misaligned by the transformation process: instead of appearing in the correct location, the point is slightly to the left in Figure 4.19. Two aids can be

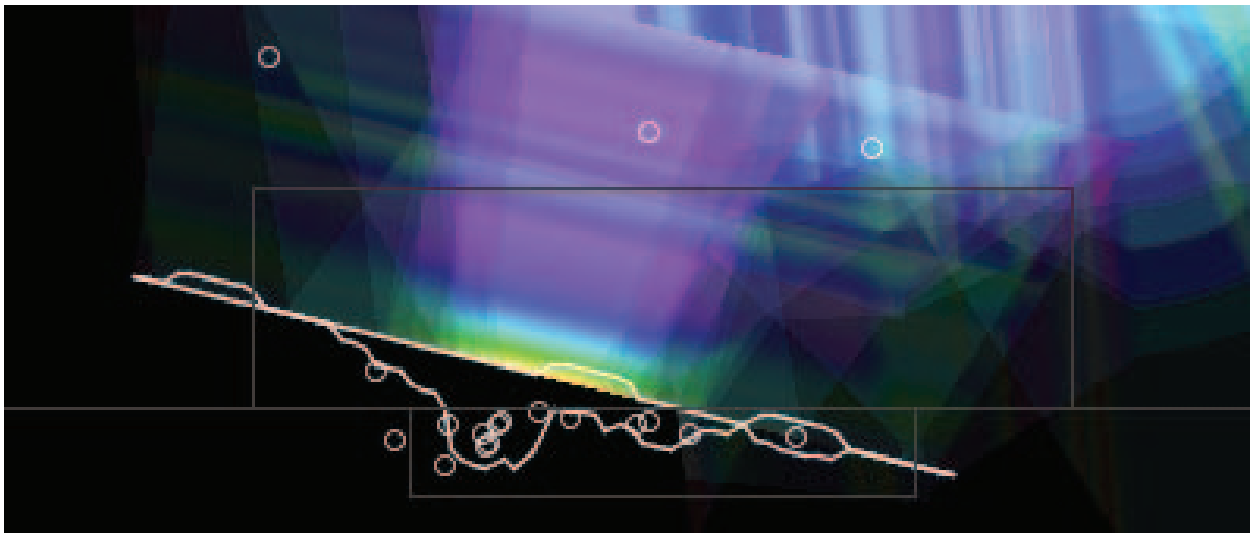

FIG. 4.24. End point histogram. A histogram based on the trajectory end points provides detailed information on the original starting and ending locations.

added to the visualization to show the actual data. The first is to overlay the actual end points on the composition, as shown in the right side of Figure 4.23. In the drawing, the trajectory start positions are shown as circles, and the ends are represented as rectangles. In order to show point concentration, the shapes are overlaid with a slightly transparent setting. The second aid is to add a histogram at each end of the trajectory composition, as shown in Figure 4.24. This histogram is created by accumulating the contributions of each end point. To form the histogram, each end point is projected onto the perpendicular line at the end of the average trajectory. The line is then "pulled" toward the end point at that location by using a self organizing map (Kohonen 1997). The necessity of adding the end point visualization is application-dependent—while it is relevant for the RoboCup data, it may not be required in other problem domains.

Another limitation, related exclusively to the color blending approach, is that it is difficult to use the complete dynamic range for the chosen color space. This is the result of single trajectories having the highest and lowest attribute values but almost no representa-

tion, since they occur in low-density regions. The highest and lowest colors do occur, but are almost completely faded out. One potential solution is to expand the color spectrum used for the visualization, as shown in the left side of Figure 4.23. Figure 4.23 is the same as Figure 4.19's left-most tile (i.e., the minimum distance to an offensive player). As one can see, the variations in the player distance are much clearer. While the complete spectrum is not as intuitive as the limited one chosen in the design, it does provide a higher dynamic range. Another potential solution is to remove the density contribution from the HSB mixing function and provide a separate, independent density view.

Chapter 5

APPLICATION: STUDENT COURSE HISTORY

When analyzed in aggregate, student course history data provides a deep understanding of the many factors influencing an individual student's performance. In addition to identifying student-dependent variables, the analysis can aid in recognizing deficits in existing academic programs and potential methods for improving the curriculum. In both cases, examining individual student information in isolation would not yield equivalent results—only when the data is analyzed in aggregate do general trends become apparent.

At the individual student level, analyzing trends in past student performance can aid in predicting future performance. For example, analyzing and correlating a student's high school GPA and SAT score with the course plan that they pursue can determine which factors play the most significant role. Alternatively, characterizing unsuccessful student strategies can identify existing students on the same downward trajectory. These students can then be counseled towards more productive strategies, resulting in a successful outcome for both the student and the academic institution.

Whereas improving the individual student's academic career is a local optimization, the conclusions from analyzing student data can also help to refine and shape the curriculum, leading to global optimizations, the focus of this application. A curriculum is composed of a recommended course plan including prerequisites and a course-to-topic

coverage model that ensures graduates are well rounded. By determining dependencies among mandatory prerequisites and recommended, or even potentially unrelated, coursework, academic program developers can improve the existing curriculum. Improvements may include requiring new prerequisites, modifying the topic coverage for specific courses, or determining threshold grades for students to progress to more advanced courses. By enacting global optimizations, an academic institute can increase retention and graduation rates, leading to a more successful program for both students and the institution. Course history data can also help determine new policies and procedures. For example, the effect that course load has on student performance could be used to determine new restrictions on the maximum number of courses that a student could take per semester.

From a technical perspective, this application domain was chosen due to the complexity for analysis and its applicability to the approach. While generic visualization solutions exist for many similar types of time-series data types, most notably xy-scatter plots and trend lines for scalar values over time, solutions for set-based elements are rare and have not been adequately addressed. Student course history falls into this category and is difficult to analyze in aggregate. However, when the approach is applied, analytical results can yield significant insights into student performance and the overall curriculum.

5.1 Application

To formalize the application notation, the following definitions are used for the *student course history* domain. For discrete time durations, referred to as *semesters*, a student attends multiple courses. Over the course of a student's academic career, multiple semesters are completed with increased expectations as the student progresses towards a degree. *Courses* have intrinsic characteristics, such as *academic discipline* (e.g., history or mathematics), *prerequisites*, and a level designation denoted by a *course number*. When as-

sociated with a student, extrinsic *attributes* become associated with a course—final grade, semester taken, instructor, etc. Similarly, students have intrinsic associations, referred to as demographic data, and attributes that change over time, most notably accumulated GPA. This model can be generalized for other applications. For example, dietary history for a weight loss program or daily personal spending binned by category could both be modeled similarly.

Analyzing this type of data is challenging due to the concurrent nature of course work in the student history. For a slice in time, a student attends multiple courses, breaking the basic definition for a function (i.e., a single time value maps to multiple values) and eliminating most existing temporal visualization approaches. The problem is further complicated by both the intrinsic and extrinsic characteristics of the courses and students. However, if the temporal and concurrent aspects could be accurately represented, analytical conclusions derived from the aggregate student trends could then be leveraged to improve academic programs.

The approach leverages the previously described composition techniques to depict trends in student course history data. First, students are grouped by the grades that they receive for specific courses—most notably, the gateway and mathematics courses. *Gateway courses* expose starting students to the core concepts of the major and determine a student's aptitude for the coursework. After the groupings are created, the visual composition techniques are applied to identify overall performance, trends across time, and aggregate semester grades through a *small multiples* (Tufte 1990) approach. In addition to examining course-grade dependencies, students are also grouped by demographic categories to identify related correlations and develop a clustering metric to organize the students based on their academic career similarities. The results successfully visualize expected trends and discover unexpected relationships in student performance.

In summary, the approach needs to reveal aggregate student-level performance and

curriculum-driven features. Such a technique would then address the following questions:

- Which courses are most critical for later success?
- Are there places in the curriculum where students struggle to progress smoothly?
- Do particular demographic groups face distinct challenges at particular points in the curriculum?
- Which demographic features play the most prominent role in student success?

5.2 Related Work

The most similar projects to this research are visualizations applied to similar data sets (e.g., student course histories). Traditional techniques were the most prevalent, and the analytical goals (i.e., improving student performance) directly aligned with the goals and objectives. Wortman et al. (2007) developed an application to interactively explore the first three courses in computer science using traditional visualization techniques. Edwards et al. (2006) provided students with feedback based on data collected from electronic submission systems. The data was visualized with traditional visualization techniques and gave students feedback into their progress relative to their classmates. CourseViz (Mazza & Dimitrova 2004) visualized web-based data to identify trends in behavior and identify remote students in need. DynMap (Rueda *et al.* 2005) visualized the evolution of computer security students through particular subjects using concept maps. Two studies were performed to confirm that the characteristics were appropriate and understandable and that DynMap visualization system was usable for analyzing student data.

While all of these efforts gave additional insight into student performance, most of them relied on traditional visualization techniques that only focused on a narrow window in time. A new approach is needed that examines students throughout their entire academic

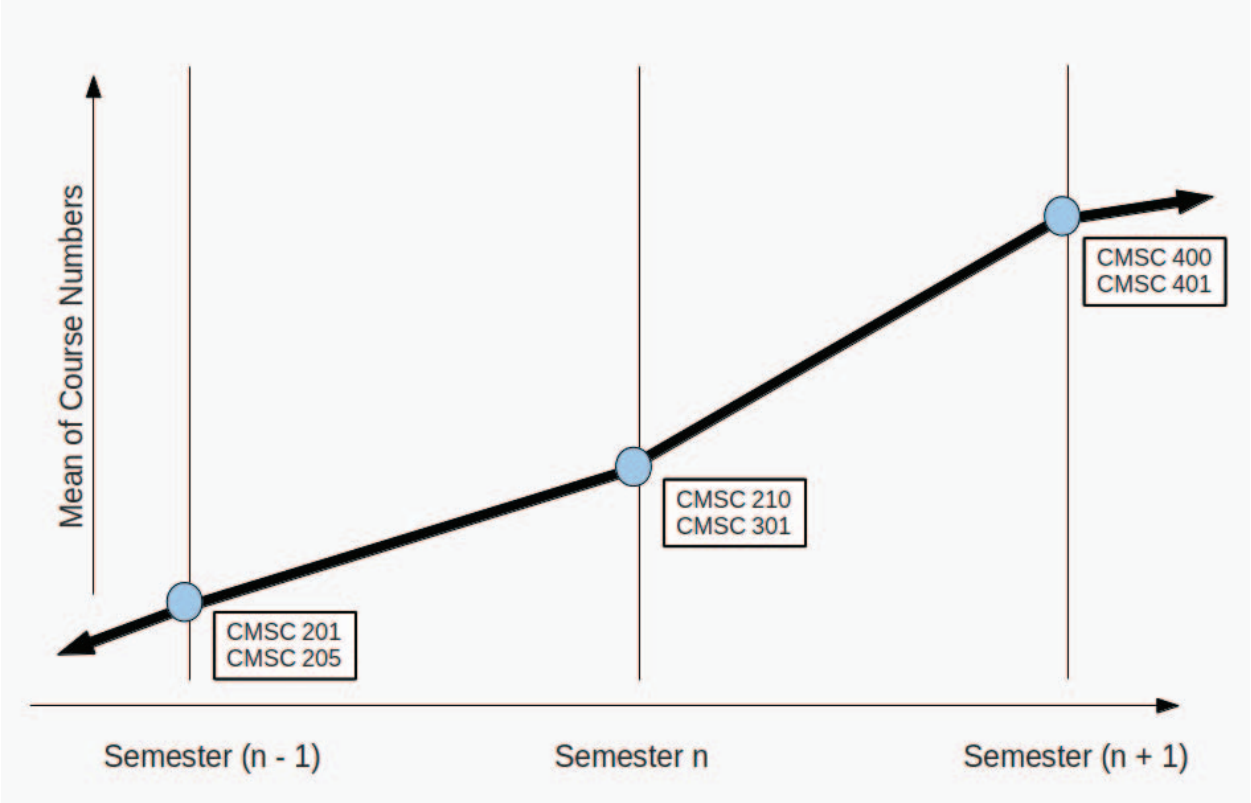

FIG. 5.1. Two-dimensional representation for student course history. The x-axis corresponds to the semester index while the y-axis represents the mean of the course numbers for a semester.

program. By exposing long-term temporal trends, subtle variations and patterns become more apparent. Furthermore, new visualization techniques are necessary, because, while traditional techniques help to understand summarized aspects of the underlying data, new approaches can expose time-based dependencies.

5.3 Approach

Student history analysis is used to determine which individual performance factors or demographics significantly influence a student's overall performance. By aggregating students according to their performance in specific courses, a controlled scientific experi-

ment is approximated. Specifically, students receiving a specific grade in a specific course become an experimental group while the rest of the student population serves as a control group. By further refining the groups to cover all possible grades for all possible courses, critical points in a student's career become evident. Furthermore, the relative performance of that student in that particular course reveals the overall importance of the concepts in the course as they relate to a student's overall performance. Visualization techniques are used to graphically depict the student's academic history, as well as to show variability in semester GPAs and course advancement. An alternative approach would be to use statistical methods to score overall student performance after the initial course-grade groupings were created. While the statistical method would lead to a faster determination for the most critical course-grade pairs, the composite visualization identifies fine-grained variations in semester grades and deviations among the student groupings across time. These discoveries could then be leveraged to create statistical processes and automated analytics to understand each aspect in detail.

5.3.1 Student Trajectory Representation

To successfully use the composition process, each student's course history is represented as a two-dimensional trajectory. Trajectory representations are complicated by the fact that a single semester contains multiple courses. Discrete values for each semester are, therefore, difficult to create. If each course is individually examined, comparisons among different students who may only partially overlap in a semester are difficult to make—for instance, what ordering would be used when only some courses overlap? To solve this problem, the overall average of the course numbers is used to represent the y-axis value for a semester. For the provided application dataset, most students incrementally take higher numbered courses over their academic career, especially within their declared major. To maintain the representation's temporal characteristics, the x-axis is used to represent the

student's semester index. Figure 5.1 provides a graphical depiction of the spatial trajectory for a simplified student course history. In the figure, the student's trajectory slopes upward from left-to-right as higher-level courses are taken. Since the average of the course numbers is used, the resulting y-values for the depicted semesters would be 203, 255.5, and 400.5, respectively.

The most critical assumption for the spatial representation is that increasing course numbers correlate strongly with student progression and advancement. This assumption holds true for this dataset—a careful examination of the recommended curriculum shows that the course numbers increase monotonically across a student's career. However, for universities where this is not the case, a different metric would need to be developed corresponding to knowledge increments. For example, the aggregate number of prerequisites could be used to score each course with a numerical value. Alternatively, the core concepts for the major could be mapped to each course in the curriculum resulting in a fine grained metric. In our results, the areas of higher divergence tended to occur at the end of student careers due, in part, to specialization at the 400-level series. However, for the introductory math, science, and gateway courses through the mid-level courses, the data correlated well across the groupings.

Courses not directly related to a student's major were removed from the analysis. For example, courses in English, social studies, history, or economics had little to do with a student's progression in their science or engineering degree. Furthermore, these classes did not monotonically increase in number and were often taken at various points in a student's career. In order to keep the spatial representation clear, only courses directly related to the curriculum—i.e., computer science, computer engineering, mathematics, and physical sciences—were used.

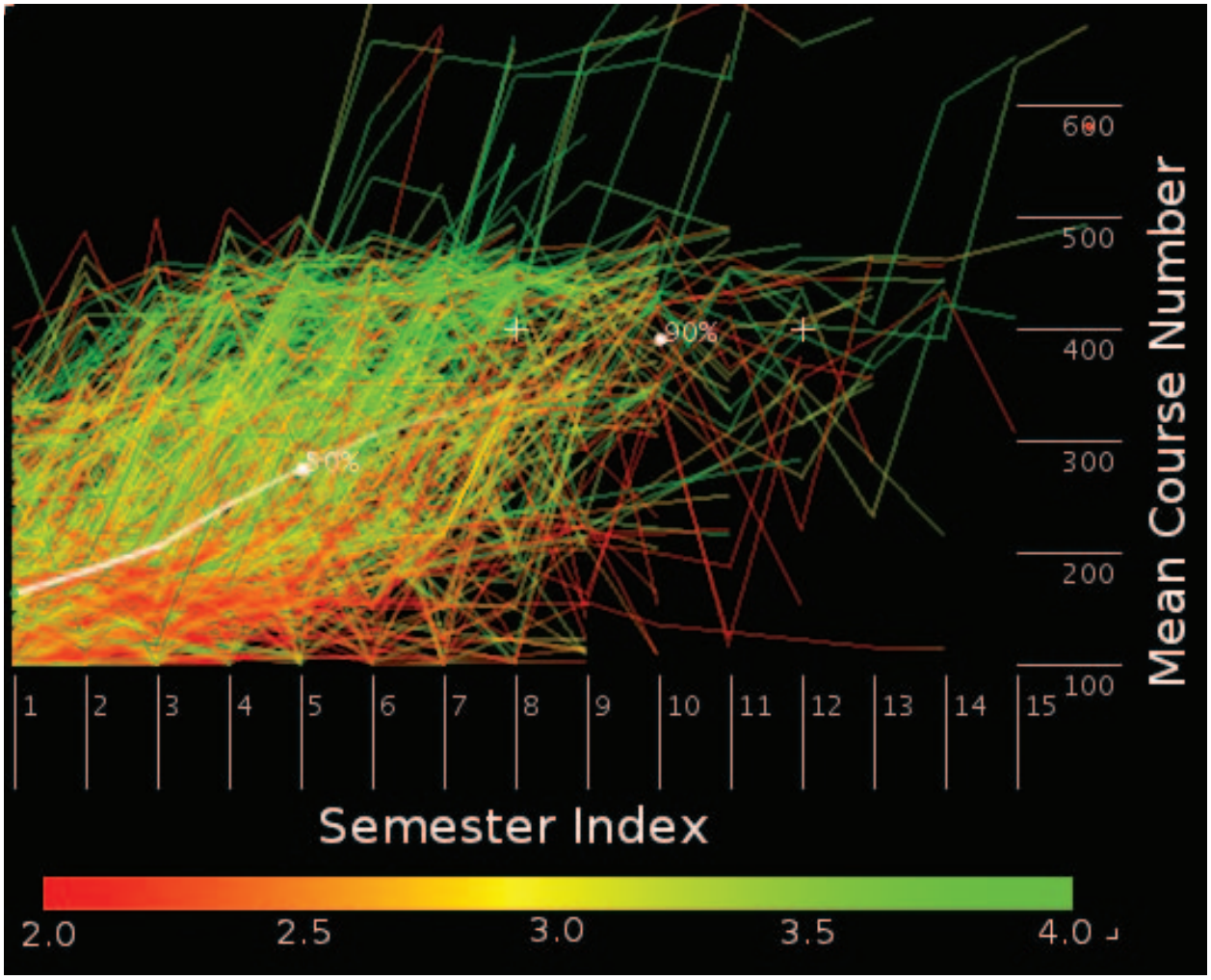

FIG. 5.2. Two-dimensional representation of historical student course data. Color corresponds to semester GPA.

5.3.2 Structural Composition Process

Unfortunately, attempting to visualize the trends and characteristics of many students at once is difficult even when using a simple representation, as described in the last section. Figure 5.2 represents all of the computer science students in the sample and is similar to Figure 4.20 in the first application. The figure uses the previously described two-dimensional representation and is rendered by coloring each line segment with the semester GPA. In the non-composited figure, it is possible to tell that students with flat trajectories tended to have lower grades—i.e., students who remained in low-numbered courses had red and yellow trajectories. However, meaningful comparisons are difficult at best.

Unless otherwise noted, a green-yellow-red color scale will be used for GPA on the 4.0 to 2.0 bounded interval. GPAs below 2.0 will map to red. These colors provide an intuitive mapping (Brewer 1999)—students doing exceptionally well are shown as green while those students clearly in trouble are red. Yellow as the intermediary provides an expected transition between red and green. It should be noted that this color scheme is problematic for color blind individuals due to the use of red and green. However, for this application, the use of "stop-light chart" colors provides an extremely intuitive representation to understanding the data—in accordance with Silva et al. (2007), "...the most striking features of the image reflect the most important features of the data." Furthermore, the approach is flexible and can use other color schemes to represent the trends in the student data—in the results section, an alternate colorscale is used to depict the data.

Figure 5.2 approximates a traditional line graph and, to some degree, a parallel coordinates view with several notable distinctions. First, the axes do not represent different dimensions but instead correspond to discrete times (i.e., more similar to a line graph). Second, each element in the chart (i.e., a single student) does not span every coordinate axis—instead, the student's trajectory ends when they either graduate or otherwise end

their academic careers. Density is the most observable characteristic of the straightforward approach; however, this comes at the expense of multiple course history lines overwriting one another. While opacity could be used to preserve the GPA values, color blending would result in a poor depiction of the results since only the colors are merged, not the underlying data values. Furthermore, local deviations are not represented in a meaningful way.

The trajectory composition structure needs to (1) show the overall *spatial characteristics* of the trajectories (e.g., course number progression) and (2) clearly denote the trend in the *student-related attribute* (e.g., semester GPA). *Spatial characteristics* enable the viewer to comprehend the general shape, density, compactness, and spatial variances of the trajectory set. For example, spatial variances along the vertical axis indicate differences in the course level for that semester. Furthermore, spatial density corresponds to the distribution of trajectories—e.g., tightly coupled together or generally spread out. Likewise, *student-related attributes* enable the viewer to understand how a trajectory-related attribute changes over time. For the application, the requirement is to depict both the mean and variance for the semester GPA.

In contrast to using a length-based parametric representation for the spatial trajectory (i.e., as in the first use case), a fixed time interval is used instead—i.e., parameterization is based on discrete time bins aligned with the semesters. This method more closely satisfied the need to correlate students in the same semester with one another. Earlier attempts to use the length-based approach resulted in compositions that slowly grew larger across time due to accumulated error. However, by using the fixed time interval, the error was negated because each semester re-aligned the students to one another.

5.3.3 Color Rendering

For this specific application, density should indicate the most common student trajectory within the grouping or demographic slice, and color should convey information about

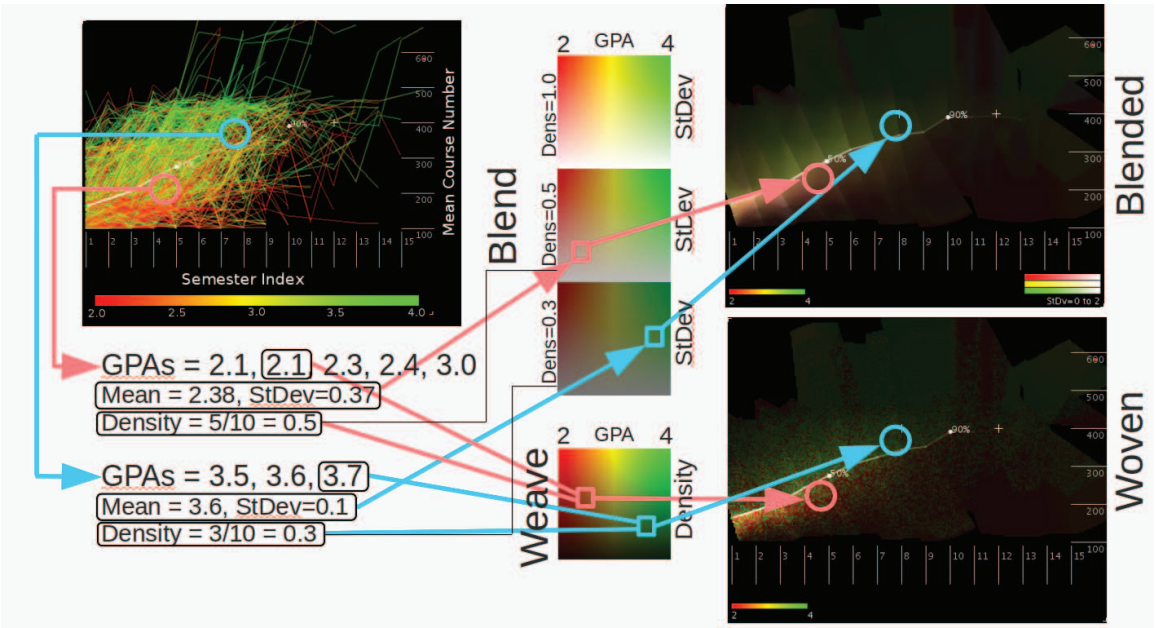

FIG. 5.3. Color composition. From the trajectory data (upper left), the associated GPAs are listed for each pixel (lower left). When blending is used for composition, the average, standard deviation, and density (number of trajectories contributing) are used to select the appropriate blending color (upper middle). Alternatively, for a woven composition, a random GPA is chosen, and when combined with the density, selects the appropriate color.

relevant attributes, especially GPA: both the average value and variability. For the application, satisfying these requirements addresses the following domain-specific questions:

- What is the distribution of the number of students?
- How successful were the students on a per semester basis?
- How consistent is performance from student to student?

In order to display intuitive colors for the application, a Green-Yellow-Red color scale, as described previously, is used. By modulating the saturation by the standard deviation, areas where contributing trajectories have similar attribute values deepen to saturated colors. Areas where the contributing trajectories disagree or have a wide variance have low saturation and are washed-out (i.e., gray). Finally, density corresponds directly to the brightness component, and, against a black background, provides an opaque representation where the maximum density is reached.

In Figure 5.3, the color blending step is shown in the upper middle of the figure and is comparable to the earlier diagram in Figure 3.18. The first example pixel has more trajectories contributing to the composition and therefore results in a brighter color since the density is higher. The first pixel also has a lower mean GPA value and a higher variance, resulting in a lower saturation on the lower end of the color scale. On the other hand, the second example pixel has fewer trajectories (less density) and therefore uses a darker color value that, when placed against black, provides the intuitive appearance of less density. The second pixel has a higher mean GPA value but lower variability, resulting in a higher color value that is more saturated.

The weaving method also uses the same hue-saturation-brightness color model, but does not modulate the saturation by the standard deviation. Instead, the algorithm randomly chooses a trajectory's attribute value and selects the appropriate hue component for

the weaving cell. Density is still used with the brightness component to correctly convey the number of students and therefore the group's overall structure. Weaving provides the viewer with an exact value of a single data point in that region; deviation must be inferred by looking at neighboring point samples. The lower portion of Figure 5.3 shows a simple depiction for how the weaving color is chosen. First, a random value is chosen from each pixel list (lower left). This value selects the weaving color, and the density determines the brightness (lower middle).

5.4 Results

The approach is used to explore historical course data for undergraduate computer science and computer engineering students. The following case study provides real-world results of how trajectory composition can aid in understanding large trajectory sets, especially when they can be spatially represented and grouped into natural bins. Figure 5.4 composites all 1,456 computer science students together into a single rendering using the composition process. Both a blended approach (top) and a woven version (bottom) are shown in the figure. Several features immediately stand out. The average trajectory (the white line) has a upward slope representing that, in general, most students successfully progressed to more advanced classes over time. Ninety percent of students were finished by the tenth semester, while fifty percent never made it past the fifth semester—possibly graduating if they were transfer students. In regions below the average trajectory, the composition is tinged red, indicating that the students had lower grades (i.e., closer to Cs or 2.0s) than those above the line (shown in shades of green). There were significant variations in the grades, as shown by the washed-out appearance closest to the average trajectory. The variations can also be inferred from the woven version on the bottom—there are significantly more red patches below the trajectory. However, these patches also occur above as well, in-

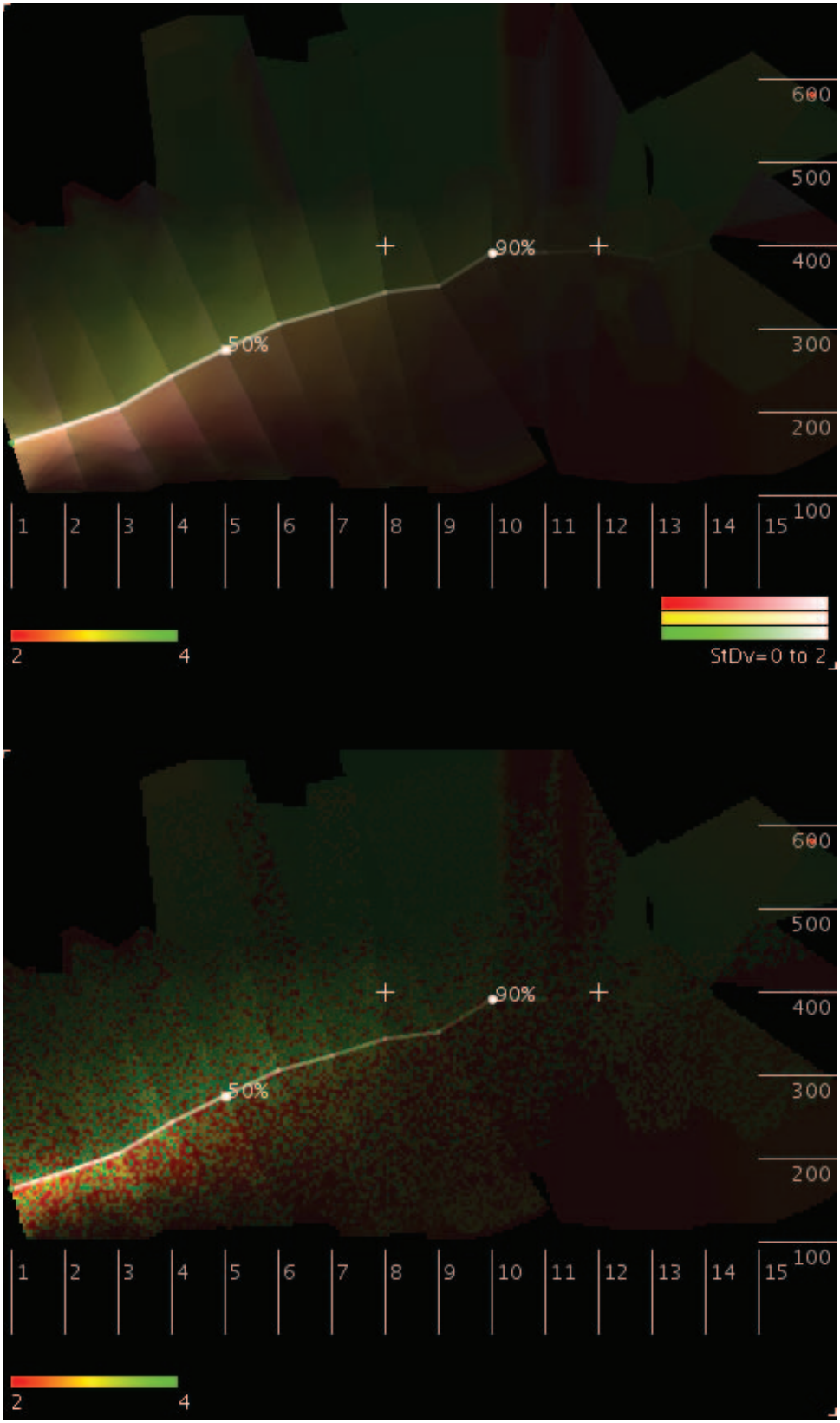

FIG. 5.4. Blended and woven composition for all computer science students. Color is mapped to the semester GPA for each student.

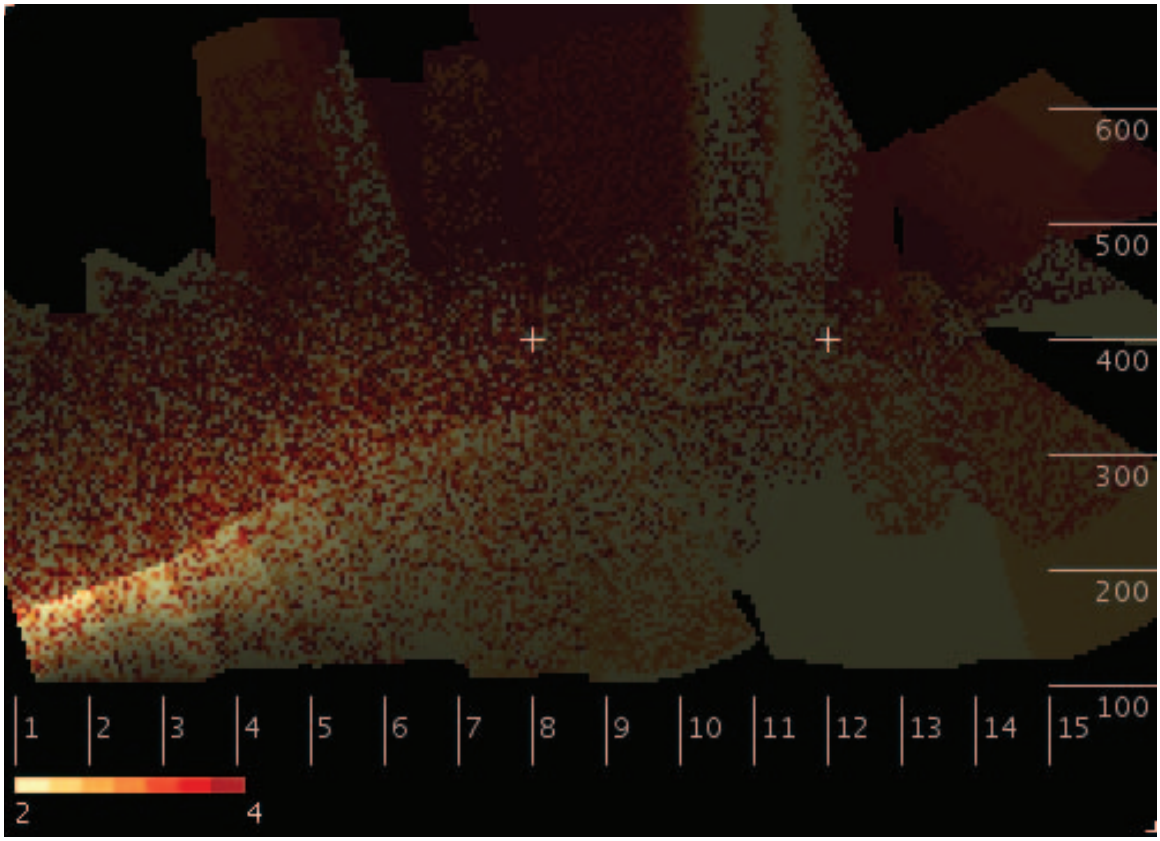

FIG. 5.5. Woven version using a sequential Brewer colorscale.

dicating a range (i.e., variation) of differing semester grades. The remainder of the results use a small multiples (Tufte 1990) approach to compare and contrast course-grade bins and specific demographics groups—small multiples is an approach where different combinations of the overall dataset are depicted in a grid to understand similarities and identify differences.

Figures 5.2 and 5.4 both depict all of the students within our data set. However, the structure of the overall group becomes much clearer when composited together. For example, while some density information can be derived from the number of lines in Figure 5.2, it is difficult to determine the central trajectory for data. GPA information is also much more difficult to infer—especially what the average value is and where it significantly

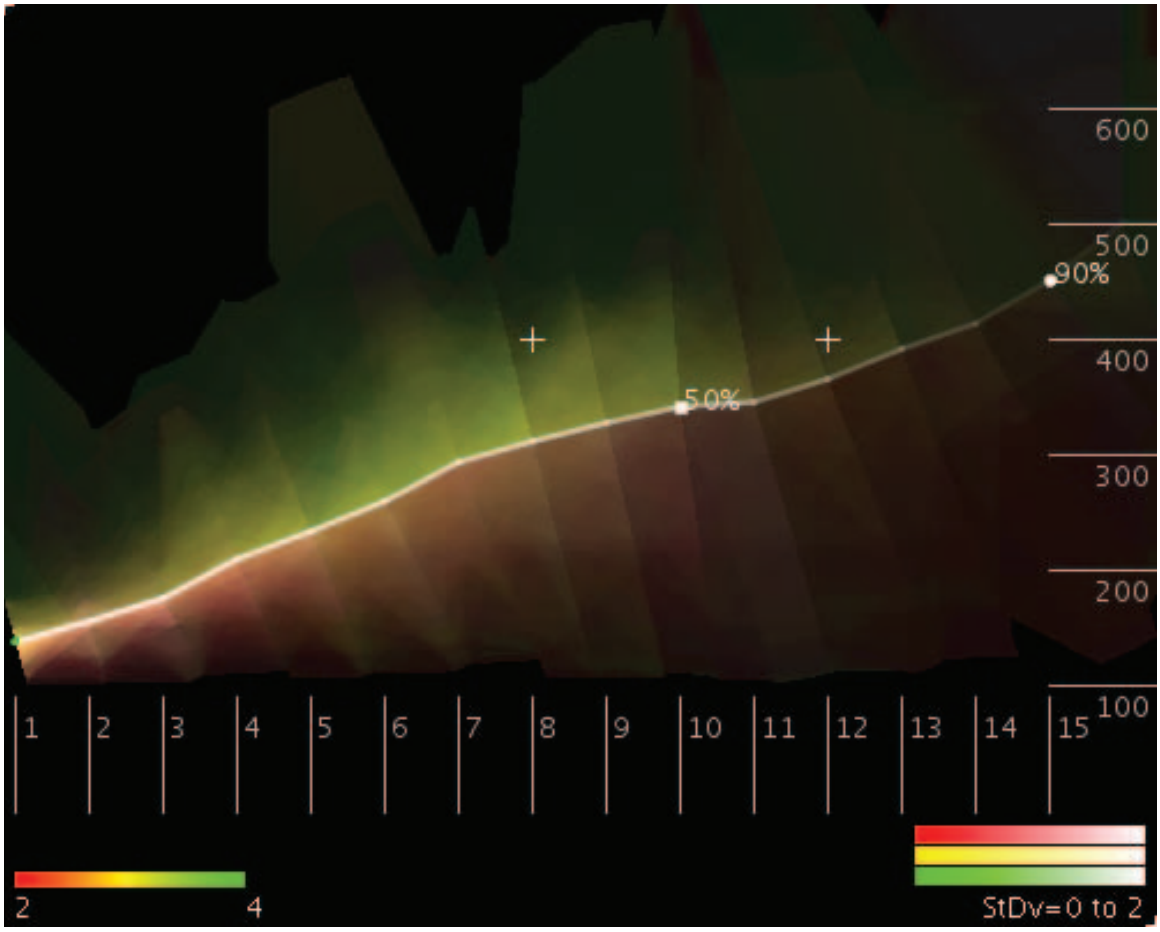

FIG. 5.6. Blended composition for all computer engineering students. Color is mapped to the semester GPA for each student.

deviates. On the other hand, the composition in Figure 5.4 clearly shows the deviations towards the beginning of the students' careers and the overall trend throughout the student trajectories. As an alternate color scheme, Figure 5.5 shows the same composition but using a sequential red-orange-yellow Brewer colorscale. Because the colorscale contains only discrete color indices, the woven color model is a more natural fit than the blended model.

As a comparison to the computer science students, the computer engineering composition for all students is shown in Figure 5.6. The figure composites 408 students together

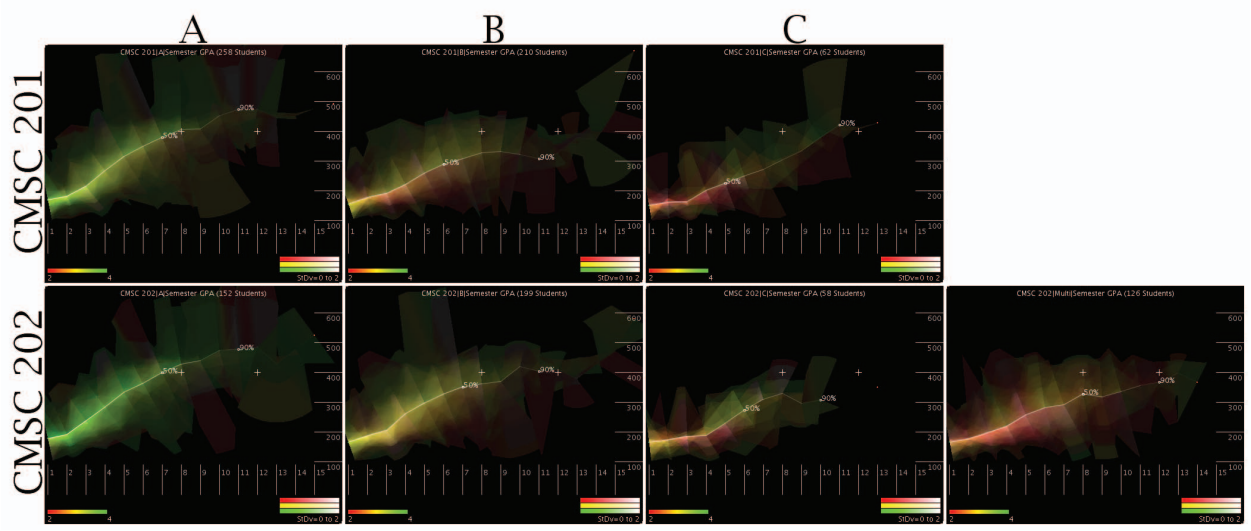

FIG. 5.7. CMSC gateway course trends. Small multiples shown for each gateway course-grade combination.

(as opposed to 1,456 for computer science). As described in the approach section, fewer trajectories lead to compositions that are more granular and that have more abrupt transitions. The most surprising aspect was that half of computer engineering students stayed in the program until the tenth semester, as compared to the fourth semester for computer science majors. Statistically, the median and average number of semesters for computer engineering students was ten and eight. For aspiring computer scientists, the median and average were four and five, respectively. However, the computer engineering data set may have been incomplete—for instance, there are no computer engineering transfer students. Other trends are noteworthy. For example, computer science students tended to advance more quickly to higher-numbered courses. Additionally, computer science students tended to show more variance in grades throughout their careers, regardless of their progress, whereas computer engineering students had slightly higher variance if they weren't progressing smoothly (note the washed-out appearance for computer engineering students below the line in Figure 5.6.

The computer science gateway courses, CMSC 201 (Computer Science I) and CMSC 202 (Computer Science II), exhibit the expected trajectories for the A students (Figure 5.7, column one)—specifically, successful advancement towards higher-numbered courses. However, students who received a B in CMSC 201 (row one, column two) struggled, relative to those receiving a B in CMSC 202 (row two, column two)—note the leveling off effect towards the eighth semester for CMSC 201; a B in CMSC 202 had little effect on the trajectory but did affect grades (i.e., more yellow). This indicates the relative importance of excelling at the concepts taught in CMSC 201. This is further emphasized by the relative performance of C students for CMSC 201 versus CMSC 202—namely, the dropout rate (50th percentile marking, although both occur early) and the ending location. The C CMSC 201 trend line (row one, column three) indicates higher performance after an initial struggle (flat initial line). A careful examination of the underlying data reveals that the increased slope was due to students taking high-level math courses (e.g., MATH 381 (Linear Methods in Operations Research), STAT 451 (Intro to Probability Theory), STAT 454 (Applied Statistics)). Since the computer science program requires a B or better in both CMSC 201 and 202 and doesn't require any of these courses, these students most likely switched to another major. The last column shows the students who took the courses multiple times before completing with an acceptable grade. CMSC 202 had 126 students who repeated the course. This group's perseverance is notable—out of all of the course-grade combinations for the gateways, these remained in school the longest but in many cases did not progress to the 400-series courses necessary for graduation.

As stated earlier, one objective for visualizing aggregate student histories is to identify critical points in the curriculum that determine a student's success. For the gateway courses, both the students receiving a B in CMSC 201 and those completing CMSC 202 after multiple attempts should have been able to successfully complete their degree requirements. However, the trajectories tell a different story. From these results, in-depth analysis

should be performed to determine which skills are not adequately addressed for these students. Then, the curriculum and appropriate classes/class topics should be modified to overcome these deficiencies. Alternatively, students falling into these categories could be offered additional support to supplement their knowledge and skill sets.

Figure 5.8 shows the results for analyzing computer engineering students completing the gateway courses for their major. Horizontally from left to right, the individual tiles represent the student's grade—A, B, C, or multiple attempts, respectively. Each row represents a single class as follows:

- CMSC 201, Computer Science I
- CMPE 212, Principles of Digital Design
- MATH 251, Multivariable Calculus
- PHYS 122, Introductory Physics II

For computer engineering majors, the gateway courses paint an inconsistent picture (Figure 5.8). Whereas computer science students exhibited a correlation to the plateauing or regression for poor performance in the gateway courses, computer engineering students exhibited this characteristic for stronger grades. In the figure, leveling off is noted in students who received an A in CMPE 212, MATH 251, and PHYS 122 (first column in the figure). One explanation is that the course curriculum numbers differs from computer science–e.g., the leveling off effect is expected due to the recommended courses. However, a careful review of the recommend curriculum reveals that the average course numbers do increase monotonically by substantial amounts. A closer look at the actual data did not reveal significant insights—while the leveling off was confirmed, no apparent reason was immediately obvious. Students receiving a B in all of the gateway courses (second column) except for CMSC 201 did not exhibit this feature. C students in MATH 251 and PHYS 122

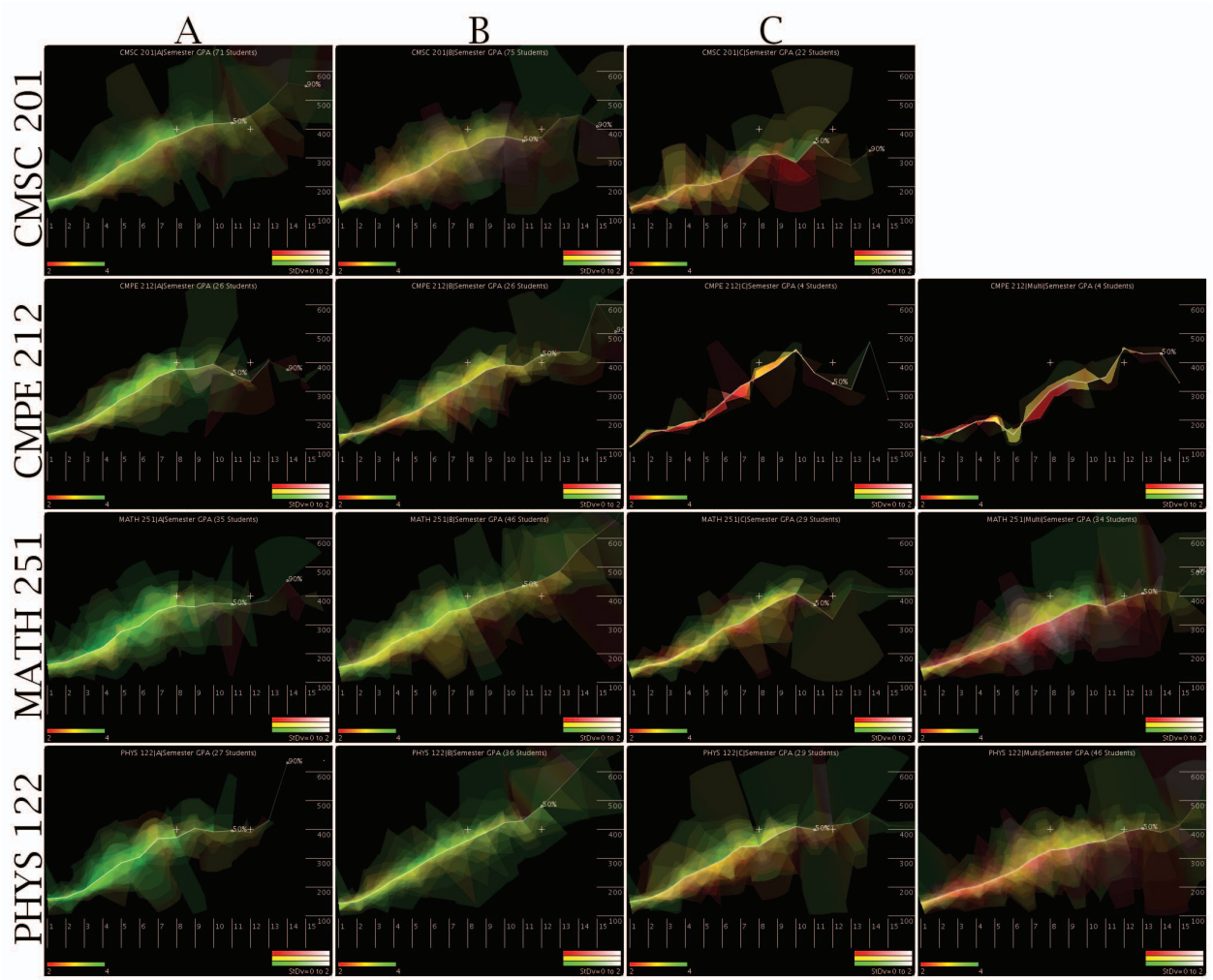

FIG. 5.8. CMPE gateway course trends. Small multiples shown for each gateway course-grade combination.

progressed similarly to their higher grade counterparts up to the tenth semester. However, later semesters for MATH 251 C students exhibited a decrease in course progression—as noted early, low sampling over exaggerates the trend line. Multiple takers in MATH 251 and PHYS 122 showed the strongest signs of struggling. In addition to the slow course progression, these two groups also showed the lowest semester grades across all of the gateway groupings—these students were most in need of additional support or intervention. However, similar to computer science, computer engineering semester grades correlated well with performance overall–e.g., A students in any of the particular gateway courses exhibited higher semester GPAs (i.e., more green) throughout their history.

The computer science core courses are shown in Figures 5.9 and 5.10. From top to bottom, the rows represent the following courses:

- CMSC 201, Computer Science I
- CMSC 202, Computer Science II
- CMSC 203, Discrete Structures
- CMSC 304, Ethical and Social Issues in Information Technology
- CMSC 313, Computer Organization and Assembly Language Programming
- CMSC 331, Principles of Programming Languages

Figure 5.10 has the remaining core courses:

- CMSC 341, Data Structures
- CMSC 345, Software Design and Development
- CMSC 411, Computer Architecture

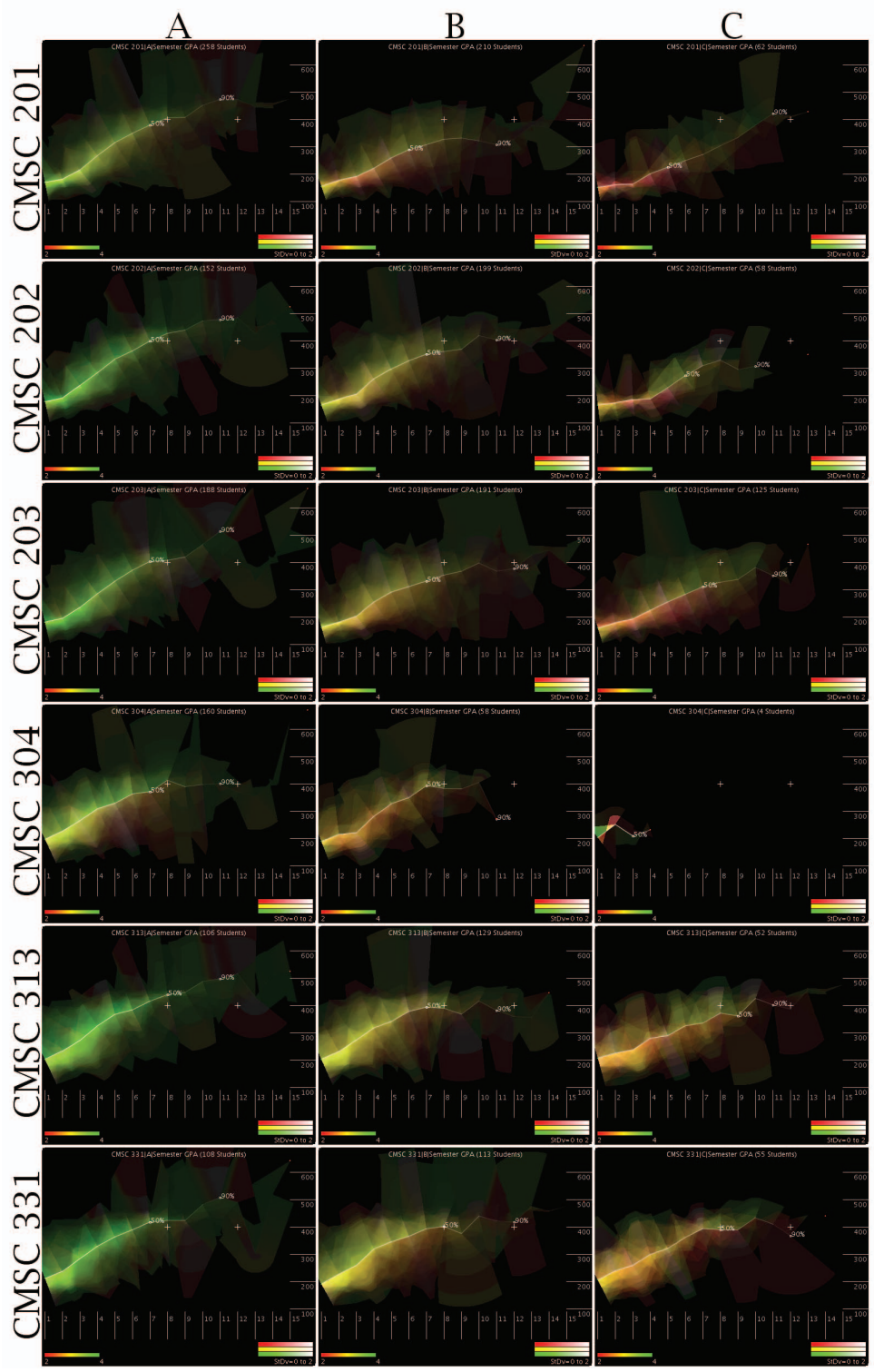

FIG. 5.9. CMSC core course trends. Small multiples shown for each gateway course-grade combination.

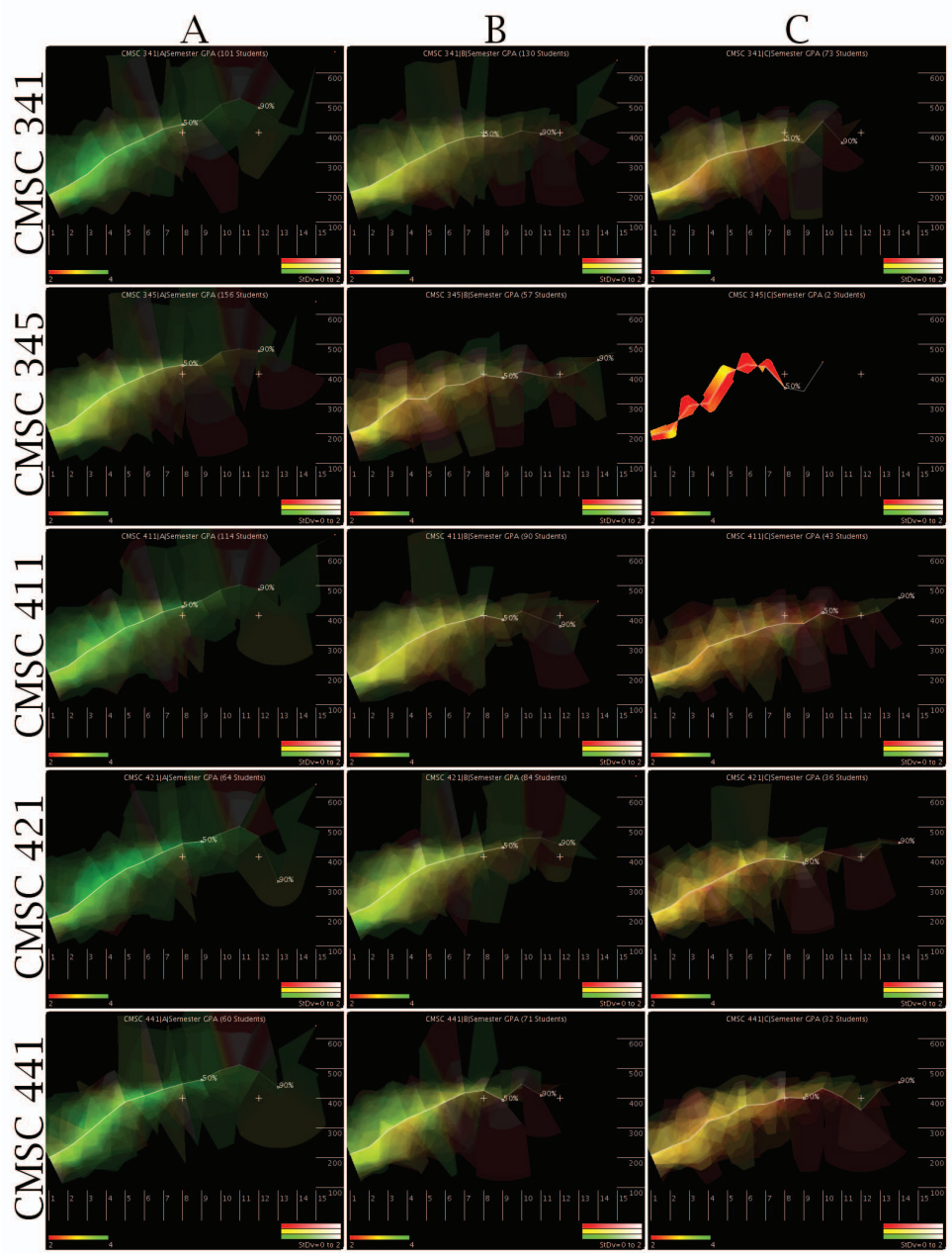

FIG. 5.10. CMSC core course trends. Small multiples shown for each gateway course-grade combination.

- CMSC 421, Principles of Operating Systems
- CMSC 441, Design and Analysis of Algorithms

Computer science students who received a C in CMSC 201 (first row, third column in Figure 5.9) experienced significant losses over at least two steps. The initial step, from the second to third semester, shows a sharp decline in volume (represented by opacity in the visualization), which, when compared with the actual data samples is accurate—from 58 to 39 students, roughly two thirds. From there, the volume drops by half by the fifth semester (39 to 19). Since the color opacity corresponds directly to the number of contributing students (i.e., density), these patterns become quickly apparent.

In addition to the density representation, the saturation of the color (i.e., the standard deviation of the underlying data) is key to understanding the underlying meaning of the attribute rendered—in this case, semester GPA. For example, students completing CMSC 203 with an A (row three, column one) and those finishing CMSC 304 with an A (row four, column one) differed significantly in their accumulated GPAs. The CMSC 203 A students exhibit a more vibrant green, indicating less variability while the CMSC 304 A students have a washed-out appearance (i.e., high variability in their semester GPAs). CMSC 304, Ethical and Social Issues in Information Technology, is a non-technical requirement, and a student's performance in CMSC 304 has little to no relation to their academic abilities in computer science (i.e., no correlation to overall progress—as denoted by the similar average trajectories for both diagrams). Furthermore, CMPE 304 appears to be an easy A or B based on the limited number of students receiving a C in the course. CMSC 345, Software Design and Development, has similar characteristics—i.e., almost no C students.

In addition to the variability in accumulated GPAs, the ending point and direction of the average trajectory indicates the relative importance of each course/grade combination to the computer science major's principles. For example, CMSC 201, CMSC 202, CMSC

203, and CMSC 411 all show widely varying fiftieth percentile marking locations. The fiftieth percentile is marked on all of the images because it identifies where fewer data samples (i.e., fewer students) begin to cause the average trajectory to have erratic movements. The horizontal and vertical location of the fiftieth percentile marking indicates the significance of these courses for the overall progression of students. For example, the vertical marking at the higher-numbered course level indicates more students progressed to higher-level courses. Horizontal locations can be ambiguous—an earlier marking may mean that more students dropped out of the major or that students graduated earlier. To understand the horizontal position of the fiftieth percentile mark, the vertical position needs to be taken into account.

The core computer engineering courses describe the required classes across the engineering curriculum. Figure 5.11 shows the results for students completing the following courses with "A", "B", or "C" grades (left-to-right columns in the figure). The core courses are as follows (top to bottom in the figure):

- ENES 101, Introduction to Engineering
- CMPE 212, Principles of Digital Design
- CMPE 306, Basic Circuit Theory
- CMPE 310, Systems Design and Programming
- CMPE 314, Digital Electronics
- CMPE 320, Probability and Random Processes
- CMPE 450, Capstone I
- CMPE 451, Capstone II

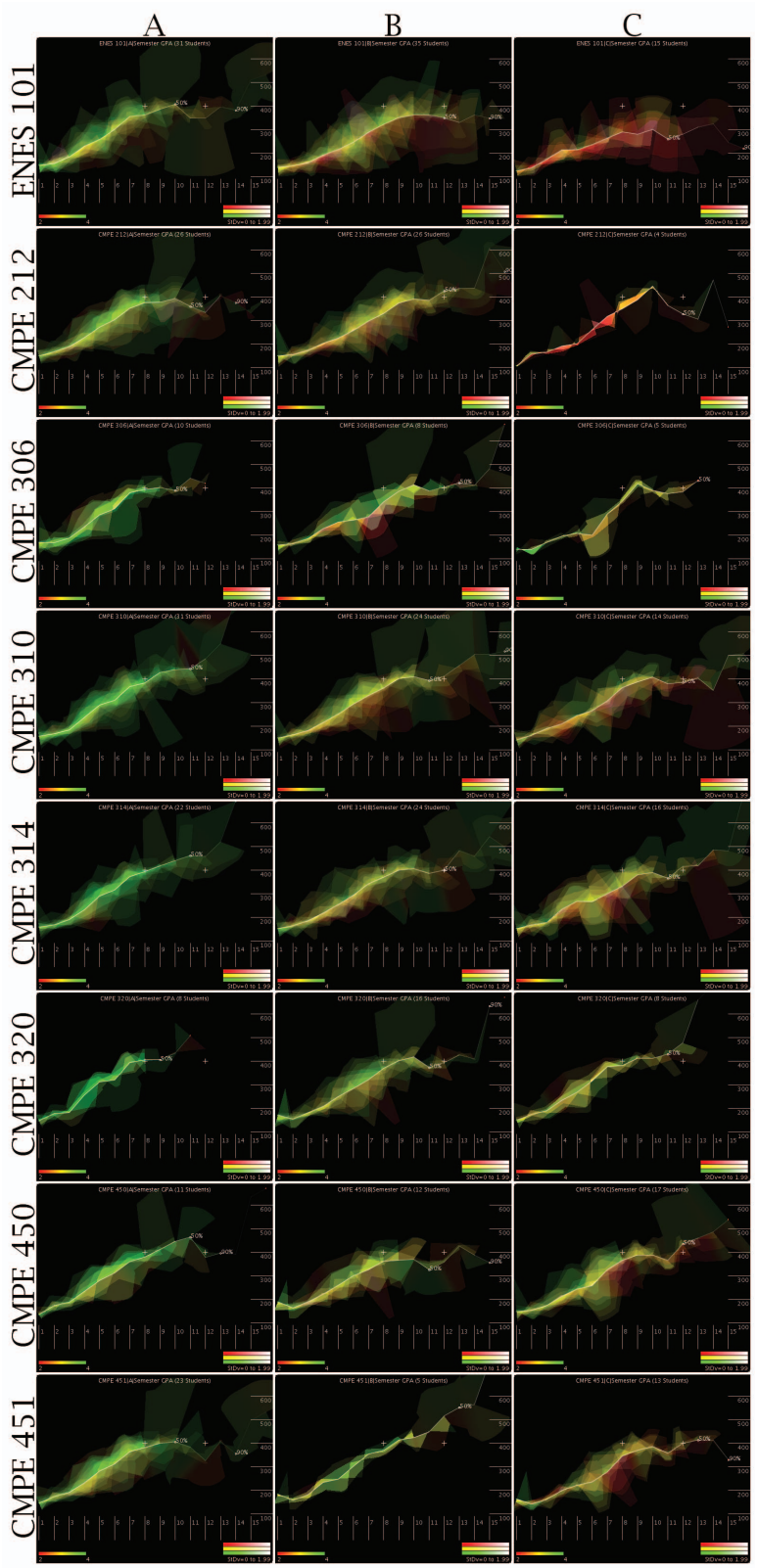

FIG. 5.11. CMPE core course trends. Small multiples shown for each core course-grade combination.

The premature leveling-off effect is most pronounced for B and C students in ENES 101, indicating the relative importance of the concepts taught. Leveling-off does occur naturally in other courses, but these cases show that the effect did not occur until the student had reached the 400-series courses (e.g., A students in CMPE 212 and CMPE 451). In many of the CMPE cases, there just weren't enough trajectories (i.e., students) to smoothly aggregate the data—35 is the highest number of students in Figure 5.11. The low number of students led to exaggerated distortions in the compositions. For example, students completing CMPE 212 with a C (second row, third column) show a sharp jump at the end of their careers towards lower number courses. While the underlying data does show a significant decrease at the fifteenth semester, the composite exaggerates this amount. The root cause of the problem is the limited number of students that contribute to the end of the trajectory due to the length averaging modifications made for this dataset. Some possible fixes are outlined in a later section.

The first three math courses (MATH 150 (Precalculus), MATH 151 (Calculus I), and MATH 152 (Calculus II), Figure 5.12, rows ones through three, respectively) show similar trends for students receiving Bs or Cs. MATH 150 is technically not required for computer science; students enrolled in this course were likely addressing gaps in their high school programs. However, students not receiving an A for this course showed the worst trends for all of the groups—i.e., very shallow progression, an earlier 50th percentile for students leaving the program, and lower semester GPAs, as depicted by the red coloring. MATH 150 B and C students also had relatively narrow densities, indicating that the students were restricted to very similar courses for their (short) computer science careers. The red areas are most prevalent for C students in either MATH 150 or MATH 151, indicating the correlation between mathematical ability and computer science success. The trend is not quite as apparent for the remaining math courses, possibly revealing independence between the course concepts and the needs of computer science.

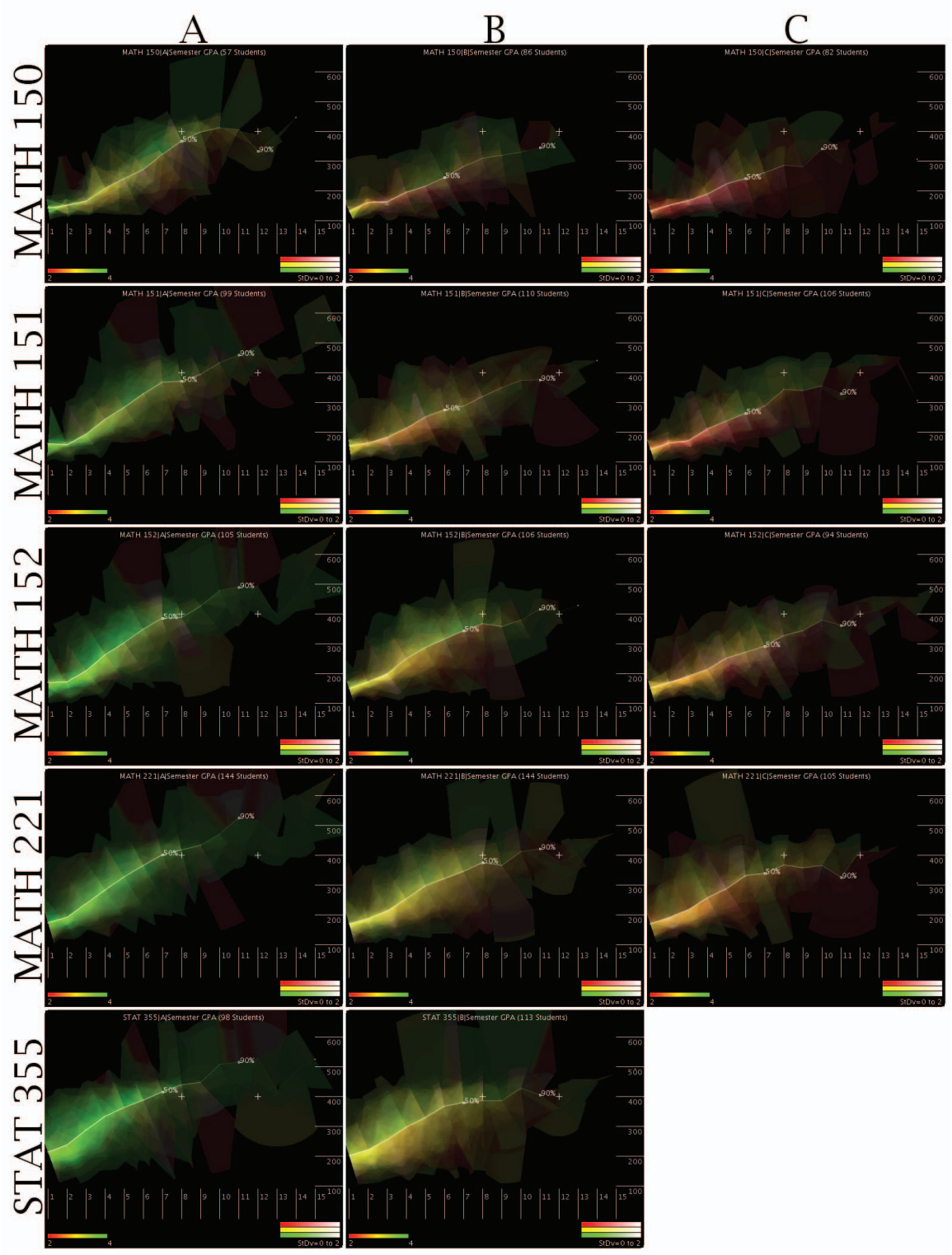

FIG. 5.12. Core math for computer science students

Students receiving an A for the required math courses (column one in Figure 5.12) had a much stronger tendency for A GPA semesters than those in either the B or C categories. This held true regardless of which side the student fell on from the average trajectory. Note the green areas occurring on both sides of the average trajectory for the A math students (i.e., column one) as compared to the other grades—regardless of the student's trajectory, they retained high semester GPAs throughout their academic careers.

For computer engineering students, MATH 151 (Calculus I) most heavily influenced later course progression. In particular, MATH 151 C students showed the overall worst trend in their course progression. MATH 150 students also struggled regardless of which grade they received. Since this course is not required, these students most likely entered the program at a disadvantage and were never able to fully recover. The overall grade remains relatively fixed for MATH 151, MATH 152, and MATH 221—namely, that the course grades correlated well with the student's semester grades. For MATH 225 and MATH 251, however, the B and C students had roughly equivalent semester grades. This may indicate that proficiency in these courses does not overly affect a student's progression for computer engineering.

5.4.1 Demographic Compositions

To compare high school GPA partitions with SAT Math performance (Figure 5.14), the students were first binned in arbitrary increments of 75 for their SAT Math scores (e.g., 800 to 725, 725 to 650). Next, the students were sorted by their high school GPA. From the sorted list, partitions were calculated to match the number of students in the SAT Math bins. For example, there were 100 students in the highest SAT Math bin. Students from 1 to 100 in the sorted GPA list were then chosen for comparison against the highest SAT Math bin (their high schools GPAs ranged from 5.0 to 4.21). Due to multiple students falling into discrete GPA bins, the number of students between the corresponding SAT Math and high

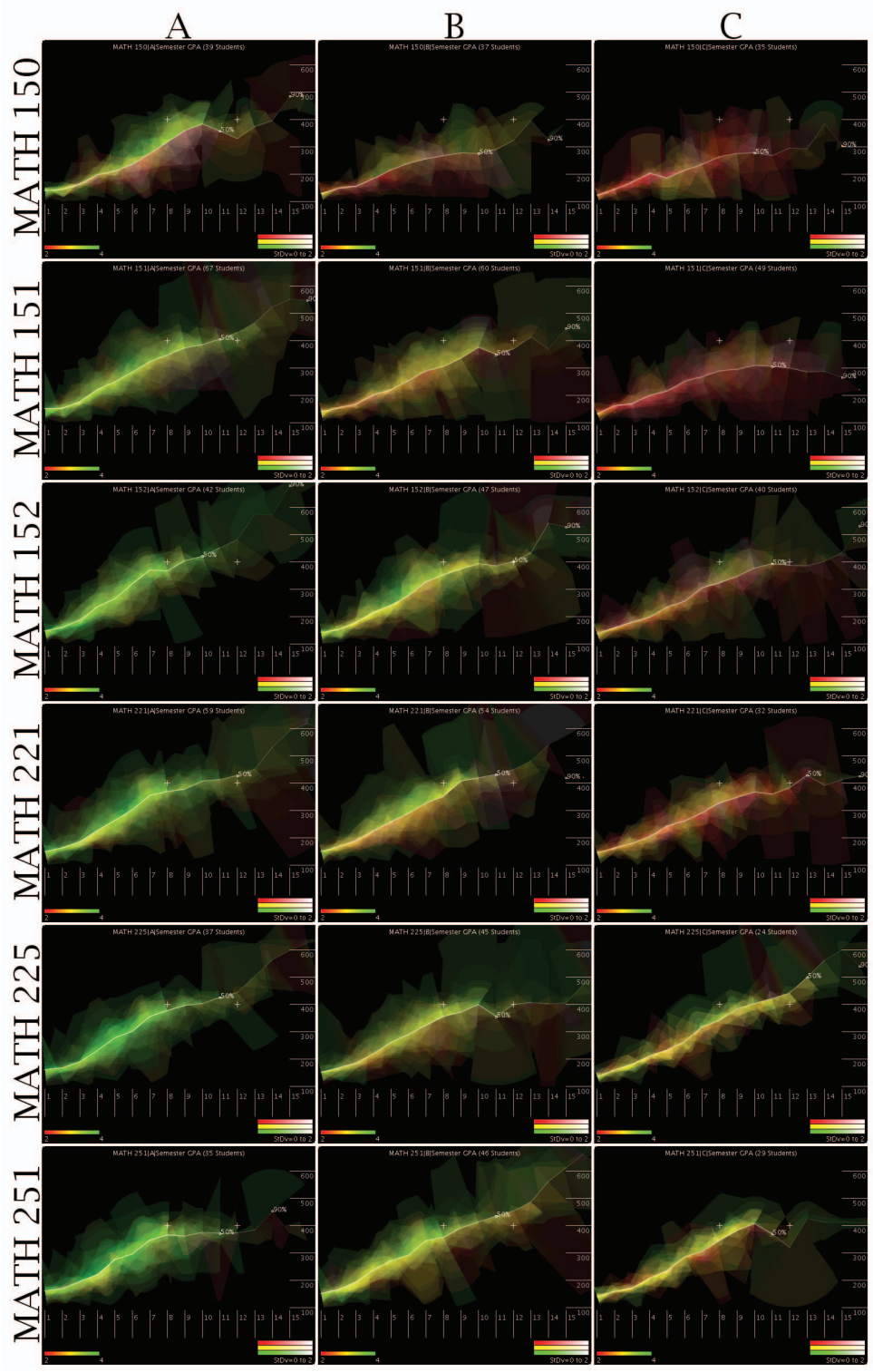

FIG. 5.13. Core math for computer engineering students

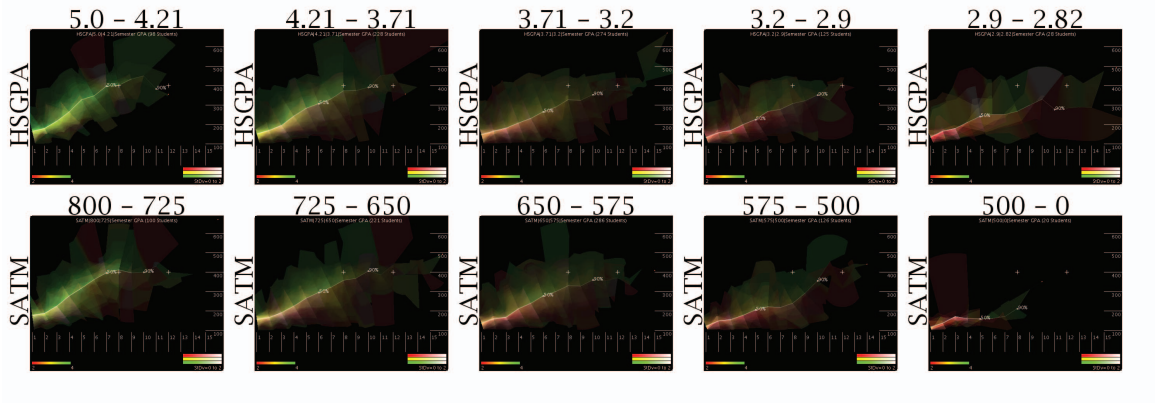

FIG. 5.14. High school GPA versus SAT Math for computer science students

school GPA bins were not exactly equivalent.

The results between the high school GPA partitions and the corresponding SAT Math bins did not show significant differences—especially in the three highest groups (Figure 5.14, columns one through three, rows one and two). For example, the top SAT Math bin (i.e., 800 to 725) compared very similarly to the highest high school GPA bin (i.e., 5.0 to 4.21). The SAT Math results did show a higher variance in semester GPAs as shown by the somewhat washed-out appearance in the composition. However, there was only a slight increase in the standard deviation. The fourth column does show a degree of variation between the separate bins. In particular, the high school GPA group has a steady progression towards higher-numbered courses, indicating some amount of success despite the lower semester grades. The equivalent SAT Math grouping shows a leveling off around the fifth semester followed by a sharp dive in the eighth. Other than this last example, these results are contrary to conventional wisdom—high school GPA does not appear to be significantly more important than SAT Math scores.

Except for the lowest performing bins, every computer engineering curve for both the high school GPA and SAT Math partitions exhibited significantly lower performance below the curve (i.e., students progressing more slowly than average for the bin) versus

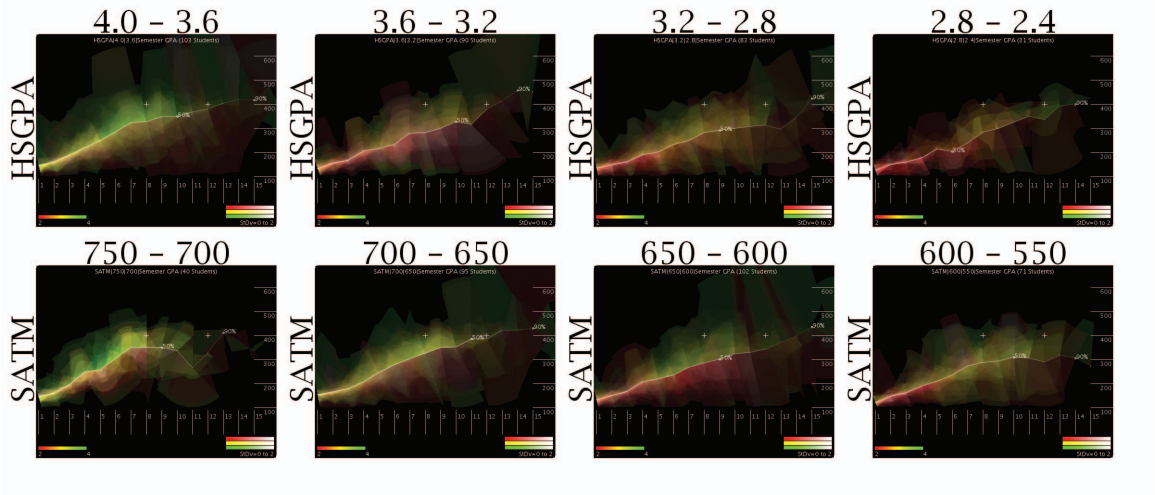

FIG. 5.15. High school GPA versus SAT Math for computer engineering students

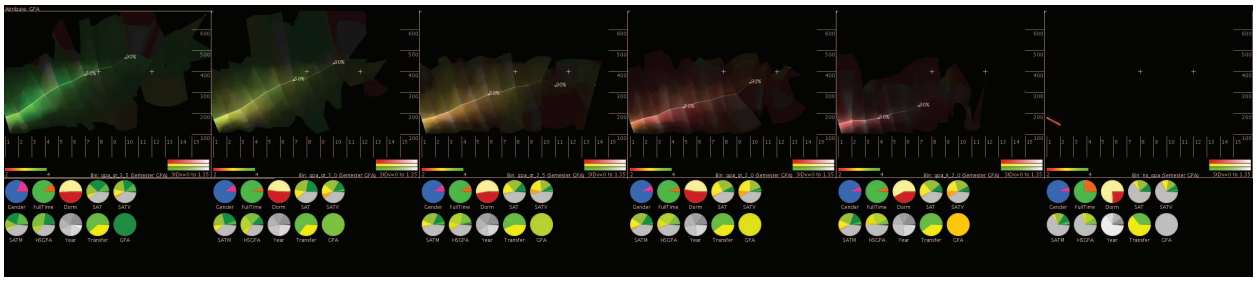

FIG. 5.16. Final GPA partitions for computer science students

those above the curve. This is shown as the stronger green/yellow color above and the mostly red color below. Unlike their computer science equivalents, the highest high school GPA bin (column one, row one) showed the smoothest progression towards higher-level courses and the highest semester GPAs, as depicted by the strong green color. The second SAT Math bin (column two, row two) shows the strongest monotonically increasing trend for the SAT Math row. The third bin appeared to be most similar for the corresponding high school and SAT Math scores.

Both the computer science and computer engineering final GPA partitions (Figures 5.16 and 5.17, respectively) trend as expected: students with high final GPAs (left-hand

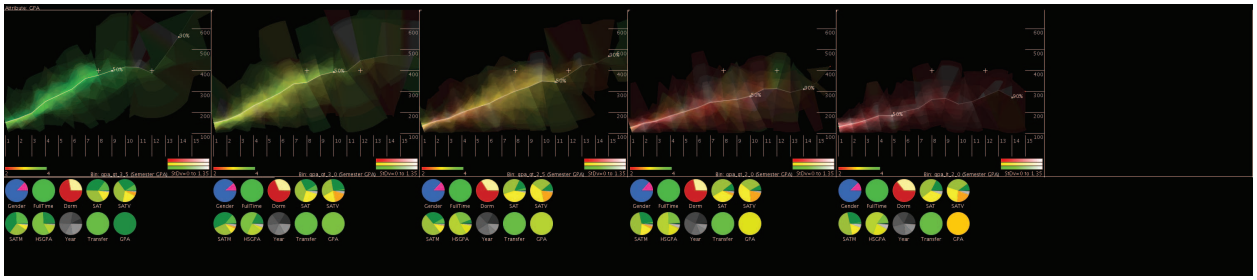

FIG. 5.17. Final GPA partitions for computer engineering students

side) had much greener compositions that progressed more smoothly to advanced courses than their lower GPA counterparts (right-hand side). Surprisingly enough, the saturation was fairly consistent across all of the small multiples, indicating little variance for semester GPA when grouped and binned by the final GPA. However, the third bin for both data sets (i.e., GPA between 2.5 and 3.0) showed higher levels of variance.

When compared with students entering as freshmen (Figure 5.18, top), transfer students (bottom) started at a higher course number due to prerequisites/early course work already being completed. However, the remaining trend is roughly the same, including the number of semesters completed. This may indicate that transfer students don't actually save any time or money by completing the initial courses at another university. Unfortunately, the provided dataset did not show any computer engineering students that transferred into the program.

5.4.2 Identifying Plateaus

During discussions with academic subject matter experts, the flat-line trend late in a student's history was identified as particularly interesting. To identify other course-grade pairs that exhibited the same characteristic, the student course history was analyzed using clustering to find commonalities in course-grade trajectories. In order to use clustering, a distance metric had to be defined to characterize the feature of interest—in this case, an

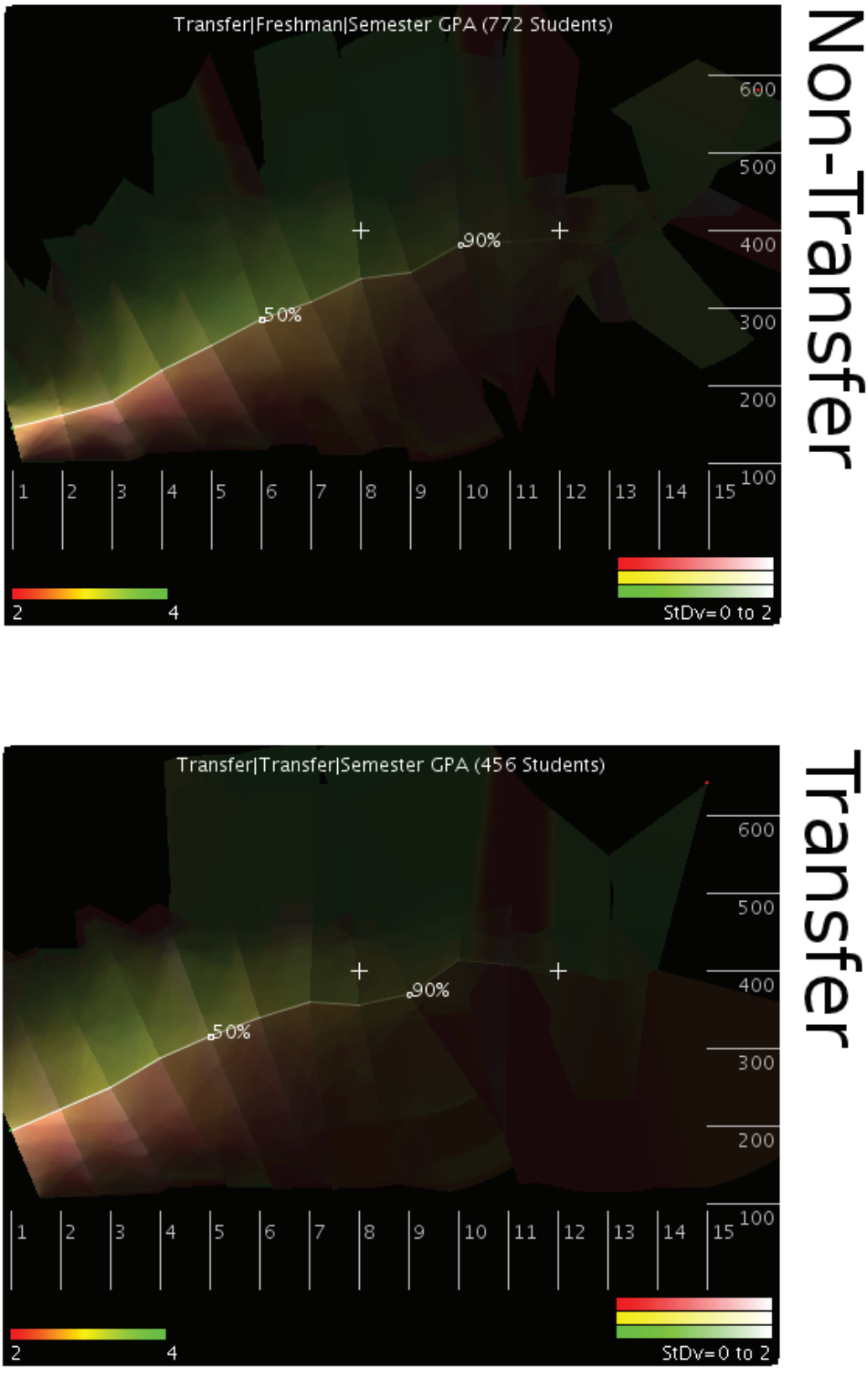

FIG. 5.18. Transfer status for computer science students

initial upward slope followed by either a plateau or decrease in performance toward the end of the student trend line. A distance metric was created to compare the students' trajectory (i.e., slope) irrespective of the coordinate spatial position. To accomplish this goal, each spatial trajectory was broken into three parametric parts. For each parametric part, the normal vector was calculated. To compare two trajectories, the root mean square of the angle difference between the two comparison trajectories was calculated.

The items for the clustering algorithm were derived from each possible letter grade and course combination. Each combination was aggregated and the average spatial trajectory was calculated for that specific pair. For example, all students who received an A in CMSC 201 were averaged into the same trajectory and the "CMSC 201 A" trajectory was then compared against all other course grade combinations.

Figure 5.19 shows the main clusters resulting from the clustering algorithm. In the figure, six separate clusters are shown. Within each tile, the course-grade trajectories (colored by grade) and the overall trajectory (white) are shown. In addition to the trajectories, the individual course-grade combinations that create the trajectory are shown in the bottom of each tile. All of the tiles show leveling off or dips late in the students' careers. Cluster 24 is of particular interest since all of the grades show As but a sharp dip occurs at semester 13. A careful examination of the underlying data shows students taking either PHYS 122 or lower numbered math courses—possibly to satisfy degree requirements. Across many of the course-grade trajectories, this was a common theme late in a students career—i.e., taking early requirements. Cluster 21 is also significantly different. Most of the courses represent early general science or engineering topics. As one can see, the inability to grasp fundamental scientific concepts is fatal to progressing to higher course numbers. As a result of this analysis, educators can revise the curriculum to address identified deficiencies—for instance, by moving core concepts (e.g., math) to precede courses that require that skill set or by providing refreshers.

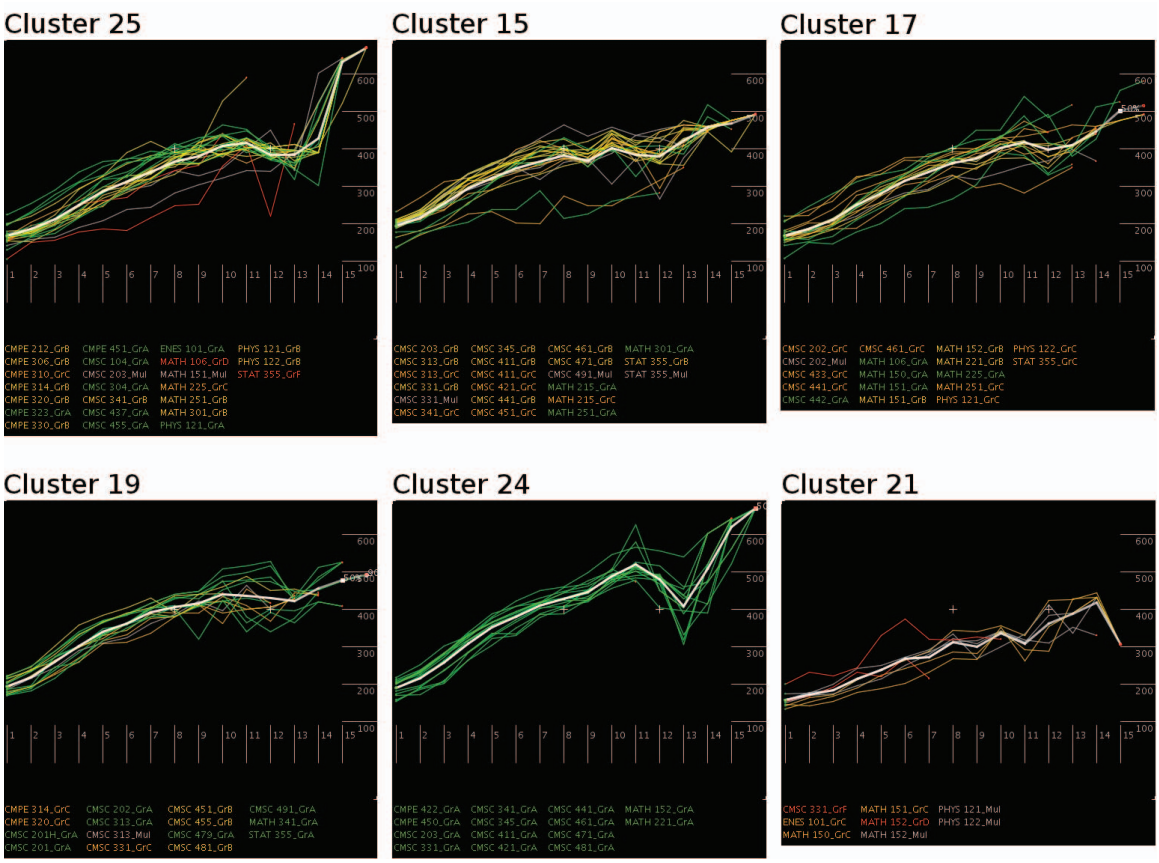

FIG. 5.19. Cluster results for average course-grade trajectory bins (computer science)

5.5 Complete Population Clustering

In addition to grouping the students by their course-grade combinations or natural demographic identities, they were also clustered by their course progression. This determined the most common course curricula and to see additional correlations to demographic information. To cluster the data, a distance metric was developed to compare two student course histories and calculate a similarity score. The results of the clustering process were then fed into the composition visualization technique to understand trends in student GPAs. However, while anomalies and patterns were identified in the visualization, understanding the underlying cause was often difficult.

5.5.1 Clustering Distance Metric

Clustering algorithms require a distance metric which compares two student histories and provides a score on their similarity. This metric is used by the clustering algorithm to associate the most similar students into distinct partitions. In order to satisfy the goals, the distance metric should yield low scores for students who have similar or nearly identical histories; variances in when, or whether, the same courses are taken should result in higher, or less similar, scores. When a course is taken at a later point in a student's history, this may indicate low academic performance or repeated attempts to pass the course. Courses that are unique to one of the students occur more frequently in specialized course tracks. In the computer science dataset, specializations may exist in networking, graphics, databases, etc.

Because general education courses in unrelated areas (e.g., political science, history) are not strongly correlated with student performance in the computer science curriculum, these courses are ignored by the distance metric. However, related areas such as mathematics and engineering are included and compared along with the computer science curriculum

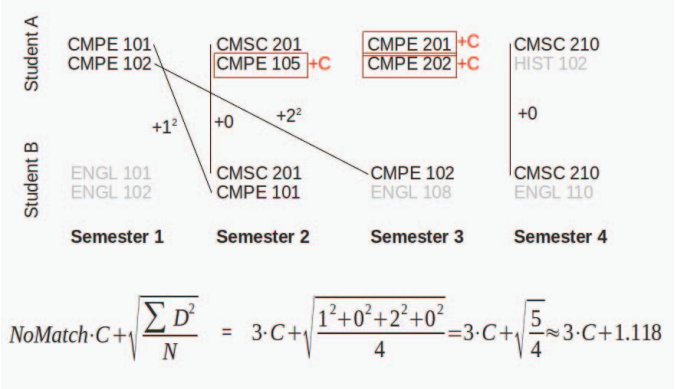

FIG. 5.20. Course comparison example. For courses in common, a root mean square calculation based on the relative time for each course is used. In cases where one student did not have a corresponding match (courses in red), an empirically derived constant is added to the final sum.

classes.

Unlike other research efforts that focus on student subject choices (Ferral 2004) and preferred topics (Quevedo *et al.* 2009), the distance metric accounts for the temporal nature of a student's history. The distance metric is the root mean square of the mean of the sum of the differences of the course occurrence with respect to the relative semester index. For example, if Students A and B take "CMPE 212" in their first and third semesters, respectively, then four (the squared value of three minus one) will be added to the accumulated sum. If, however, they both take the course in the same semester, then the sum will not be incremented (i.e., zero squared). In cases where the course is absent (i.e., one of the students did not take the course), an empirically derived constant is added to the final calculation. By adding this constant, students who did not share courses in their histories are significantly separated from one another. Figure 5.20 shows a graphical depiction of the distance metric scoring for two simplified student histories.

Students who repeat the same course are handled as a special case. When the distance metric attempts to find a matching course for Student A, it starts by examining the same

semester index for Student B (i.e., if CMSC 341 was taken in Student A's fifth semester, the algorithm would first look for that course in Student B's fifth semester). If Student B's semester does not have the same occurrence, then the semesters adjacent (i.e., the one right before and the one right after) to the algorithm's starting semester are examined for the course. If a match is still not found, then the adjacent semesters are searched in an incremental sequence until a match is found. This greedy approach guarantees that the closest course (i.e., the minimum distance) will be added to the accumulated sum. Once a match is found, the distance metric's state is updated to exclude the matching course from future comparisons. By applying this heuristic, repeated courses can be accounted for in a systematic function that satisfies the goal to separate students with course variations.

5.5.2 Clustering Results

Figures 5.21 and 5.22 depict the largest clusters for computer engineering and computer science, respectively. For both figures, the rows correspond to the following attributes (from top to bottom): High School GPA, College Final GPA, College Accumulated GPA, and Semester GPA. The columns denote each cluster—for computer engineering (Figure 5.21), the cluster sizes from largest to smallest (left to right) are 41, 29, 19, 16, and 12. For computer science, the clusters were significantly larger because of the data set size—176, 88, 86, 79, and 33, respectively.

The two largest clusters in Figure 5.21 show that most students completed their studies roughly around the 400-level series. Furthermore, these students required around eleven semesters to complete the work. Judging by the slightly washed-out appearance in the final GPA tile, a minor amount of variation in grades occurred (columns one and two, row two). Also, this group exhibited fairly good high school GPAs, as shown in the first row of the image. The overall compactness of the trend lines (excluding the end), especially for the second largest cluster (column two) reveals that the clustering metric closely aligns with

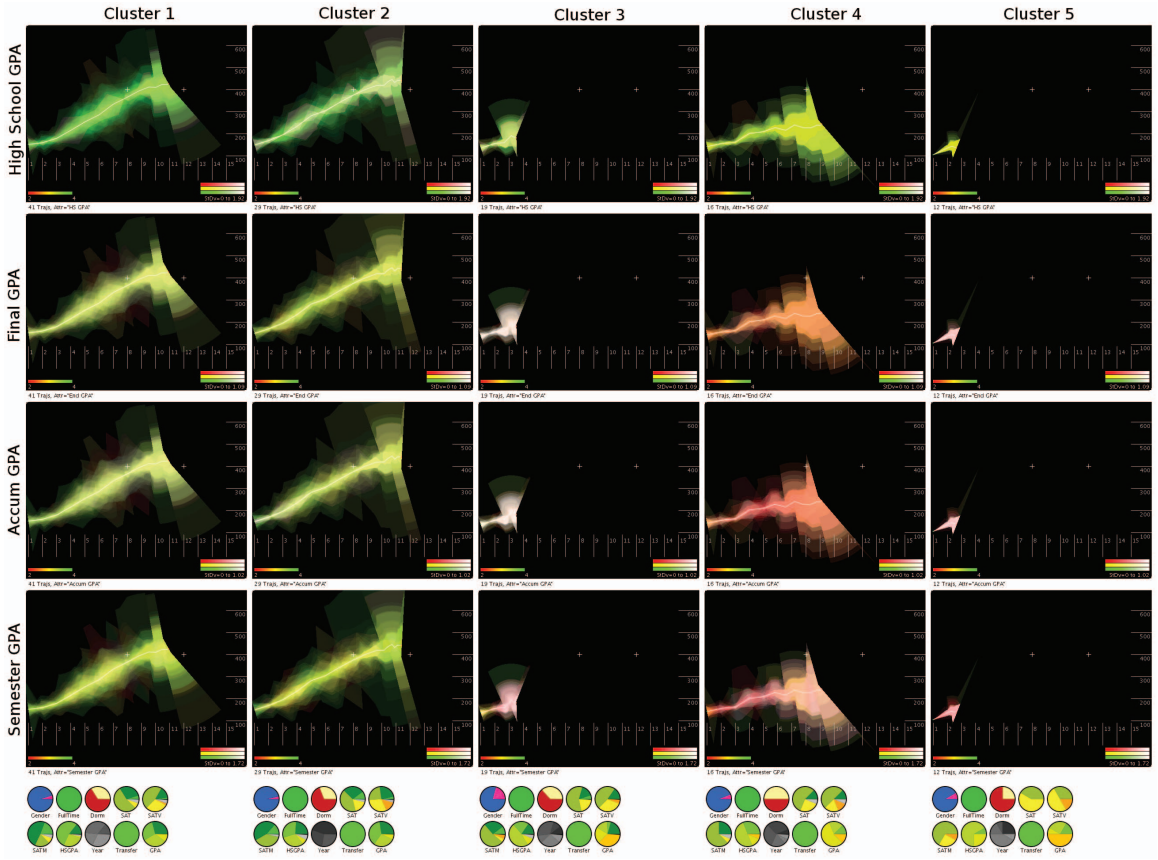

FIG. 5.21. Computer engineering top clusters

the visualization representation—e.g., that the students in the cluster took similar courses in the same semesters. The ends of the trajectories do show an expansion, probably due to a lower sampling interval as students graduated. This may also be due to the earlier noted trend of students taking low numbered graduation requirements. The lower portions at the end of the renderings exhibit the over-representation artifact of the composition process—the areas add no additional value but draws the eye to the pattern.

The fourth largest cluster (column three in Figure 5.21) struggled more to progress to the higher-numbered courses. The average trajectory ended above the 200-level series but not quite at the 300-level. The students also exhibited lower GPAs in their accumulated college GPAs (i.e., the red appearance for the fixed color scale in the second row) but had good grades in high school (row one). The semester GPA attribute also exhibited an interesting pattern (column four, row four). In particular, there was less variability in the student grades early in their coursework, as shown by the stronger, more saturated red values. Later in their academic careers, the variability increased indicating that a portion of students began to improve in their grades. One distinct possibility is that the "natural" computer engineering majors improved their semester GPAs once they were taking only their major courses—although this is unlikely due to the inability to progress to higher numbered courses.

The clustering of computer science majors exhibited more interesting trends than the computer engineering students (Figure 5.22). The largest cluster (first column) represents a majority of transfer students. The average trajectory begins higher (slightly above the 200-level) and quickly progresses to the 400-level series in under eight semesters. The cluster does have significant variability in the high school GPAs (row one) but that's due to transfer students not having high school GPAs in their records. As with the computer engineering students, the end of the trajectory exhibits large variability in the course numbers.

The second largest cluster for computer scientists (Figure 5.22, column two) shows

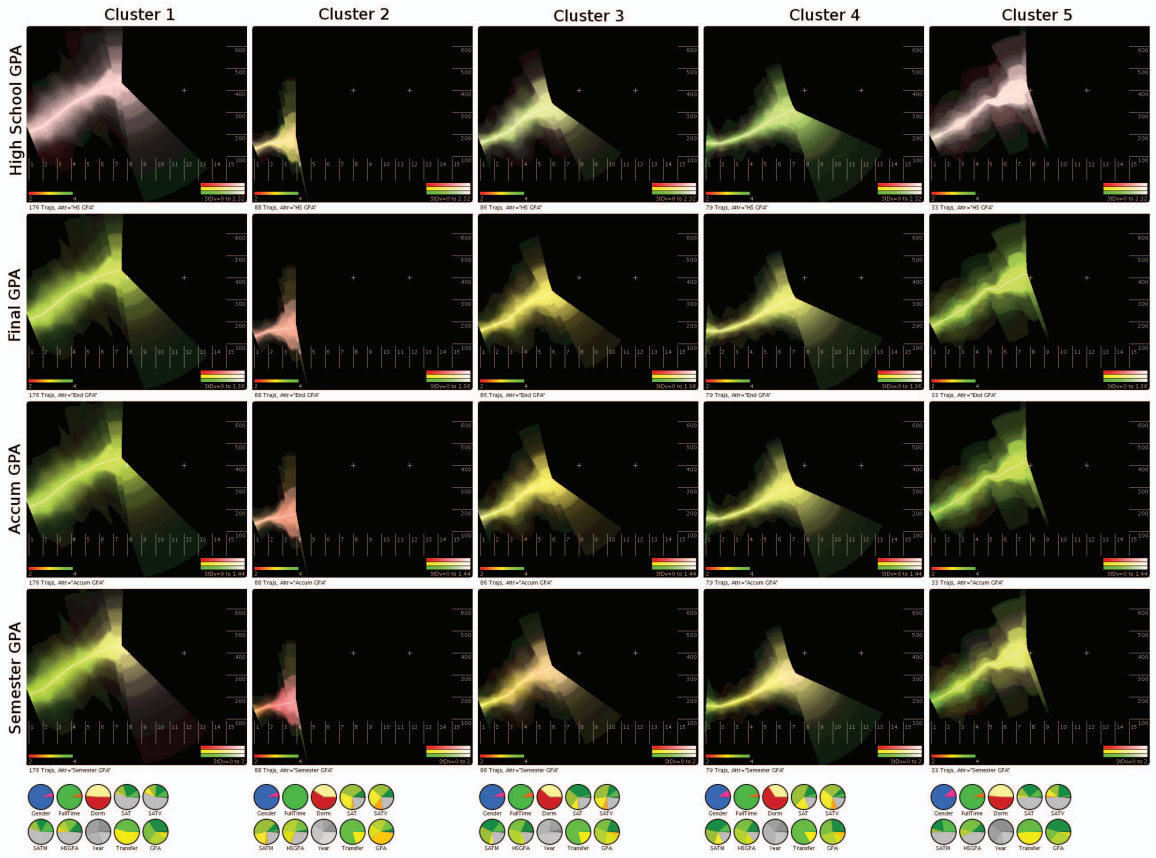

FIG. 5.22. Computer science top clusters

students failed to complete the introductory courses. Judging by their lack of progress, very few of these students made it past the fourth semester. Their high school GPAs (row one) shows some variability but, in general, edging towards a "B".

The third and fourth clusters (columns three and four) show two different progressions that end in the same way—around the sixth or seventh semester without achieving the necessary 400-level series courses. The primary difference is that cluster three (column three) progresses more quickly at the beginning while cluster four (column four) has a slower uptake at the beginning. Judging by their high school GPAs (row one), cluster four had higher GPAs in high school which may account for the higher variability in cluster three's beginning distribution (width).

The fifth largest cluster (column five) also represents an even number of transfer students. The cluster is more compact than the other transfer student cluster (column one). However, the cluster starts at a slightly lower course number and ends one semester later than the largest cluster in the first column. Based on the final GPA (row two), the fifth cluster performed slightly better (more saturated green) than the first cluster—possibly because the students completed some introductory or prerequisite courses at the university.

While clustering students according to the order and semester for courses taken produced interesting anomalies and discoveries, it was very difficult to narrow in on the reason. Patterns did emerge in the average trajectory, semester GPAs, and variability in course series, but the underlying issue was elusive due to the amount of concurrent data and the interdependencies between the semesters for individual students. Binning the students by their course-grade combinations was easier and more intuitive to understand than attempting to cluster them by their course ordering.

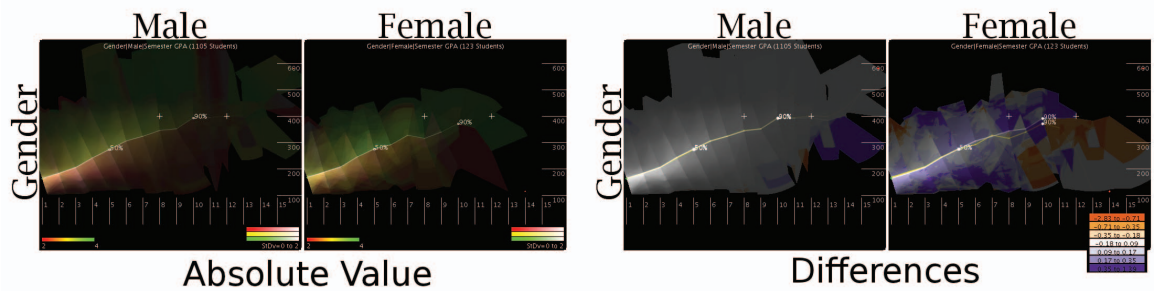

FIG. 5.23. Computer science gender differences. The left-hand renderings show the results of the semester grades while the right-hand tiles show the difference images. Note that computer science females had higher grades even though they had slightly lower progressions.

5.5.3 Differencing Student Groupings

While the small multiples approach reveals differences when the trajectories vary significantly, understanding subtle differences between the student groups is difficult. By modifying the approach to show the differences between two compositions, these variations were able to be more easily detected. To modify the approach, student subsets (e.g., gender, gateway courses) are compared against the entire student population. The composition process remains mostly the same—first, the entire student population is rendered and then the subset is rendered using the same frame of reference (i.e., same average trajectory and parametric mapping) separately. This process results in two separate compositions that can be compared pixel-by-pixel.

To render the difference composition, a Brewer colorscale (Harrower & Brewer 2003) based on a divergent color scheme of seven values is used. The lower three values show negative differences while the upper three show positive ones. The middle value is used to denote no change. The composition itself still implements the density functionality to provide "dimming" in areas where fewer students existed.

The results are revealing especially when subtle differences exist. Figure 5.23 shows

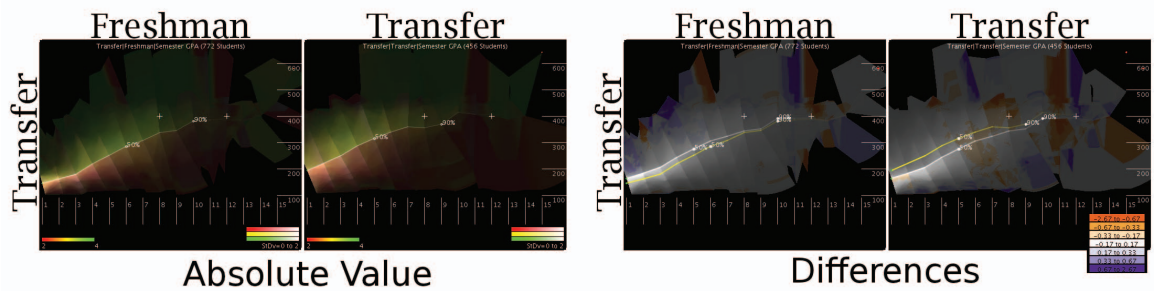

FIG. 5.24. Computer science transfer comparison. The left-hand renderings show the results of the semester grades while the right-hand tiles show the difference images. Note that grades did not significantly vary but the average trajectory for transfer students started at higher course numbers—due to completed pre-requisites.

both results—the left-hand tiles show the male and female trajectories for the computer science data set using just the composition technique. While the results look very similar, the difference images on the right-hand side reveal subtle variations in the semester grades. While the males had very little differences from the overall student set, the females had slightly higher grades on both sides of the average trajectory. To draw out the minute differences, a graduated, discrete binning range is used. The largest bin covers the maximum difference to a quarter of the maximum value. The second largest bin covers from a quarter to an eighth of the maximum difference, and the smallest bin covers the eighth to the sixteenth. Therefore, the middle bin ranges from the negative sixteenth to the positive sixteenth. Mathematically, if $floor$ if the minimum value and $ceil$ is the maximum value, the bins are arranged as follows: [$floor$,$floor/4$], [$floor/4$,$floor/8$], [$floor/8$,$floor/16$], [$floor/16$,$ceil/16$], [$ceil/16$,$ceil/8$], [$ceil/8$,$ceil/4$], and [$ceil/4$,$ceil$]. To show the difference in the trajectory, the population trajectory is depicted in white while the subset trajectory is shown in yellow. For the male / female dataset, these are almost indistinguishable from the overall population. However, by comparing the overall grades using the differencing approach, the slight improvement of the female students becomes recognizable.

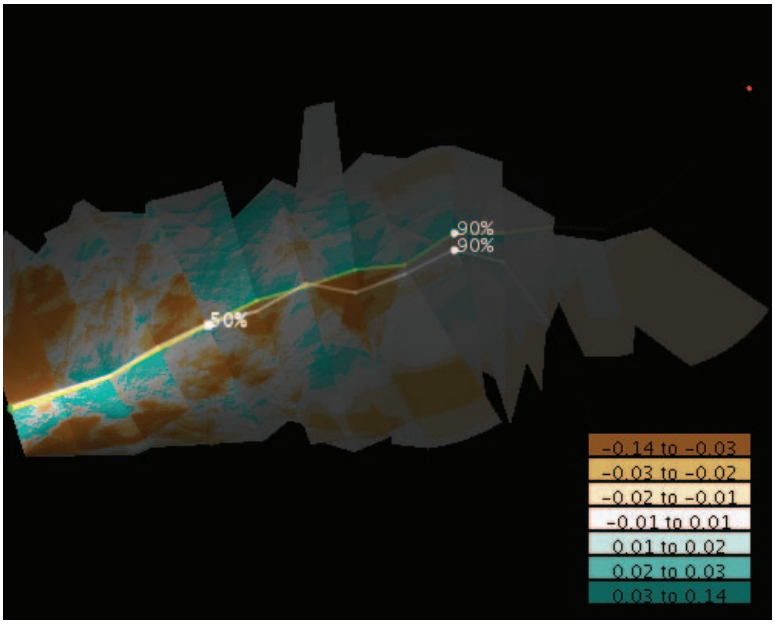

FIG. 5.25. Computer science gender density differences. In addition to showing differences in the application data (e.g., GPA), the difference composition can also denote differences in density.

However, not all demographic comparisons show significant variations in the semester grades. Figure 5.24 depicts the results for transfer students. As with the gender example, the left-hand tiles show the composition while the right-hand images render the difference image. Unlike the gender example, both subsets (i.e., non-transfer/freshmen and transfer) have vary little variation in semester grades throughout their academic careers. However, by showing the average trajectory for both the subset in yellow and the entire population in white, the increased starting position for transfer students becomes apparent. The increased position is due to transfer students having already completed the lower course numbers at their starting college. Another takeaway lesson from this example (and previously mentioned) is that transfer students complete the program in roughly the same time frame as their freshmen counterparts.

Since the difference composition has access to the underlying data for both compo-

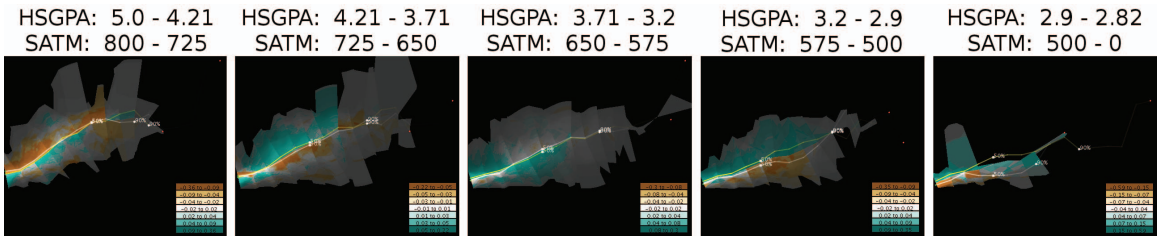

FIG. 5.26. Computer science high school GPA versus SAT Math density differences.

sitions, it can also calculate the differences in the density. Figure 5.25 shows the density difference for gender. In this figure, the tan and brown colors correspond to higher densities of female students. As one can see, female students were slightly more prevalent for higher course numbers early (i.e., semester one and two) while males were more prevalent in later semesters (i.e., semester five and onward). As with the previous difference compositions, the bins are graduated to maximize the contrast for slight differences. Furthermore, both data sets were normalized to percentiles—e.g., the difference compares the percent of male students to the percent of female students and not the absolute numbers.

Figure 5.26 shows a more interesting density comparison. In this figure, the composition is comparing the densities between high school GPA and SAT Math students within their corresponding bins. In this figure, browns and tans represent higher densities for SAT Math; blues indicate higher high school GPA densities. For the most successful (i.e., left-most) bin, SAT Math had an edge in overall course number progression throughout their academic careers. However, that trend is reversed for the second bin—the high school GPA students took higher-numbered courses, again, throughout their studies.

5.6 Domain Expert Input and Experiences

I solicited input from several educational domain experts during the design and realization phases, including two Undergraduate Program Directors, one Accreditation Coor-

dinator, and one scholarship program director. Their input is reflected in several aspects of the current system, including the capability to show disaggregated groups, the overlay of one group on another, the orientation of the images, and features to support drill-down to specific clusters and trajectories.

These domain experts have also engaged in extended interaction sessions with the completed prototype. While multiple known trends were confirmed during these sessions, unexpected relationships were also discovered in the data. For instance, the importance of a student really doing well in the first computer science gateway course over the second was an unexpected find in the data. This could indicate either that the concepts in the first course were much more critical to success than the second course or that. This insight into the critical importance of the first course could be used to identify interventions to help students who don't start off as strongly as they should.

In addition to determining which courses most prominently influenced a students career, the demographic analysis also revealed additional insights to educators—most notably, the high school GPA comparison with SAT Math. The results were contrary to traditional beliefs that high school GPA is more critical to success than standardized test scores. By identifying these anomalies, educators can delve into the underlying details to understand the nature and cause of the trend. This insight could be used to fine-tune admission standards or to identify students who would benefit from early intervention.

5.7 Issues and Limitations

Multiple issues and limitations became apparent as the results of the composition process were analyzed. The primary limitation was identifying how to group the students to reveal trends of interest. While a small multiples approach was used to organize the visualizations according to course-grade combinations, this process required manual analysis

to identify differences and important anomalies. Furthermore, the clustering of students by their course history did not reveal as many insights as hoped. A possible solution would be to cluster the small multiples results. This would group the first stage aggregate trajectory results (i.e., course-history combinations) into partitions to better identify similar average trajectories and anomalies. In effect, this would decrease the manual review and increase the probability that important features were detected.

Transfer students also presented an issue. While the start of their trajectories began at the first semester index, calculating their initial offset might have yielded more coherent compositions. For example, because all student trajectories began at the first semester (spatially), transfer students tended to pull the average trajectory higher in the beginning. If, however, a method could be identified to place them in their true offset, then the composition would not have been skewed by the transfer-student bias.

The last issue was the visual banding caused by the enforced temporal rigidity. This artifact did not occur in the first use case and was the result of students taking widely varying course numbers towards the end of their academic careers. These artifacts were also related to the reduced sample size towards the end of the student career—e.g., after the fiftieth percentile marking, the average trajectory was much more likely to fluctuate, due to the small number of students contributing to the calculation.

Chapter 6

LESSONS LEARNED

This chapter addresses how to apply the approach to a new dataset. Specifically, the following questions will be addressed:

- Is a trajectory data set applicable to this approach?
- What preprocessing steps need to be applied to the data set?
- How should the clustering distance metric and trajectory averaging algorithm be defined?
- How can one tell if the trajectories were successfully clustered?

6.1 Approach Applicability

As discussed in the introduction and use-case chapters, the approach is designed for *entity-based* trajectory data. Specifically, trajectories where analysis of the entities is needed. Analytic results and conclusions should be focused on how the entity changes over time as a result of the environment or as the entity evolves over time. The two documented use cases exemplify these objectives. RoboCup results are used to understand the state of the soccer match and how the offense is moving the ball across the field (i.e., the

environment). Student course history data is solely focused on the student's progression through academia—that is, how the entity evolves over time. In the latter case, the environment is simplified but attempts to model the higher dimensional knowledge and skill set acquisition.

The dataset should also contain a large number of trajectories with a firm expectation that the data lends itself to clustering. As shown in the test data section, numerous, cohesive trajectories produce smoother results that leverage the composition's approach—that is, aggregate analysis techniques yield more insight than a more traditional rendering of the individual trajectories.

6.2 Preprocessing Steps

Preprocessing is often necessary to segment the trajectory dataset into meaningful sub-trajectories. For instance, the first use case required that the RoboCup match information be separated along discrete team possessions. In a similar fashion, GPS data is often "pre-clustered" to identify interesting locations (Palma *et al.* 2008) (Marketos 2009) (Lee *et al.* 2008) or to group similar entities together (Jeung *et al.* 2008) (Li *et al.* 2007). To a certain degree, the second use case exemplifies this step—that is, the trajectories were grouped together not by their similarity, but by the individual grade a student made in a particular course.

At one extreme, no preprocessing is required—for example, the student history data. On the other hand, every trajectory is broken into discrete components for further processing (Lee *et al.* 2008) (Lee, Han, & Whang 2007). Most applications, however, would need to logically segment the trajectories into sequences that naturally indicate the start and stop of an entity's behavior. For example, Lee et al. 2008 and Palma et al. 2008 identify interesting locations in large trajectory populations. To preprocess these data sets,

the individual trajectories between interesting locations would be extracted for further clustering. The resulting analytics would be to determine the most common paths between the identified locations.

6.3 Distance Metric and Averaging Algorithm

The most relevant detail for the distance metric is to compare the "best" corresponding parts of the trajectories. While subjective, several measures are of importance. First, if common begin and end points can be identified, those should be used to constrain the trajectories. For instance, if interesting locations can be identified from other approaches, the paths between these locations would be applicable to my approach. Natural changes of state are also analogous—in the RoboCup example, changes of possession marked natural boundaries for the data.

Second, If common resynchronization points exist, they should be used to align the trajectory point comparison and averaging algorithm. For example, in the student history data, semesters formed a common alignment point. By resyncrhonizing at each location, the underlying visualization distance measurement was correctly modeled. Early attempts to use a length-based parameterization resulted in a "funnel" in the composition. The funnel was a direct result of the accumulated error from the lack of synchronicity across the student set.

Furthermore, the distance metric should appropriately handle the temporal nature of the trajectory data. For instance, if the closest point comparison is used, the temporal ordering of the trajectory may be distorted or lost during the comparison (and resulting visualization). The parametric form of the trajectory is a natural representation that correctly models the temporal nature of the data—e.g., by monotonically increasing throughout the comparison. In the use case data sets, both length-based and duration-based parametric

representations were used. The latter (duration-based) was chosen for the second use case because the trajectories needed to be "re-synced" at semester boundaries. However, for the RoboCup data, length-based was more appropriate since time-varying plays would distort the overall results.

6.4 Cluster Results

Lastly, the most effective way to verify and validate the results is to examine the visual compositions. These provide a quick, intuitive check to see if the visual metaphor accurately represents the underlying data. If possible, a well-known, previously identified pattern should be identified in the composition. Common patterns provide a starting point to verify known trends and to begin to identify "one-off" discoveries in the data. Furthermore, these "knowns" build confidence in the approach and sense-making abilities of both the techniques and the analysts.

In addition to validating known patterns, the visualization process can quickly identify deficiencies in the distance metric. As discussed earlier, the length-based parametric version failed to effectively capture the corresponding locations for the student history data. Two issues were the underlying cause. First, students who attended for a different number of semesters caused incorrect points to compared to one another. For example, if one student attended for 13 semesters, then that student's 13th semester would be compared to the end of every student's history—whether it be the 11th, 6th, or 18th. To resolve this issue, the distance metric and composition process was modified to align to the semester index.

The second issue was that the length-based parametric representation accumulated errors over the students' histories. While minor, the error accumulated because students that had steep slopes in their trajectories had slightly longer trajectory segments. This cause the distance metric to slowly increase over the course of the student history. As

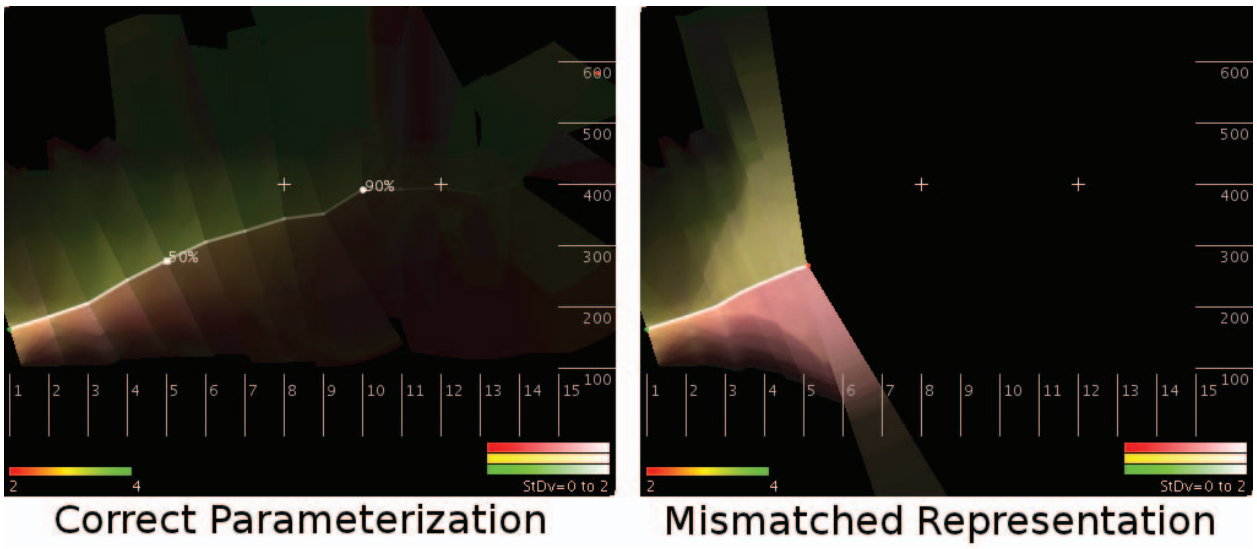

FIG. 6.1. Parameterization mismatch. When trajectories are incorrectly represented, the resulting visualization (right) shows misleading trends as compared to the correct representation (left).

described earlier, the solution was to parameterize over the duration of the trajectory forcing resynchronizations of the trajectories across time.

Figure 6.1 depicts both of the parameterization issues with the student data. In the figure, several issues are immediately apparent with the right-hand tile. First, the average trajectory ends abruptly slightly after the fifth semester. This is a result of the distance and averaging metric not incorporating the varying number of student semesters. Second, the composition shows an exponentially increasing distance by the end of the trajectory. This is due to the distance metric rapidly increasing because the average trajectory is too short. This feature also occurs when the length-based parametric representation is used for the data as a result of the accumulated error.

Chapter 7

FUTURE WORK

Areas of future work fall into three categories: (1) overcoming visualization limitations with the composition process, (2) developing additional automation to refine results for human analysis, and (3) combining the composition process with traditional visualization techniques to increase interactivity and overcome composition limitations.

Several limitations exist with the visualization and composition process. First, even though a level set is used to deconflict the trajectory contributions in areas of over- and underrepresentation, the composite visualization still "sweeps" duplicative values in areas of overrepresentation while abbreviating values in underrepresented sections. A post-processing step to the level set calculation could be added to ensure that each trajectory attribute value received an equal amount of image real-estate. The overall risk of such an approach would be the dispersion of the correct spatial location for each parametric trajectory value—e.g., even though values are duplicated in overrepresented, swept areas, that is still the most appropriate location for them to contribute. Second, the technique does not work with overlapping trajectory segments. For example, if the average trajectory had a loop or crossed over a previous section, the level set process would (incorrectly) overwrite sections of the data. While multiple sub-visualizations could be used in areas of contention, that approach would be an imperfect solution for the problem.

As alluded to in the issues and limitations for the student course-history use case, techniques to prioritize which groups of trajectories (and their related compositions) need to be developed to more effectively handle the volume of data. This is especially true in cases where the clustering results in an overwhelming number of partitions or when there are many application-specific attributes. In summary, either a clusterer for clusters or a method to analyze each resulting composition would significantly enhance a user's ability to handle the volume of data and the combinatorics necessary to try each setting.

The last category for future work is combining the approach with other visualization techniques and interactive paradigms. While the approach does provide a method to combine thousands of trajectories while retaining information about the application-specific attribute and overall spatial characterization of the group, traditional visualization techniques can provide additional insights. Most notably, brushing and linked visualizations (Keim 2002) would enable the integration of traditional techniques with the approach.

Chapter 8

CONCLUSION

With this research, I addressed a difficult data analysis problem that has multiple real-world applications. By combining machine automation with intuitive and insightful visualization techniques, new light was shed on trajectory populations not currently addressed by traditional visualization techniques. In order to show the value, the technique was applied to two application domains—artificial intelligence agents and student course-history data. The results of the analysis led to new discoveries about both datasets. However, several shortcomings were identified with my approach that require follow-on research to address.

Specifically, I developed a straightforward approach to clustering two-dimensional trajectory populations. By basing the metric on a parametric model of a trajectory, the metric correctly and efficiently handled trajectories of varying fidelity. Furthermore, the parametric values were tailored for specific applications—in the use cases, length-based parameterization of the trajectory data was effectively used for a spatial application while a time-based version for the second use case was required. Furthermore, through the use cases, I justified the decision process for the underlying model by providing detailed results with real-world conclusions and recommendations.

Additionally, I created an aggregate visualization technique that intuitively captures the underlying trajectory population. This technique visually captures the shape of the av-

erage trajectory, spatial variability of the population, and the mean and standard deviation of application-specific trajectory attributes. By effectively combining all of these characteristics, viewers can intuitively understand the rendered trajectory population and make analytical conclusions. Additionally, I integrated and implemented two separate rendering schemes that discerned different aspects of the underlying attributes. The visualization technique complemented the clustering approach because it enables a viewer to determine if clustering produced accurate results.

In order to demonstrate the effectiveness of my approach, the techniques were applied to two different applications: RoboCup soccer and student course-history data. These applications showed the general applicability of the approach since the representations and goals significantly differed. By applying traditional visualization techniques to the first use case, I demonstrated limitations for understanding the underlying data using these types of techniques. However, by applying the clustering and visualization techniques, I was able to provide novel and intuitive results for identifying (through clustering) and analyzing (through visualization) common plays within the tournament. By also including RoboCup attributes in the rendering, I provided additional details to the viewer to better understand the individual trajectory data.

Through the analysis of student course-history data, I tailored the visualization approach to match the analytical objectives. By choosing to group the data by its natural binning, I effectively determined which courses and topics have the largest impact on a student's academic career. Although the clustering algorithm was applied, the results were not as meaningful as the aforementioned natural bins. Furthermore, I supplemented the approach to compare the differences between individual student groupings and the overall student population. This technique enabled viewers to determine and understand subtle differences in subsets of the population.

REFERENCES

[1] Andrienko, G., and Andrienko, N. 2008. Spatio-temporal aggregation for visual analysis of movements. *IEEE VAST* 51–58.

[2] Berkhin, P. 2002. Survey of clustering data mining techniques. Technical report;, Accrue Software.

[3] Bowman, D.; Koller, D.; and Hodges, L. 1998. A methodology for the evaluation of travel techniques for immersive virtual environments. *Virtual Reality: Journal of the Virtual Reality Society* 3(2):120–131.

[4] Brewer, C. 1999. Color use guidelines for data representation. In *Proceedings of the Section on Statistical Graphics*, 55–60.

[5] Chittaro, L.; Ranon, R.; and Ieronutti, L. 2006. Vu-flow: A visualization tool for analyzing navigation in virtual environments. *IEEE Transactions on Visualization and Computer Graphics* 12(6):1475–1485.

[6] Chlan, E., and Rheingans, P. 2005. Multivariate glyphs for multi-object clusters. *Information Visualization, 2005. INFOVIS 2005. IEEE Symposium on* 141–148.

[7] Edwards, S. H.; Pérez-Quiñones, M. A.; Phillips, M.; and RajKumar, J. 2006. Graphing performance on programming assignments to improve student understanding. 6.

[8] Ferral, H. 2004. *Clustering students by their subject choices in the Learning Curves project*. Research report : New Zealand Council for Educational Research. NZCER.

[9] Hagh-Shenas, H.; Kim, S.; Interrante, V.; and Healey, C. 2007. Weaving versus blending: A quantitative assessment of the information carrying capacities of two alternative

methods for conveying multivariate data with color. *IEEE Transactions on Visualization and Computer Graphics* 13(6):1270–1277.

[10] Harrower, M. A., and Brewer, C. A. 2003. **ColorBrewer.org**: An online tool for selecting color schemes for maps. *The Cartographic Journal* 40:27–37.

[11] Hershberger, J., and Snoeyink, J. 1992. Speeding up the Douglas-Peucker line-simplification algorithm. In *Proceedings of the 5th International Symposium on Spatial Data Handling*, 134–143.

[12] Holten, D., and van Wijk, J. J. 2009. Force-directed edge bundling for graph visualization. *Comput. Graph. Forum* 28(3):983–990.

[13] Jeung, H.; Yiu, M. L.; Zhou, X.; Jensen, C. S.; and Shen, H. T. 2008. Discovery of convoys in trajectory databases. *Proceedings of the VLDB Endowment* 1:1068–1080.

[14] Johansson, J.; Ljung, P.; Jern, M.; and Cooper, M. 2005. Revealing structure within clustered parallel coordinates displays. In *Information Visualization, 2005. INFOVIS 2005. IEEE Symposium on*, 125 – 132.

[15] Kapler, T., and Wright, W. 2004. Geotime information visualization. In *INFOVIS '04: Proceedings of the IEEE Symposium on Information Visualization (INFOVIS'04)*, 25–32. Washington, DC, USA: IEEE Computer Society.

[16] Keim, D. A. 2002. Information visualization and visual data mining. *IEEE Transactions on Visualization and Computer Graphics* 8:1–8.

[17] Kohonen, T., ed. 1997. *Self-Organizing Maps*. Secaucus, NJ, USA: Springer-Verlag New York, Inc.

[18] Laramee, R. S.; Hauser, H.; Doleisch, H.; Vrolijk, B.; Post, F. H.; and Weiskopf, D. 2004. The state of the art in flow visualization: Dense and texture-based techniques. *Computer Graphics Forum* 23(2):203–222.

[19] Lee, J.-G.; Han, J.; Li, X.; and Gonzalez, H. 2008. Traclass: trajectory classification using hierarchical region-based and trajectory-based clustering. *Proceedings of the VLDB Endowment* 1:1081–1094.

[20] Lee, J.-G.; Han, J.; and Whang, K.-Y. 2007. Trajectory clustering: a partition-and-group framework. In *Proceedings of the 2007 ACM SIGMOD International Conference on Management of Data*, SIGMOD '07, 593–604. New York, NY, USA: ACM.

[21] Li, X.; Han, J.; Lee, J.-G.; and Gonzalez, H. 2007. Traffic density-based discovery of hot routes in road networks. In Papadias, D.; Zhang, D.; and Kollios, G., eds., *SSTD*, volume 4605 of *Lecture Notes in Computer Science*, 441–459. Springer.

[22] Ma, S., and Hellerstein, J. L. 1999. Ordering categorical data to improve visualization. In *Proceedings of the IEEE Symposium on Information Visualization*, 15–17.

[23] Marketos, G. 2009. *Data Warehousing and Mining Techniques for Moving Object Databases*. PhD in Informatics, University of Piraeus.

[24] Mazza, R., and Dimitrova, V. 2004. Visualising student tracking data to support instructors in web-based distance education. In *Proceedings of the 13th International World Wide Web Conference on Alternate Track Papers & Posters*, WWW Alt. '04, 154–161. New York, NY, USA: ACM.

[25] Palma, A. T.; Bogorny, V.; Kuijpers, B.; and Alvares, L. O. 2008. A clustering-based approach for discovering interesting places in trajectories. In *Proceedings of the 2008*

ACM Symposium on Applied Computing, SAC '08, 863–868. New York, NY, USA: ACM.

[26] Quevedo, J.; Montanes, E.; Ranilla, J.; and Bahamonde, A. 2009. Automatic choice of topics for seminars by clustering students according to their profile. *World Academy of Science, Engineering and Technology* 597–602.

[27] Raines, T.; Tambe, M.; and Marsella, S. 1999. Towards automated team analysis: A machine learning approach. In *Proceedings of the RoboCup Workshop at the International Joint Conference on Artificial Intelligence (IJCAI'99)*.

[28] Rheingans, P. L. 2000. Task-based color scale design. In W. R. Oliver., ed., *Society of Photo-Optical Instrumentation Engineers (SPIE) Conference Series*, volume 3905 of *Society of Photo-Optical Instrumentation Engineers (SPIE) Conference Series*, 35–43.

[29] Riley, P., and Veloso, M. M. 2000. Behavior acquisition and classification: A case study in robotic soccer. In *AAAI/IAAI*, 1092.

[30] Rogers, S.; Langley, P.; and Wilson, C. 1999. Mining GPS data to augment road models. In *KDD '99: Proceedings of the fifth ACM SIGKDD International Conference on Knowledge discovery and data mining*, 104–113. New York, NY, USA: ACM.

[31] Rueda, U.; Larraaga, M.; Kerejeta, M.; Elorriaga, J.; and Arruarte, A. 2005. Visualizing student data in a real teaching context by means of concept maps. In *Proceedings of the 5th International Conference on Knowledge Management (I-Know 2005)*, 561–569.

[32] Salmani, V.; Fard, A.; and Naghibzadeh, M. 2006. A fuzzy two-phase decision making approach for simulated soccer agent. *IEEE International Conference on Engineering of Intelligent Systems* 1–6.

[33] Sethian, J. A. 1999. *Level Set Methods and Fast Marching Methods: Evolving Interfaces in Computational Geometry, Fluid Mechanics, Computer Vision, and Materials Science*. Cambridge University Press, 2 edition.

[34] Silva, S.; Madeira, J.; and Santos, B. 2007. There is more to color scales than meets the eye: A review on the use of color in visualization. In *Information Visualization, 2007. IV '07. 11th International Conference*, 943 –950.

[35] Thawonmas, R., and Iizuka, K. 2008. Visualization of online-game players based on their action behaviors. *International Journal of Computer Games Technology* 2008(5):1–9.

[36] Tufte, E. R. 1990. *Envisioning information*. Graphic Press.

[37] Verhein, F. 2009. Mining complex spatio-temporal sequence patterns. In *2009 SIAM International Conference on Data Mining (SDM)*, 605–616.

[38] Visser, U., and Welande, H.-G. 2002. Using online learning to analyze the opponents behavior. In *RoboCup 2002: Robot Soccer World Cup IV*, volume 2, 78–93. Springer Berlin / Heidelberg.

[39] Whiteson, S.; Kohl, N.; Miikkulainen, R.; and Stone, P. 2005. Evolving keepaway soccer players through task decomposition. *Machine Learning* 59(1):5–30.

[40] Willems, N.; van de Wetering, H.; and van Wijk, J. J. 2009. Visualization of vessel movements. *Computer Graphics Forum* 28(3):959–966.

[41] Wortman, D., and Rheingans, P. 2007. Visualizing trends in student performance across computer science courses. *Proceedings of the 38th Annual ACM Technical Symposium on Computer Science Education* 39:430–434.

[42] Yang, J.; Ward, M. O.; and Rundensteiner, E. A. 2003. Interactive hierarchical displays: a general framework for visualization and exploration of large multivariate data sets. *Computers and Graphics* 27(2):265 – 283.

[43] Zafarani, R., and Yazdchi, M. R. 2007. A novel action selection architecture in soccer simulation environment using neuro-fuzzy and bidirectional neural networks. *International Journal of Advanced Robotic Systems* 4(1):93–101.

[44] Zanbaka, C. A.; Lok, B. C.; Babu, S. V.; Ulinski, A. C.; and Hodges, L. F. 2005. Comparison of path visualizations and cognitive measures relative to travel technique in a virtual environment. *IEEE Transactions on Visualization and Computer Graphics* 11(6):694–705.