

COMPARING AGENT TRAJECTORIES

H.V. PARUNAK,* S. BROPHY, S. BRUECKNER
NewVectors division of TTGSI

ABSTRACT

Sometimes it is desirable to measure the difference between the spatial trajectories of two or more agents. The naïve measure (the sum of Euclidean distances between locations at successive timesteps) increases with the lengths of the trajectories, which is not suitable for some applications. This paper explains the problem that motivates such a comparison, describes the design of the comparison that we are using, and gives an example of its application.

Keywords: trajectories, prediction, comparison

INTRODUCTION

It is often useful to invoke spatial metaphors, such as “location,” “move,” and “trajectory,” in describing agent behaviors.

Like any system, a software agent has a state, the vector of all variables that describe its condition. By analogy with the $\langle x, y, z \rangle$ vector of physical location, we call the set of all states that the agent can assume its “state space,” and its current state is its “location” in that space (which may be continuous or discrete, and may or may not have a proper metric). For some agents (e.g., robots or routing agents), an important component of their state is their physical location, but it is also useful to think of an agent searching for information as having a location in “semantic space,” or of a planning agent as occupying a location in “task space.”

When agents make decisions, they often change their state, and we say that they “move” in their state space. Similarly, successive decisions constitute a “trajectory.” Again, these terms are understood literally for physically situated agents, but are applicable metaphorically to any agent.

For some applications, an agent’s trajectory is more important than its individual movements, and the set of trajectories of several agents is more important than their individual trajectories. To analyze such systems, we need to compare trajectories and characterize them collectively. This paper offers some tools for this purpose

Section 2 motivates the comparison of agent trajectories in the context of a specific modeling construct, the polyagent. Section 3 describes several measures that can be used to compare trajectories. Section 4 gives an example of using the measure.

* Corresponding author address: H. Van Dyke Parunak, NewVectors, 3520 Green Court, Suite 250, Ann Arbor, MI 48105; e-mail: van.parunak@newvectors.net.

MOTIVATION FOR A MEASURE

Our polyagent technology for predicting the future (Parunak and Brueckner 2006) represents each domain entity by multiple ghost agents, each exploring a different alternative future for the entity. For clarity, we assume that the future under consideration is a possible path through two-dimensional space, though paths through more complex structures (such as semantic networks or hierarchical task networks) can also be explored.

We wish to interpret the set of trajectories discovered by the ghosts. In particular, we are interested in characterizing their divergence over time. The probabilistic behavioral models of the ghosts emulate interactions of their entities with one another and with the environment. Since these interactions are highly nonlinear in most domains, they tend to result in phenomena such as divergence and bifurcation, and can also characterize the ghosts' environment.

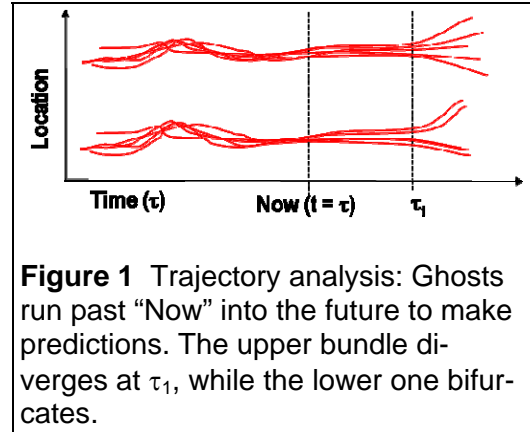


Figure 1 Trajectory analysis: Ghosts run past “Now” into the future to make predictions. The upper bundle diverges at τ_1 , while the lower one bifurcates.

Figure 1 illustrates divergence and bifurcation. Ghost time is indexed by τ and real-world time by t . Ghost simulation begins in an environment whose state corresponds to a point in the past relative to t . When $\tau = t$, we compare ghosts with the real entities that they represent, and allow the fittest ones to run into the future to form predictions. The upper bundle diverges beyond the “prediction horizon” (Parunak, Belding et al. 2007). Detecting this divergence would enable the system to avoid wasting resources on exploring further. The lower bundle bifurcates. In this case there is still predictive value in running the ghosts ahead, but detecting the branch point is crucial for understanding the system.

The degree of divergence depends heavily on the environment. For example, if ghosts are exploring possible paths for a pedestrian in the middle of an open field, they will diverge more than ghosts exploring paths for the same pedestrian at the bottom of a long, narrow valley. Distinguishing these cases can enable us to make more efficient use of the population of ghosts, and can also provide a useful characterization of the environment in its own right.

For our purposes, a measure of trajectory similarity should meet three requirements.

1. It should be independent of trajectory length, so that we can apply it across trajectory bundles of different lengths, and use it to monitor similarity as a trajectory evolves.
2. It should be tolerant of both temporal and spatial offset. Two trajectories that follow the same path but at slightly different times, or that run parallel to one another but not in exactly the same location, should be considered similar to one another, with the degree of similarity decreasing smoothly as the differences increase.
3. It should be efficient to compute. This requirement is motivated by our desire to use the measure in a real-time feedback loop to modulate the generation of polyagent ghosts.

The gold standard for measuring things is a metric, which is a function d from a Cartesian power of a set X to the reals that exhibits non-negativity, identity of non-discernibles, symmetry, and subadditivity. In general, our functions do not satisfy all of these conditions, so we call them “measures.”

DEVELOPING A MEASURE

Our approach to comparing trajectories has three components: measuring the difference of a pair of paths, extending this measure to a bundle of trajectories, and converting the unbounded measure of difference to a bounded measure of similarity.

Pairwise Comparisons

We will compute our metrics on some experimental paths. Figure 2 shows the first test set: eight trajectories in two bundles, moving from left to right. Half of the trajectories in each bundle zigzag to simulate stochastic variation around the main course of the bundle. We want our distance measure to show that these two bundles separate, then converge.

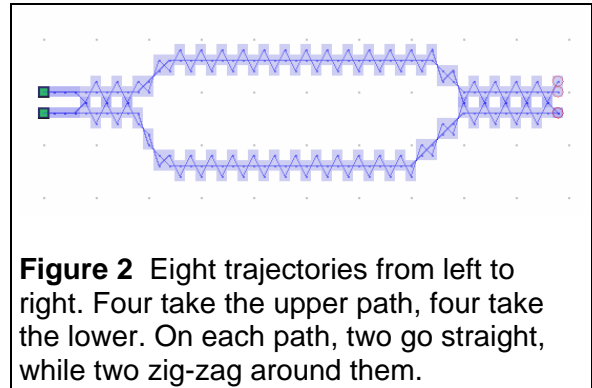


Figure 2 Eight trajectories from left to right. Four take the upper path, four take the lower. On each path, two go straight, while two zig-zag around them.

The naïve starting point for comparing trajectories is the sum of the Euclidean distances between corresponding points in the trajectories. If d_i is the Euclidean distance between the i th pair of points in a trajectory of length N , the distance is $\sum d_i$. When we have more than two trajectories, we take the mean of the pairwise distances. This approach is reasonable when

- All trajectories have the same number of steps
- All trajectory steps are of the same time duration
- All trajectories start at the same location

The sum of pointwise Euclidean distances is monotone nondecreasing as the length of the trajectory increases, since each additional step may add more difference. Thus pairs of long trajectories show a larger difference from each other than pairs of short ones, simply because they include more points, violating our first requirement. Some form of normalization is needed.

The obvious normalization is by the length of the trajectory, giving the average separation per step,

$$\frac{1}{N} \sum_{i=1}^N d_i$$

If we apply this measure in real-time, the number of items in the sum and thus the normalizing constant increase throughout the run, with undesirable consequences. Figure 3 shows the point-by-point Euclidean distances (the upper zigzag line), and the running average separation (the lower line). In the lower line, while the divergence of the trajectories is clearly marked,

their subsequent convergence is much less clear, because the change is diluted by the many differences already included in the average.

For monitoring trajectory proximity during execution, a running average of point-wise trajectory separations over a **Scoring Window** is more effective than an overall average. In our tests, a scoring window of 4 is long enough to smooth the scores. The wider the scoring window, the longer it takes for the score to reflect a change in path similarity patterns. We use the scoring window to normalize scores for the various refinements discussed below. Figure 4 shows the behavior of a scoring window of width 4 on the trajectories of Figure 2. It smooths out the zigzags and gives a distance profile that corresponds to our intuition about the overall behavior of the bundles, but it lags the actual movement of the trajectories by 2 time steps (half the width of the window).

The Euclidean measure does not recognize path pairs that follow identical routes with a small time lag as being similar, and thus does not satisfy our second requirement. Two alternative mechanisms can accommodate time lapses, step windows and the Laurinen algorithm.

The step window method uses two parameters, the **Past Step Limit** and the **Future Step Limit**, to define a window of comparison around the matching point on the paired path. For each point on one path, the distance is computed to every point on the other path that falls within this window. The shortest such distance is that point's distance from the other trajectory. Then these distances are averaged over the trajectory.

This approach captures the similarity between some lagging paths, but shows discontinuities as paths move within the window, and cannot discriminate between paths that lag at different distances if they all fall within the window. These problems result from the abrupt boundaries and arbitrary length of the window. In addition, of the four conditions for a formal metric, the step window method violates all except nonnegativity. The main culprit is asymmetry: the sum of distances of points in trajectory A to the closest points in trajectory B is not necessarily the same as the sum of distances of points in trajectory B to the closest points in trajectory A.

A more general method for aligning paths that are not exactly aligned temporally is Laurinen's algorithm (Laurinen, Siirtola et al. 2006), which explicitly includes temporal distance when measuring

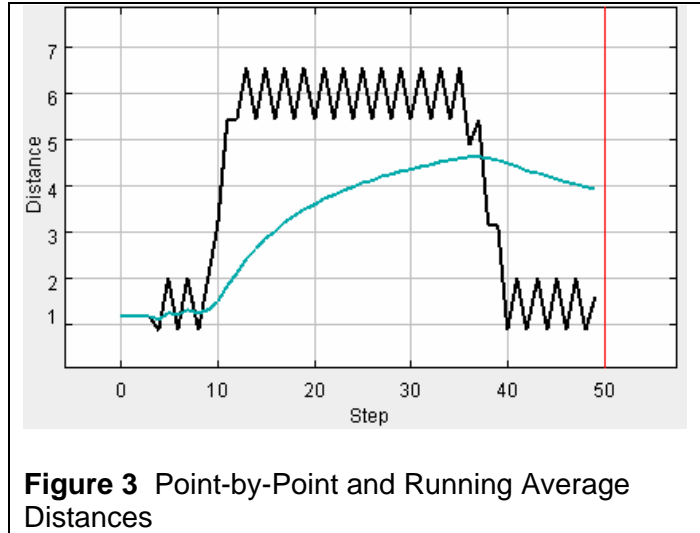


Figure 3 Point-by-Point and Running Average Distances

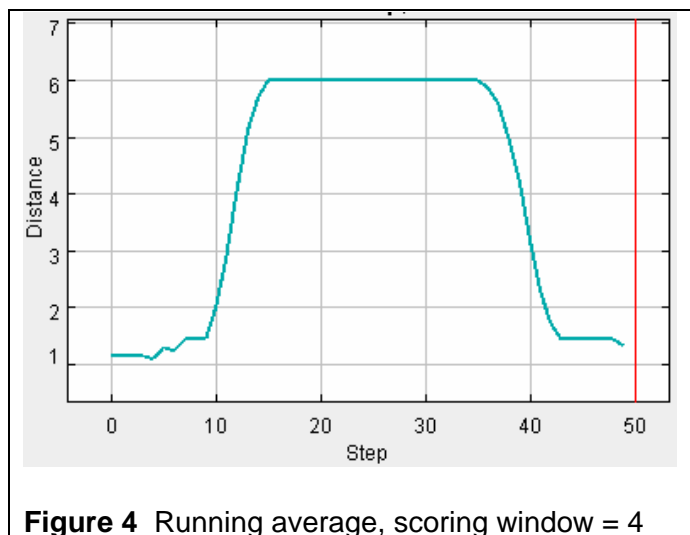


Figure 4 Running average, scoring window = 4

the separation between points on two trajectories. Applying this algorithm requires defining a mapping from time to space. We multiply the time-distance between the comparison steps by a **Step Weight Factor** and use the result as a third component in the Euclidean distance computation (along with the x-distance and y-distance components) in selecting the closest matching points between two trajectories. (The Step Weight Factor is analogous to the speed of light in special relativity, in its role of rendering space and time commensurate.) In our polyagent application, agents can move a maximum of five cells at each time step, so we set the step weight to $1/5 = 0.2$. This approach allows lagging paths to score as similar, and provides a smoother function than does the step window approach.

By itself, this computation is asymmetrical, and violates the same three metric conditions as the step window method. To ameliorate the problem, Laurinen computes the distance in both directions and chooses the maximum of the two. This approach violates only the triangle inequality. In practice, in spite of this shortcoming, it is serviceable as a well-defined measure of trajectory similarity.

Figure 5 shows the effect of these two adjustments on time-lagged paths. Four trajectories (a straight one and a zigzag one for each of the upper and lower branches) are synchronized with each other. One straight trajectory for each branch is delayed by three time steps, and one zigzag trajectory for each branch is delayed by four time steps. The upper curve uses a scoring window of 4, but makes no correction for lagging, and as a result gives a higher distance (about 7) than the same measure applied to time-synchronized trajectories in Figure 4 (about 6). The lower two curves, nearly superimposed, show the Laurinen measure with step weight 0.2 (slightly higher) and past step limit = future step limit = 5. Both cases greatly reduce the penalty imposed by the time lag.

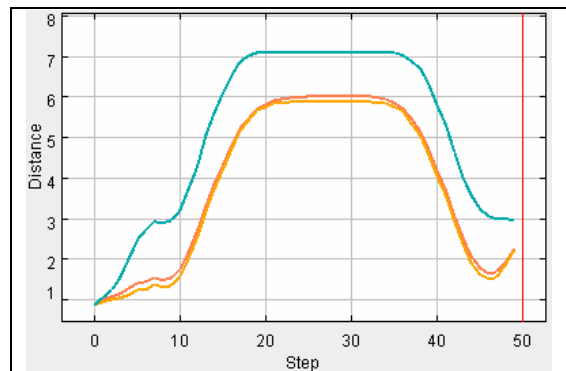


Figure 5 Distances of time-lagged paths. Top: scoring window 4. Middle: Laurinen with step weight 0.2. Bottom: past limit = future limit = 5.

Dealing with Bundles

The methods discussed so far define a similarity between two paths. In some applications, we want to characterize the tightness or looseness of a bundle of trajectories.

The naïve approach (used in the plots so far) is to average the similarity scores of all possible pairs in the bundle, requiring $O(N^2)$ operations. In keeping with our third requirement, we prefer a linear time algorithm to enable the similarity score to be used as a live feedback control. Various **Pairing Strategies** can reduce the computation while maintaining the same scoring pattern. We explored four strategies:

1. PATH_PAIRS computes all path-pair combinations ($2N(N-1)$ operations).
2. MEAN_PAIRS compares all paths against the bundle mean location ($2N$ operations).

3. INTO_MEAN measures the distance from individual paths to the mean (N operations).
4. FROM_MEAN measures the distance from the mean to individual paths (N operations).

The MEAN_PAIRS approach results in a score that follows the same trend, but is generally lower than the full PATH_PAIRS score, because the bundle mean is usually closer to a path than the score that path would get when compared to all of the paths individually.

Laurinen measures the difference between two paths in both directions and takes the maximum. PATH_PAIRS and MEAN_PAIRS follow this convention (thus the factor of two in the number of operations). Notice the impact of this convention when reasoning with mean paths. INTO_MEAN uses only the components of the score from each individual path to the mean, while FROM_MEAN uses only the components from the mean to the individual paths. The mean path naturally tends to be straighter than the individual paths, resulting in a systematic difference between INTO_MEAN and FROM_MEAN.

Consider comparing the mean path with a path that mostly follows the group, but loops out and then back into the bunch (Figure 6). First, consider the INTO_MEAN score from point e on an individual path to the mean path. All of the nearest points (a , b , or c) on the mean path are far away. But in computing the FROM_MEAN score, points a , b , and c will find close points on the individual trajectory (d , d , and f , respectively), and their relatively large distance to point e will never enter the computation.

Figure 7 shows all four scores for the trajectories of Figure 2.

This observation enables a further efficiency. Since MEAN_PAIRS uses the larger of the INTO_MEAN and FROM_MEAN scores, and since INTO_MEAN is usually larger than FROM_MEAN, INTO_MEAN is an efficient surrogate for MEAN_PAIRS. However, one may prefer to use FROM_MEAN instead, for the following reason. The mean over a set of trajectories tends to smooth out their individual variations, and so FROM_MEAN automatically smooths without the time lag imposed by a scoring window. Figure 9 compares the FROM_MEAN scores with windows of 1 and 4. As the distance increases relative to the variance, the measure with scoring window of 1 (the left-most curve) becomes almost as smooth as that with a window of 4 (to the right), and without the lag.

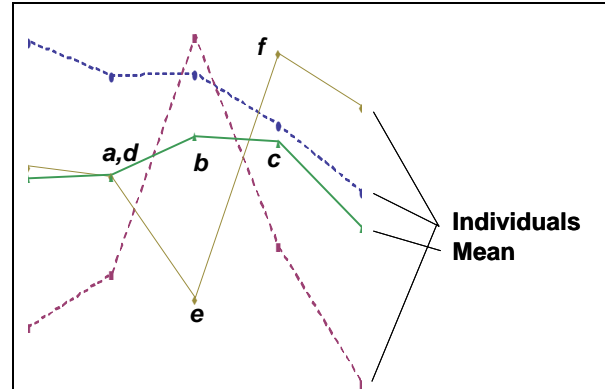


Figure 6 INTO_MEAN vs. FROM_MEAN. Points a , b , and c are on the mean path; points d , e , and f are on one of the individual paths.

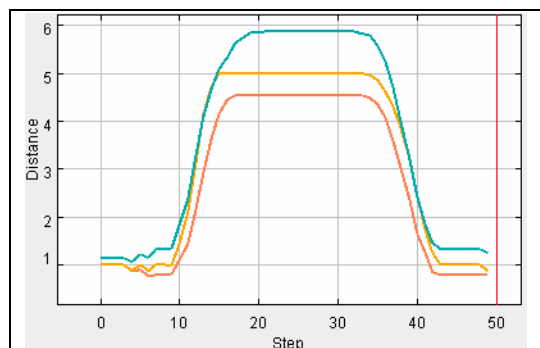


Figure 7 Bundle scores. Top: PATH_PAIRS. Middle: INTO_MEAN = MEAN_PAIRS. Bottom: FROM_MEAN.

In applications, the use of the mean both to compute the baseline trajectory and to combine the differences of individual trajectories from the baseline is sensitive to outliers, and in practice we prefer to use medians for both of these computations.

Similarity Calculation

Our measures so far are unbounded upward. It is often more convenient to have a measure that is bounded (say, in $[0,1]$). The naïve transform, to compute the similarity as the inverse of the distance, $1/d$, would work if our separations were always > 1 . When computing the distance for a bundle, rather than just a pair of paths, or when using a scoring window, the distance can be < 1 , resulting in similarity scores > 1 . Several approaches are possible.

We could define the similarity to be 1 for any distance < 1 . The step function generated by this approach loses information as to whether the computed bundle distance is increasing or decreasing for small separations.

We could scale the similarity as $N/(N+d)$. This transform avoids the step function, and raises the values to use more of the 0 to 1 range. But the shape of the curve still drops off too quickly for distances that should all be close to similar.

The transformation we have found most satisfactory is a sigmoid (Figure 8),

$$similarity = \frac{1}{1 + e^{-steep(offset-d)}}$$

Offset determines the distance that is mapped to a similarity of 0.5, and *steep* determines the steepness of the transform at that point. For our test cases, *offset* = 2 and *steep* = 2.5 closely follow the naïve $1/d$ transformation. Figure 10 shows the similarity obtained by this transformation from the INTO_MEAN measure with scoring window of 4.

USING THE MEASURE

This section analyzes some actual ghost trajectories from a military scenario that shows

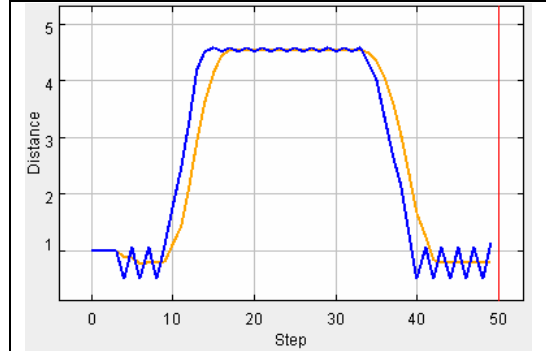


Figure 9 Smoothing effect of FROM_MEAN

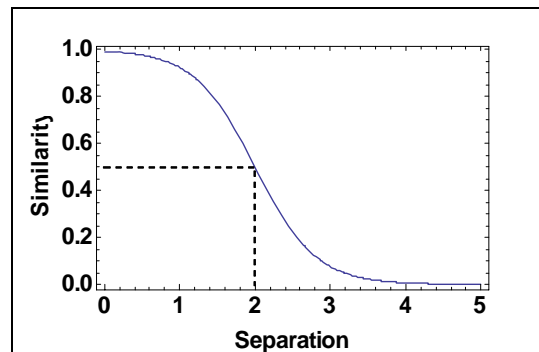


Figure 8 Sigmoid Transformation, for *steep* = 2.5 and *offset* = 2

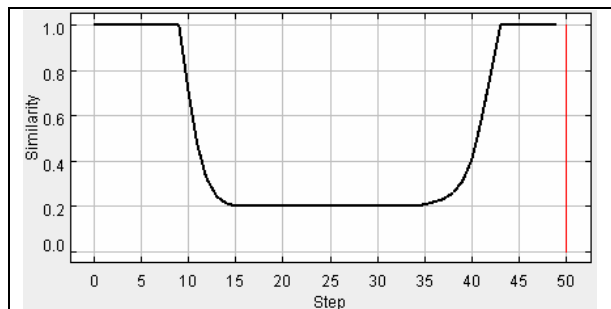


Figure 10 INTO_MEAN distance for Figure 2 transformed to similarity

the effect of the environment on their movement. The terrain includes both open areas and roads. When ghosts are on a road, they prefer to follow it, but in open terrain they move more freely. Our plots do not show terrain features explicitly, but we will describe them for the examples we discuss.

In addition to plotting the INTO_MEAN similarity score, we also plot the option set entropy (OSE). Our similarity scores are global measures, appropriate for centralized use in managing a polyagent system, but not accessible to individual agents. An agent can monitor its option set entropy locally. So the relation between these two characteristics is of great interest.

In this application ghosts live on a square lattice, and make their choices stochastically, spinning a roulette wheel with as many segments as they have next possible steps (the “option set”) whose segments are weighted in the following fashion. First, the ghost combines a number of environmental signals (“digital pheromones”) from each option that it may choose into a single attractiveness score for that option. In our application, the options are the cells to which the ghost may move in the next step. Then, to adjust the degree of determinism in the system, we map the attractiveness to a probability using the Boltzmann distribution,

$$p_i = \frac{e^{w_i/t}}{\sum_i e^{w_i/t}}$$

where w_i is the attractiveness of the i th option, p_i is the probability of moving to that option, and t is the Boltzmann temperature. When t is large compared with w_i , each option has an equal chance of being selected. When t is small, the choice becomes more deterministic in favor of the most attractive option.

The entropy over the option set probabilities, normalized by the log of the number of possible steps, reflects how much guidance the ghost has at that step. This option set entropy (OSE) varies from 0 when the ghost is moving deterministically to 1 when it is executing a random walk. OSE is a good summary of how converged an agent system is (Brueckner and Parunak 2005). Might it serve as a local indicator of the convergence of an agent bundle?

Figure 11 shows 19 trajectories that remain on a road system. The trajectories all begin at the dark area toward the lower-right of the figure. Figure 12 shows the similarity¹ and average OSE across all agents for this system.

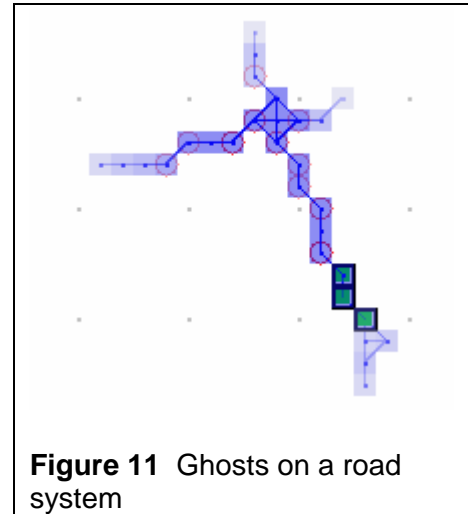


Figure 11 Ghosts on a road system

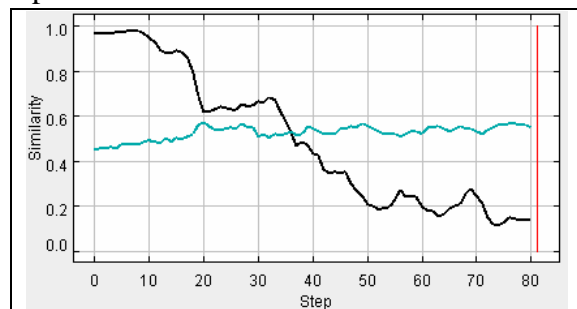


Figure 12 Similarity (descending curve) and OSE (gently rising curve) for Figure 11

¹ INTO_MEAN, Laurinen step weight 0.2, scoring window 4, transformed through sigmoid with offset = 2 and steep = 2.5.

Consider first the similarity. The trajectories diverge initially as the ghosts spread out. Then, between time 20 and 30, they come closer together, before continuing to diverge beyond time 33. At time 20 the ghosts reach the crossroads. Because they have several options available (note the small peak in OSE at time 20), they tend to loiter in the area for a few moments, and their local trajectories converge. Once each ghost converges on a road to follow out of the crossroad, the similarity again falls.

The OSE rises gently until the ghosts reach the crossroad, where it reaches a local maximum, then levels off for the rest of the run. The initial increase reflects the ghosts' initial exploration. The peak reflects their contemplation of the crossroad, and the final level portion corresponds to their constrained exploration of the various roads.

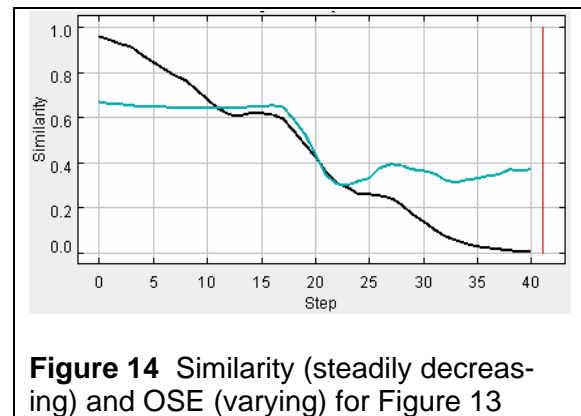
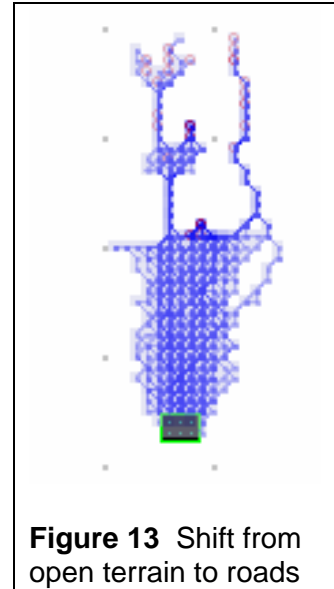
Now consider the 59 trajectories in Figure 13. Figure 14 shows

similarity and OSE. The ghosts, again moving from south to north, begin in open terrain where they spread out, reflected in decreasing similarity. The OSE is constant during this time: the environment does not constrain the ghosts, other than a general attraction toward the roads at the north. At time 12, some trajectories discover the road emerging from the right-hand side of the main cluster, and this constraint causes similarity to level off. At time 17, further roads branch out. Because the ghosts have multiple roads from which to choose, similarity begins to drop again, while the additional movement constraints from the roads cause the OSE to drop. The rise in OSE from time 22 to 27 corresponds to the first wide spot on the left-hand road, offering ghosts more options. Because they loiter in this region, the decline of similarity is less pronounced. OSE again decreases as the ghosts follow the roads leading from this wide spot, then increases gently again after time 33, as they discover the wider set of options at the end of the left-hand road.

These examples show that while OSE and similarity are sometimes correlated, they measure different things. OSE reflects how constrained individual ghosts are, while similarity reflects how close they are to one another. All four combinations can occur. Highly constrained ghosts can be close to or far from one another, as can ghosts that experience little constraint. Correlations emerge when ghosts that are generally traveling in the same direction reach a decision point, which increases their OSE and at the same time allows them to catch up with one another, increasing their similarity.

CONCLUSION

It is often desirable to characterize the trajectories exhibited by a set of agents. In our work, these trajectories represent alternative possible futures being generated by a polyagent, and



the degree to which they converge is an important index of the quality of the predictions. Such measures may be useful in other applications as well (for example, clustering targets into groups within which the behavior is similar). We seek measures that are independent of the trajectory length (so they can be used for real-time control of the agents), tolerant of both temporal and spatial offset, and efficient to compute. Naïve measures do not satisfy these requirements, but the transforms presented in this paper provide a rich toolbox that we are using in analyzing predictive trajectories.

ACKNOWLEDGMENT

This research is funded by the National Geospatial-intelligence Agency (NGA) under contract GS-35F-4912H. The views expressed in this paper are solely those of the authors, and are not endorsed by the NGA or the US Government.

REFERENCES

- Brueckner, S. and H. V. D. Parunak, 2005, *Information-Driven Phase Changes in Multi-Agent Coordination*. Workshop on Engineering Self-Organizing Systems (ESOA, at AAMAS 2005), Utrecht, Netherlands, Springer. 104-119.
<http://www.newvectors.net/staff/parunakv/AAMAS03InfoPhaseChange.pdf>.
- Laurinen, P., P. Siirtola, et al., 2006, *Efficient Algorithm for Calculating Similarity between Trajectories Containing an Increasing Dimension*. 24th International IASTED Multi-Conference on Artificial Intelligence and Applications (AIA2006), Innsbruck, Austria. 392-399.
<http://delivery.acm.org/10.1145/1170000/1166957/p392-laurinen.pdf?key1=1166957&key2=6755105711&coll=&dl=ACM&CFID=15151515&CFTOKEN=6184618>.
- Parunak, H. V. D., T. C. Belding, et al., 2007, *Prediction Horizons in Polyagent Models*. Sixth International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS07), Honolulu, HI. 930-932.
www.newvectors.net/staff/parunakv/AAMAS07PH.pdf.
- Parunak, H. V. D. and S. Brueckner, 2006, *Polyagents Model Multiple Futures Concurrently*. Social Agents: Results and Prospects (Agent 2006), Chicago, IL, Argonne National Laboratory.