Dynamic Visualization of Battle Simulations

Paul R. Cohen, James A. Davis and John L. Warwick

Computer Science Technical Report 00-16

Experimental Knowledge Systems Laboratory
Computer Science Department
140 Governors Drive
University of Massachusetts
Amherst, MA 01003-4610

Abstract

We present a case study of visualization in understanding encounters between multiple agents in an adversarial environment. The information visualized consists of time series of attributes and relations such as mass, velocity and distance, which we preprocess with a Bayesian clustering algorithm. We differentiate between the encounters based on their outcomes, and generate two and three-dimensional maps that can be used to determine good courses of action from different points in the agents' environments.

Dynamic Visualization of Battle Simulations

Paul R. Cohen, James A. Davis, and John L. Warwick

Department of Computer Science
University of Massachusetts, Amherst, MA 01003

ABSTRACT

We present a case study of visualization in understanding encounters between multiple agents in an adversarial environment. The information visualized consists of time series of attributes and relations such as mass, velocity and distance, which we preprocess with a Bayesian clustering algorithm. We differentiate between the encounters based on their outcomes, and generate two and three-dimensional *maps* that can be used to determine good courses of action from different points in the agents' environments.

Keywords: Bayesian Clustering, Dynamic Maps

1. INTRODUCTION

This is a case study of visualizations of encounters between adversarial agents in simulated war games. We visualize time series data which captures the dynamics of the encounters. As a preprocessing step, we use a Bayesian clustering algorithm to partition large numbers of encounters of data into clusters which share important dynamical similarities. Our visualizations allow an observer to analyze past events leading to the current state, to make predictions about the future course of events, and to make plans based on an abstract representation of the war game environment.

The simulation environment used in this work is the Abstract Force Simulator (AFS), developed at the University of Massachusetts. In AFS a set of abstract agents, called blobs, are described by a small set of physical features, including mass and velocity. A blob is an abstract unit, it could be an army, a planet, a corporation, or a political entity. Every blob has a small set of primitive actions it can perform, primarily move and apply-force, which can be composed into more complex actions and coordinated group actions with multiple blobs. For this work, blobs in AFS represent and behave as battalions and regiments of artillery, mechanized artillery and infantry, as well as air units. The actions available to the blobs are familiar tactics, including attack, envelop, fix, block, follow-and-assume, defend, and defend-in-place. The blobs are controlled by human players or by an AI planning system.

As with many visualization tasks, we must reduce enormous amounts of data from AFS to key aspects and attributes that illustrate critical factors in encounters. Our visualizations focus on the dynamics of encounters. In previous work, we tracked data representing the physical characteristics of the agents and their locations (i.e. mass, velocity, relative distance between agents, and agent distances from fixed points). We graphed pairs and triples of these time series of values with respect to each other as well as with respect to time. Overlays of these time series for dozens or hundreds of encounters are called *dynamic maps*, in which each line represents the time series of a single encounter. We colored these lines according to the winner of the encounter. By placing the trajectories of similar encounters on the same map, the observer can clearly see any bifurcation points, or points at which encounters cease following the same path through the state space. Often, a bifurcation indicates a place in an encounter where a single decision or the value of a single agent's attribute affects the outcome of the encounter. It is straightforward to calculate the probability of winning an encounter at any given point in a dynamic map. We created a three-dimensional representation where the Z value assigned to a region represents the probability of a win (for one side or the other) if the encounter goes through the region. This representation is an abstraction of the dynamics of the encounter in which the elevation and slope of the surface indicates dangerous regions, safe regions, and points at which a small change in the state space can result in much greater (or lesser) probability of success. This feature allows plans to be developed in which abstract properties of teams and agents are adjusted to move to a desired region.

When a map includes time series for many encounters, it can be difficult to read. One reason is that the encounters might have very different dynamics, which, when overlaid in a map, gives a confusing muddle of lines. As a preprocessing step, we use a Bayesian clustering algorithm to partition the time series of encounters based on similarities in their dynamics. This produce subsets of data that can be more readily graphed and analyzed. Ongoing work in this area involves automating the search for nonlinearities that correspond to regions of opportunity and danger in the maps.

In the remainder of this paper we describe the Abstract Force Simulator in detail, as well as the domain of war games. Next, we discuss the Bayesian clustering algorithm. In the third section, we explain and illustrate how visualizations are generated. We conclude with a discussion of results and future work in this area.

2. ABSTRACT FORCE SIMULATOR

The Abstract Force Simulator[1,2,3] (AFS) was designed to model a range of physical domains. Physical processes, military engagements[4], and games such as billiards, are all about agents moving and applying force to one another[5]. Even the somewhat abstract realm of diplomacy can be viewed in these terms. A government may try to apply pressure to another for some purpose, or may try to contain a crisis before it spreads. Furthermore, it appears that there are common sets of terms upon which all the above processes operate. Some examples include move, push, reduce, contain, block, and surround. Collectively, these terms are referred to as physical schemas. If moving an army is conceptually no different than moving a robot, both of these processes can be represented with one move action in a simulator. We believe that people think and solve problems in terms of physical schemas. As an example, a person notices that their sink is leaking, and considers what

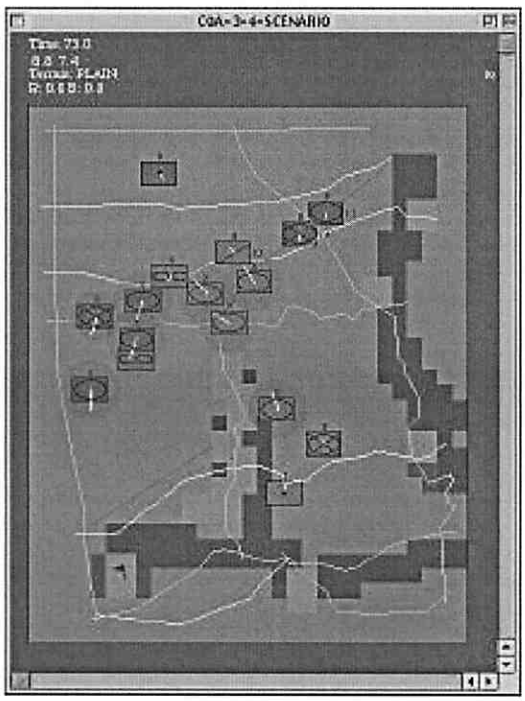

Figure 1. A screen from the Abstract Force Simulator, 73 time steps into a simulated course of action. Icons represent mechanized, infantry, artillery, and air cavalry units. Colored areas represent terrain: Light brown is open terrain; dark brown is hilly terrain; blue is a lake; green, forest; and gray, towns. Lines represent roads and boundaries of areas of responsibility.

to do about it. One possible solution is to plug the holes. This is not significantly different than the problem that faces a military decision maker when hostile forces are moving across a mountain range. What should be done about it? If the decision maker understands that military forces can behave like water, and that the holes correspond to passes in the mountain range, they can prevent the hostile forces from getting through the range by blocking the passes.

AFS has been developed to handle exactly this kind of situation. AFS operates with a set of abstract agents, known as blobs, which are described by a small set of physical features, including mass, velocity, friction, radius, attack strength, and so on. A blob is an abstract unit; it can represent an army, a soldier, a planet, or a political entity. Every blob has a small set of primitive actions it can perform, primarily move and apply-force. All other schemas are built from these actions. Simply by changing the physics of the simulator (how mass is affected by a collision, what the friction is over a certain terrain, etc.) AFS can simulate everything from billiards to unit movements in a military domain.

In this work, we used the basic military domain package available with AFS. Blobs are differentiated by type: infantry, mechanized, artillery, and helicopter. Each type has a specific movement model and an attrition model that governs how opponent units can damage each other. Blobs are organized into a hierarchical structure, designed to mimic the structure of military organizations. We instrumented the simulator to supply us with a wide range of sensor values from each unit, and information about the team as a whole. We defined complex variables from this information, such as mass near the action, a measure of how well a team's forces have been brought to bear on the enemy; and effective mass, a measure of how well a team's force have chosen their targets, based on the terrain and unit types available.

3. VISUALIZING THE INFORMATION

At this point, the challenge is to represent visually data from the Abstract Force Simulator. The input consists of dozens or hundreds of trials, where each trial represents a single simulated battle. Each trial is a time series of variables and a result. The time series of a single variable, x, is just a line in a graph of two dimensions, one of which represents the value of x and the other which represents time. Multiple trials can be overlaid on a single graph. We call an overlay of multiple time series a *dynamic map*. Each trial is represented by a line, called a *trajectory*, in a map. If one colors trajectories by outcomes and lays a grid over the map, as in Figure 2a, then the proportion of trajectories of a given color in a given grid cell is an estimate of the probability of the corresponding outcome given that the trajectory enters that cell. More formally, let x and t be a state variable and time, respectively, and let the coordinate x,t define a cell of the map, of some fixed area, centered on the point x,t. Then the probability of an outcome, such as a win, is estimated as:

$$\Pr(\text{win}|x,t) = \frac{\text{number of winning trajectories through } x,t}{\text{total number of trajectories through } x,t}.$$

We can generate a three-dimensional map from coordinates x, t, Pr(win|x,t), as shown in Figure 2b. Every cell on the surface is a probability of a Red win for a trajectory that goes through the cell defined by x,t.

The landscape plot in Figure 2b shows a probabilistic map constructed in this way for 1000 trials in which a Blue unit attacks a Red flag. The blue spot on the surface represents a state of a game; its vertical altitude represents the probability that Red will win in this state. The arrow leaving the blue spot is the most common trajectory leaving that state. You can see that Red is most likely to lose.

The probabilistic map in Figure 2 is "craggy," not flat or smoothly rising or falling. Sharp slopes on dynamic maps correspond to what commanders call *decisive points*. These are points or regions where the probability of success changes relatively quickly. The idea of nonlinearity, particularly nonlinear dynamics, is slowly permeating tactical planning centers in the U.S. military, and can be seen as a natural successor to Clausewitz, who revolutionized tactics by viewing warfare as a branch of physics[4]. The surface in Figure 2b appeals to commanders because it provides them an abstract view of the dynamics of the battlefield, one in which dangerous areas for both sides are explicit. Commanders recognize that if they have maps like Figure 2b and their opponents do not, then they have an information advantage that they might exploit by "steering" their opponents into advantageous areas of the map.

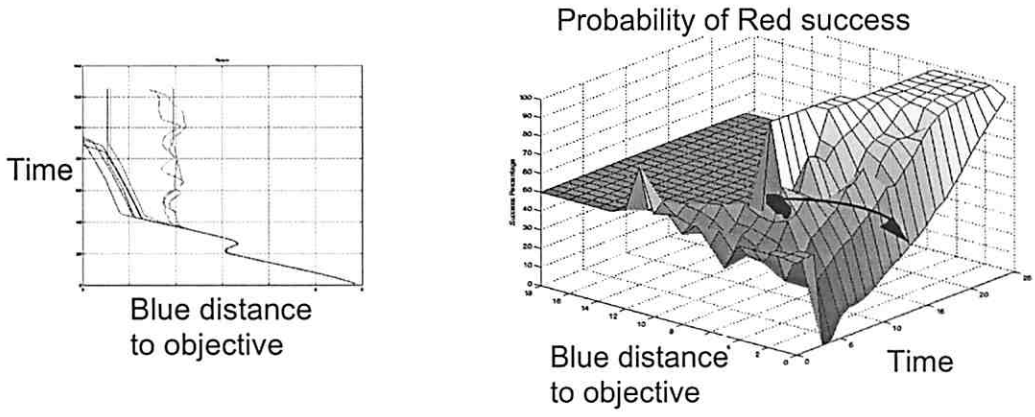

Figure 2 On the left (2a) is a map of ten trajectories, time series of distance to the objective. On the right (2b) is a map of 1000 trajectories, showing the same state variable plotted against the probability of a successful outcome.

The maps in Figure 2 represent the dynamics of engagements between a single Red and a single Blue unit. Will the same approach work – i.e., produce legible, informative surfaces relating state variables to probabilities of success – for more complex scenarios, such as the one in Figure 1? We have found no analytical answer to this question. In practice, the same approach works well for some complex scenarios, as we describe later, yet fails for some simpler ones. When we changed the scenario for Figure 2 a little, to have two Blue units attack a Red flag defended by two Red units, the resulting map was unreadable. Figure 3 shows a "thresholded" version of the map: The axes represent time and the state variable (in this case the state variable is the relative strength of the two units) and the cell colors represent probabilities. Red cells represent probabilities of Red wins greater than a threshold value, and blue cells correspond to high probabilities of Blue wins. There is no obvious continuity or pattern relating these probabilities to time and the state variable, as there was in Figure 3. The reason is that the two-on-two situation can unfold in several qualitatively different ways, and when the trajectories for these different dynamics are superimposed in a single map, the result is pretty meaningless. It is as if we tried to visualize the dynamics of several different pieces of music, or normal and irregular heartbeats, or placid and volatile markets, in one map. In any of these cases, we would first group trajectories that have similar dynamics, then build a separate map for each group.

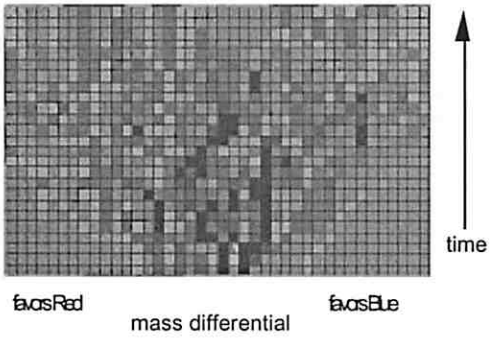

Figure 3. A probabilistic dynamic map for a two-on-two scenario in which two Blue units try to capture a Red flag defended by two Red units. The state variable (horizontal axis) is the relative strength (or mass) of the units. The vertical axis represents time. All cells with a high probability of win for Blue are colored blue, and similarly for Red. Tactically, the attacking units might concentrate on the flag, or the defending units, or split their efforts between these goals. This tactical variability is one reason that the map is not very informative.

3. BAYESIAN CLUSTERING BY DYNAMICS

We have developed a method for unsupervised clustering of univariate, categorical time series called Bayesian Clustering by Dynamics (BCD)[9,10]. (See [6,7,8] for methods for clustering continuous and multivariate time series.) Suppose we represent a military operation at time t with a state variable that takes one of k values. Between time t and time $t+1$, the state of the operation changes or remains the same, but in either case can be represented by a *transition* from a state $s_t \in \{1 \ldots k\}$ to the next state $s_{t+1} \in \{1 \ldots k\}$. Clearly, $k \times k$ state transitions are possible, and the *frequencies* with which transitions i,j occur in a military operation of duration N time steps can be stored in a *transition table* like this one:

	a	b	c	d	e
a	17	10	0	16	1
b	0	14	0	3	2
c	0	0	3	8	1
d	0	0	5	12	0
e	6	0	0	0	23

By convention, the vertical axis represents the state of an operation at time t, the horizontal axis, the state at $t+1$. Thus, the operation exhibited 10 transitions from state **a** to state **b**, and 16 transitions from **a** to **d**. The sum of the frequencies in a transition table is N–1 for an operation of duration N.

Suppose we run a Monte Carlo analysis of 100 variations on a course of action. We will have 100 tables, like the one above, each representing a single "playing out" of the plan in Capture the Flag. To cluster these variations, to find the ten or twenty qualitatively different ways the plan plays out, we need a way to judge the similarity of transition tables. The Kulback-Leibler (KL) distance is such a method. We assess the KL distance for every pair of plan instances and cluster together those that are similar. Clustering is agglomerative, which means we begin with each plan instance in its own cluster and repeatedly merge the most similar clusters; or failing that the second most similar, and so on; stopping only when no further merging would improve a quality metric. The metric in this case is the *marginal likelihood* of the data given the clustering so far. The BCD algorithm essentially does hill-climbing in a space of possible clusterings of military operations, where the height of the hill associated with any given clustering is the marginal likelihood.

BCD has been tested with data from Capture the Flag and with time series of robot data, financial data, and Bach fugues. In particular, we used BCD to cluster data from 81 trials of the two-on-two scenario that produced the unsatisfactory map in Figure 3. BCD produced eight clusters. One of them contained 94% of the plan instances in which Blue won. Three contained plan instances in which Red always won, but interestingly, the *manner* of Red's victory was very different in the three clusters; for example, in one cluster, Red won by attrition, in another it made a mad dash to grab the flag. Four clusters contained cases in which Red won (in different ways) or time expired.

Once the Monte Carlo trials are clustered, one simply builds a map, or shows the trajectories, for each cluster. Figure 4 shows the trajectories that constitute each of two clusters, one in which Blue always won, one in which both sides won some engagements. The dynamics of the clusters are quite different; one can see why superimposing trajectories from these (and the other six clusters) produces an illegible map (Fig. 3).

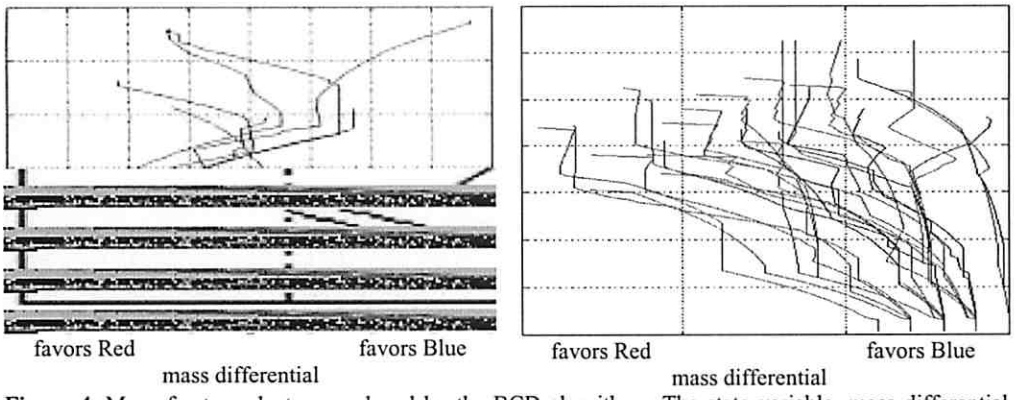

Figure 4. Maps for two clusters produced by the BCD algorithm. The state variable, mass differential (horizontal axis), behaves differently over time (vertical axis) in the clusters. On the left, the trial begins with Blue losing mass through attrition relative to Red but in most cases these losses "peak" early and Blue begins to regain ground relative to Red. In three cases, the differential later swings back toward Red, but Blue wins the games anyway. On the right, in nearly all cases, Blue loses mass monotonically throughout the game, and in most cases loses the game.

5. CONCLUSION AND FUTURE WORK

In this work, we have presented an application of visualization applied to simulations of battles. We described an abstract simulator and a domain for that simulator that allowed for data collection, a Bayesian clustering algorithm used for preprocessing data, and a technique for constructing two and three-dimensional plots of this data. In addition, sample output and analysis for a set of cases was presented.

Some human art goes into probabilistic dynamic maps, but we are working to automate the process. One trick is to find an informative state variable. By informative we mean the state variable over time is a good predictor of outcomes. Finding informative state variables is a statistical model selection problem. One could easily use regression or Bayes' networks, or related techniques, to select predictive state variables were it not for the requirement that the state variable *over time* be predictive. What we need, then, is a characterization of the *dynamics* of a state variable which can be used as a predictor of outcomes. One way to characterize the dynamics of a variable is to cluster the trajectories of the variable by their dynamics, as described previously, then use the cluster membership of a trajectory as a predictor of the outcome of the trajectory. In a recent experiment, this approach selected "distance from the flag" and "relative mass of Red and Blue units" as the state variables whose dynamics best predict the outcome of the two-on-two scenario described earlier. These results are very preliminary but suggest that one aspect of building probabilistic maps might be automated.

Another kind of automation is provided by the BCD algorithm. Maps can be difficult to interpret when several qualitatively different dynamics are superimposed (Fig. 3), but finding qualitatively different trajectories by hand is tedious, error-prone work. The BCD algorithm saves us the trouble. It clusters trajectories with similar dynamics, so when we plot maps for each cluster, the dynamics are much clearer (Fig. 4).

This work is the cornerstone of an ongoing research program in visualizations of time series data. We hope to develop algorithms to discover relevant state variables and to present candidate regions of interest that may be of use to military commanders and other consumers of time series data, such as financial analysts and roboticists. Ultimately, we hope that probabilistic maps (e.g., Fig. 2b) will inform planners, as they identify regions of opportunity and danger. Planners would evaluate their locations in the visualization and determine the best course of action move to a more favorable region.

ACKNOWLEDGEMENTS

This research is supported by DARPA and Air Force Research Laboratory under contract numbers F30602-97-1-0289 and F30602-99-C-0061. The U.S. Government is authorized to reproduce and distribute reprints for governmental purposes notwithstanding any copyright notation hereon. The views and conclusions contained herein are those of the authors and

should not be interpreted as necessarily representing the official policies or endorsements either expressed or implied, of DARPA, AFRL or the U.S. Government.

REFERENCES

1. Atkin, M., D. L. Westbrook and P. R. Cohen. 1999. Capture the Flag: Military Simulation Meets Computer Games. Presented at the *AAAI Spring Symposium on AI and Computer Games.*

2. Atkin, Marc and Paul R. Cohen. 1998. Physical Planning and Dynamics. In *Working Notes of the AAAI Fall Symposium on Distributed Continual Planning. Pp. 4-9.*

3. Atkin, Marc, David L. Westbrook, Paul R. Cohen and Gregory D. Jorstad. 1998. AFS and HAC: Domain-General Agent Simulation and Control. In *Workshop on Software Tools for Developing Agents, AAAI-98.* Pp. 89-95.

4. von Clausewitz, Karl. On War. Viking. 1983.

5. Cohen, Paul. Dynamic maps as representations of verbs. *Proceedings of the Thirteenth European Conference on Artificial Intelligence.*

6. Oates, Tim. 1999. Identifying Distinctive Subsequences in Multivariate Time Series by Clustering. *Proceedings of KDD-99, International Conference on Knowledge Discovery and Data Mining.*

7. Oates, T., Schmill, M. and Cohen, P. 1999. Identifying qualitatively different experiences: Experiments with a mobile robot. *The Sixteenth International Joint Conference on Artificial Intelligence,* IJCAI-99.

8. Oates, Tim, Laura Firoiu and Paul R. Cohen. 1999. Clustering Time Series with Hidden Markov Models and Dynamic Time Warping. To be presented at *IJCAI-99 Workshop on Sequence Learning.*

9. Sebastiani, P., M. Ramoni and P. R. Cohen. 1999. Classification of Sensory Inputs. Presented at the *IJCAI-99 Workshop on Sequence Learning.*

10. Sebastiani, P., M. Ramoni, P. R. Cohen, J. Warwick and J. Davis. 1999. Discovering Dynamics using Bayesian Clustering. In *Proceedings of The Third Symposium on Intelligent Data Analysiss*