A Glider for Every Graph: Exploring the Algorithmic Requirements for Rotationally Invariant, Straight-Line Motion

Alexander G. D. Lamb

1545 Scenic Avenue
Berkeley, CA 94708, USA
alex.lamb@gmail.com

The primary goal of digital physics research is to provide a description of the physical universe in terms of simple programs. One approach to attaining this goal is creating a toolbox of algorithms that reproduce the behavior of basic quantum phenomena. As a step in this direction, a simple pseudo-particle algorithm has been developed that exhibits rotationally invariant, glider-like motion across graphs in two or more dimensions. This algorithm is applied to a range of lattice and irregular graphs from the sparse to the densely connected, and it is shown that rotationally invariant motion can be easily obtained from irregular graphs that are sufficiently densely connected. Such graphs are also shown to be potentially compatible with spatial curvature and relativistic invariance. This work points the way toward a class of algorithms that can be used to tightly approximate the basic phenomena encountered in particle physics, while maintaining the desired properties of discreteness, determinism, and algorithmic simplicity.

1. Introduction

The study of particle physics is driven by a desire to seek out the hidden symmetries of nature, many of which appear to have smoothly varying values. However, theories based on continuous mathematics have so far not provided a fully coherent picture of the natural world, leading some theorists working in particle physics to wonder if discrete models might one day yield a clearer, more logical vision [1-3]. This view echoes one that has been under consideration in computer science for some time [4-6].

One approach to constructing such a model is through the use of simple algorithms. This technique has the advantages of unparalleled computational elegance and rich descriptive power, but has had limited success in producing realistic analogs of natural phenomena. Attempts to approximate physical systems using simple algorithms have so far broadly concentrated on two lines of research: cellular automata (CAs) [5, 7] and systems based on networks [3, 8].

https://doi.org/10.25088/ComplexSystems.18.4.439

CAs exhibit a broad spectrum of complex behaviors that have been well researched. This versatility has been employed to produce reversible patterns that show some of the properties of physical particles [3]. However, CAs produce anisotropic behavior except at very large scales and suffer from not being able to easily model spatial curvature or spatial expansion.

Network-based methods use graphs to provide an analog of physical space that is capable of modeling both spatial curvature and expansion with ease. However, in such systems, the kinds of predictable behavior witnessed in CAs have been harder to find. One such behavior is straight-line motion, as witnessed in CA “gliders”. So far, the only known examples of straight-line pattern propagation are confined to regular, planar backgrounds and lack sufficient flexibility to model natural systems [3]. Furthermore, no mechanism has been discovered that will permit a propagating pattern to follow a geodesic across a graph that has been constructed to mimic a curved surface [3]. Gliders that do not reliably follow geodesics cannot take advantage of the modeling flexibility that graphs provide.

In this paper, we describe a preliminary investigation into a new method that utilizes densely connected graphs and algorithms operating over sets of nodes. To demonstrate the potential of this scheme, we outline an algorithm called “Jellyfish” that produces a well-behaved “pseudo-particle”. This pseudo-particle is capable of approximating rotationally invariant straight-line motion across irregular graphs.

We describe experiments undertaken to test the limits of our algorithm’s performance on graphs designed to emulate two- and three-dimensional spaces at large scales. We also show the difference in its behavior between regular and irregular environments. Evidence is provided that this system is rich enough to describe both geodesic motion across a curved space and relativistic motion conforming to the Lorentz metric.

In Section 2, we describe the Jellyfish algorithm, the formula used to generate test graphs, and a metric for straightness. Section 3 outlines the details of our simulations and their findings. Section 4 covers two extra qualitative investigations conducted to test the further potential of the approach on orbital and relativistic motion. Section 5 contains a discussion of the implications of this research, and Section 6 summarizes our conclusions.

2. Algorithms

2.1 The Sample Graph Formula

This research seeks to aid in the development of background-independent models of nature. In such models, notions of distance, orientation, and motion are intended to arise naturally out of more abstract structures. However, testing our algorithm requires that we quantify straightness of motion with respect to some external measure. We therefore assess our pseudo-particle's performance using basic geometry.

For this purpose, we construct graphs designed to approximate smooth, flat manifolds at large scales. This is carried out by placing nodes at random on a manifold so that regions of equal area are equally likely to contain a node. This ensures that nodes are distributed with approximately constant density. Then, nodes within a *linking radius* r of each other (where r is a constant) are connected. In all cases described in this paper, unless otherwise noted, we use a flat two-dimensional Euclidean surface that is glued at the edges to form a torus. The following recipe outlines the process employed in this case.

- Generate a large number of nodes with randomly chosen x and y coordinates between the limit values of zero and one.
- For each pair of nodes, p and q, compute their Euclidean distance d for a torus, as given by:

  $$d = \sqrt{\left((\min(\Delta X_{p\,q}, 1 - \Delta X_{p\,q}))^2 + (\min(\Delta Y_{p\,q}, 1 - (\Delta Y_{p\,q})))^2\right)} \tag{1}$$

  where $\Delta X_{p\,q}$ denotes the x-axis distance between points p and q, $(\Delta X_{p\,q} \equiv X_q - X_p)$.
- Connect each node to all neighbors lying within the linking radius r.

This method produces graphs that vary with a single parameter, the linking radius r. We explore the dependence on this parameter in Section 3.1.

As these manifolds have zero curvature, the required calculations for testing the straightness of motion are straightforward. Because these graphs are closed, they permit unbroken motion in a straight line for an arbitrary number of iterations. A pseudo-particle's motion always wraps, regardless of its orientation.

2.2 The Straightness Metric

For irregular graphs of the sort described, changes in angle over short distances are unavoidable, as there is only a finite number of arcs leaving any given node. Such changes in angle are therefore a property of the medium and do not contribute meaningfully to an attempt to assess straightness of motion. Thus, the metric we use to measure our results is designed to ignore changes of angle of travel over short distances.

For a given time sequence of length t, the vector from the pseudo-particle position at time 0 to its position at time $t/2$ is measured. This vector is then compared to that for $t/2$ to t and the angle between them calculated. We call this value the *angular deviation* θ_d for that sequence. The position of the pseudo-particle is defined to be the mean coordinates of all nodes contained in the union of the two Jellyfish sets. The nature and purpose of these sets is explained in Section 2.3. We take care to determine the mean coordinates using functions compatible with measurements on a closed surface so as to avoid numerical errors caused by coordinate wrapping. Angular deviation is shown here in degrees for ease of comprehension, with zero representing straight-line motion. We make no attempt here to generalize this metric to curved manifolds as this lies outside the scope of the paper. However, the use of such manifolds is briefly discussed in Section 4.1.

2.3 The Jellyfish Algorithm

As alluded to in Section 2.2, there are no straight lines in a graph except those of length one that point in a finite set of directions from each node. It is perhaps more accurate, then, to describe the motion we wish our algorithm to achieve as *straightest possible* motion, or *forward motion*. However, graphs have no inherent notion of orientation. The arcs leaving any given node are treated identically. Consequently, any concept of forward motion is necessarily dependent on relationships between multiple nodes.

To circumvent this problem, our algorithm defines forward motion over sets of nodes rather than individual nodes. Using sets of nodes allows defining *position* as a function of many nodes, and *straightness* to be defined over sets of arcs that have more reliable properties in bulk than in isolation.

Jellyfish operates on two sets of nodes, A and B, each containing n nodes. The algorithm proceeds as follows.

- Each neighbor x to the set of nodes $A \bigcup B$ is assigned a score. The value of this score is given by:

  $$|\,\text{Neighbors}(x) \bigcap A\,| - |\,\text{Neighbors}(x) \bigcap B\,| \tag{2}$$

  where Neighbors(x) denotes the nodes reachable from node x via a single arc.
- Set B is emptied and populated with the members of set A.
- Set A is emptied and updated to contain the n highest-scoring neighbors.

This completes a single step forward. The algorithm then repeats to compute further steps.

We can represent the Jellyfish algorithm as an iterative function using the following formalism. Set S of size m has members $x_1, x_2 \ldots x_m$. The function Neighbors(S) indicates the set of all nodes that are linked to by any member of S. We also define the function Top(n, S, $f(x)$), which returns the n top scoring members of S, using the metric $f(x)$ to score each element x.

The function can then be written as:

$$\begin{aligned} A' &= \text{Top}\,(n, \text{Neighbors}(A \bigcup B), \\ &\qquad |\,\text{Neighbors}(x) \bigcap A\,| - |\,\text{Neighbors}(x) \bigcap B\,|) \\ B' &= A. \end{aligned} \tag{3}$$

This function gives us a single parameter to test against: the pseudo-particle set size n. Figure 1 shows the algorithm in use for $n = 3$.

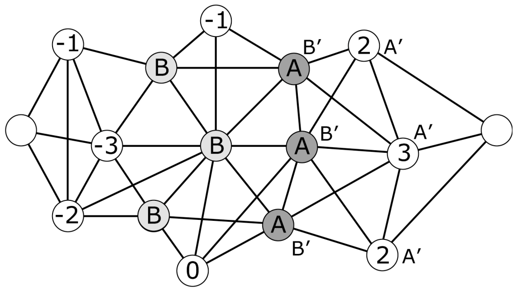

Figure 1. Motion formula for Jellyfish. Neighbors of the pseudo-particle are scored according to the number of members of set A they are adjacent to, minus the number of members of set B. The nodes marked A' and B' indicate the position of the pseudo-particle elements after a single iteration of the algorithm has been completed.

It is possible, even likely, that multiple candidate nodes examined by the Jellyfish algorithm in a given iteration will have the same score. However, the algorithm's behavior under this condition is not explicitly defined by this formula. The reason for this is outlined in Section 2.4.

2.4 Factors Governing Algorithm Choice

One reason to use Jellyfish is that the definition of forward motion it embodies is not limited to a particular dimensionality or curvature. We should expect its performance to be the same for any graph that appears locally smooth. Indeed, we briefly investigate curved spaces in section Section 4.1. Additionally, as Jellyfish utilizes node sets of arbitrary size rather than individual nodes, our anticipation prior to experimentation was that sets could be grown indefinitely to obtain an increasingly accurate conception of straightness.

Another reason we chose to use Jellyfish for motion experiments was the property of spontaneous localization that it showed during preliminary studies. Because the pseudo-particle selects the highest scoring neighbors for its new members, the slightest over-density of positively scored nodes on a graph will cause the sets to rapidly converge onto adjacent nodes. Within a few iterations the pseudo-particle fully localizes (coheres), so long as the graph conditions are sufficiently dense to permit the algorithm to operate.

It is easy to construct scenarios that one might imagine would suppress this effect. For example: a starting state for the algorithm in which every pseudo-particle element is separated from the others by a minimum path length greater than two. In this case, every neighbor examined by the algorithm has a score of either plus or minus one. However, the pseudo-particle still coheres under these conditions, even for implementations where care is taken to treat the neighbors of each set element exactly equally and to select randomly between them.

This is because the algorithm automatically introduces bias upon iteration. Even if the neighbors of every member of set *A* have identical scores, it is very unlikely that exactly one neighbor of each element will be selected in the next iteration. Thus, one iteration of the algorithm usually produces a local over-density. For all implementations examined during preliminary testing, the mechanism used to select between equally scored neighbors made no difference. (For this reason, an explicit strategy was not listed with the algorithm definition.) No formal proof that this behavior will always occur has yet been sought. However, it is anticipated that it should be relatively straightforward to show that the probability of a pseudo-particle not being formed should become vanishingly small with increased iterations.

This localization effect provides an ideal experimental starting condition for the algorithm. By randomly scattering the points of both sets *A* and *B* across a graph, the risk of a selection effect in the resultant flight-paths can be minimized. Care was taken when examining the results of the experiments to confirm that the implementation choice had not affected results. (See Section 3.3.)

3. Simulations in Euclidean Space

We used our graphs and motion algorithm to carry out four trial simulations to explore the viability of our approach, as outlined in the following sections. For those interested in an animated depiction of the motion described, video shorts generated for the 2008 Midwest NKS Conference can be found online. URLs are listed in the appendix.

3.1 Simulation One: Exploration of Parameter Space

In order to test the basic capacity of Jellyfish for straight-line motion, a range of studies were conducted for pseudo-particle set size n with $2 < n < 1024$, and for graph linking radius r for $0.012 < r < 0.036$. These were carried out on two-dimensional graphs of 10^5 nodes.

Simulations began with the elements of sets *A* and *B* randomly distributed throughout the graph as described in Section 2.4. Each run was comprised of a 20-step delay to ensure pseudo-particle cohesion, followed by a measurement sequence of $t = 20$ steps. Each angular deviation result was then averaged over a set of 20 runs.

The resulting measurements of angular deviation θ_d are shown in Figure 2. Lines indicate runs with differing values of set size n. For clarity, runs with $n \leq 32$ are shown in panel (a), while those with $n \geq 32$ are in panel (b). We also show a best-case control (labeled "ctrl") for comparison. The control particle determined its next position by examining the coordinates of each neighbor and selecting the node lying closest to the geometric ideal for straight-line motion.

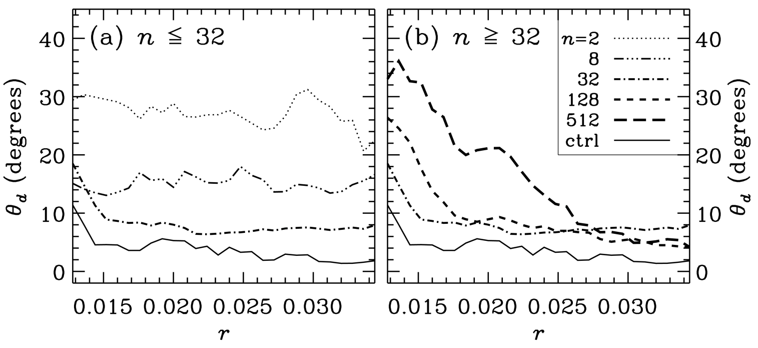

Figure 2. Change in angular deviation θ_d with linking radius r for various pseudo-particle set sizes n. For low values of n (a), the linking radius has little impact. For high n (b), performance asymptotically improves for high r, while worsening for low r. This effect magnifies with increasing n.

For $n \leq 32$ in Figure 2(a), the expected behavior was confirmed. Angular deviation decreased as n increased. As expected, larger set sizes produce straighter motion because an increased sample size is available from which to obtain a consensus notion of "forward". How ever, the algorithm showed different behavior for $n \geq 32$ in Figure 2(b). On smooth graphs (high r), increasing n resulted in straighter motion (lower θ_d) as before, while on coarse graphs (low r), performance actually degraded as n increased.

Closer observation of these cases showed that under coarse graph conditions there appears to be an insufficient number of neighbor associations to hold a large particle together. Thus, it cannot easily move in a single direction. The pseudo-particle delocalizes, spreading out on multiple paths. This gives a poorly defined average position for straightness measurement.

Figure 3 presents this data in a different orientation, showing θ_d as a function of n for a range of r. Here we can see a set of valley-shaped curves, with the minima sliding to the right with increasing r. This suggests that for any r value there is an optimum pseudo-particle size for which the straightest motion is achieved.

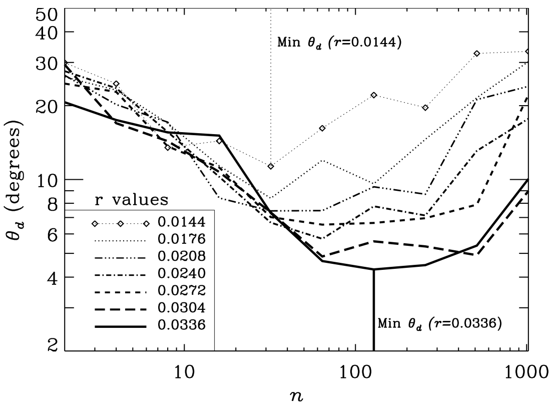

Figure 3. Change in angular deviation θ_d with pseudo-particle set size n for various linking radii r. For each r there is a valley-shaped curve with a minimum value of θ_d, with the minimum generally occurring at higher values of n for higher r. For the values of $r = 0.0144$ and $r = 0.0336$ the points corresponding to minimum θ_d values have been marked.

Pursuing this, we performed a comparative analysis of the average number of neighbors per node in each graph, and the total size of a pseudo-particle $|A \bigcup B|$. This suggested that the optimum pseudo-particle occurs when its size $2\,n$ is the same as the average number of node neighbors.

3.2 Simulation Two: Optimum Particle Behavior

Using the optimum particle size formula determined in Section 3.1, another exploratory simulation was run. This simulation was intended to determine the minimum angular deviation given increasing linking radius r, and to provide some guide as to what behavior might be expected at the continuum limit.

This experiment used the same angular deviation metric, and measured linking radii from $0.012 < r < 0.072$. Each run measured the motion of the pseudo-particle over 20 steps, after a 20-step delay to ensure cohesion. Each angular deviation result was then averaged over a set of 20 runs.

Figure 4 shows the behavior of particles with size matching the average neighbor count. This produces a well-behaved angular deviation curve that asymptotically approached zero for large r. This was expected because motion should tend toward a straight line for the most densely connected networks, given an appropriate n value.

It should also be noted, however, that even when ideally sized, the behavior of Jellyfish pseudo-particles becomes extremely poor on coarse graphs. In the limiting case of planar graphs with only three links leaving each node, the motion of the particle is effectively random.

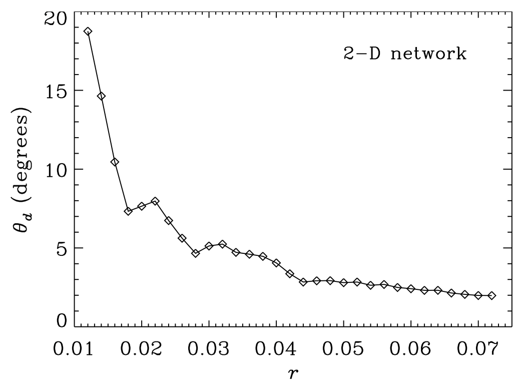

Figure 4. Change in angular deviation θ_d with linking radii r for "ideal" values of n. As r increases, the straightness of motion across two-dimensional graphs asymptotically improves.

3.3 Simulation Three: Motion on Regular Graphs

To determine whether the use of irregular graphs was a necessary ingredient for producing rotationally invariant motion with the Jellyfish algorithm, a third simulation was conducted.

In this case we conducted 500 runs of length 10 after a 20-step cohesion delay for both irregular and rectangular lattice graphs of 102 400 (320^2) nodes, with $0.02 < r < 0.038$ and $n = 50$. In both graph types, the same linking strategy was used. Only the positioning of the nodes differed—in the lattice case, nodes were arrayed on a square grid. For each run, the angle of flight with respect to the x-axis θ_f was measured. Fresh graphs were generated for each run to remove any potential bias from the results.

The resulting distributions of flight angles θ_f for $r = 0.02$ and $r = 0.038$ are shown in Figure 5. As expected, the histograms for irregular graphs showed an even distribution of angles of flight for all values of r. The pseudo-particle moved in all directions with equal probability. However, the histograms for rectangular lattice graphs always showed sharp spikes for motion in certain selected directions and often no coverage for others.

This matched our observations from preliminary analyses that in all observed cases, using lattices biased the direction of flight within a few iterations. Bias was always toward a direction defined by some minimal ratio of axis steps on the lattice. The same behavior was seen for all set-based algorithms tested, and for all kinds of regular lattice.

Interestingly, this experiment did not show the number and orientation of selected directions changing monotonically with increasing r. The relationship between n, r, and the set of resultant selected directions appears to be complex and its analysis lies outside the scope of this paper.

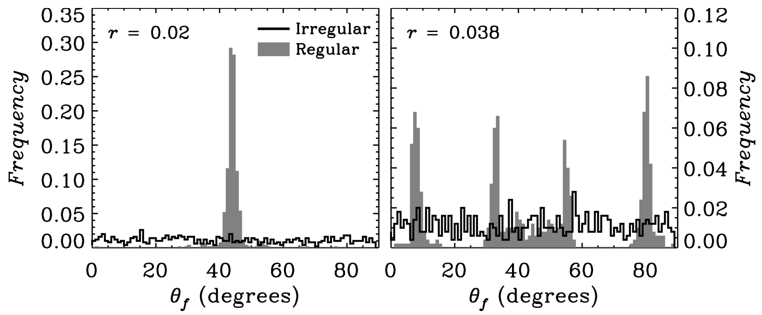

Figure 5. Histograms showing occurrences of pseudo-particle flight at angles θ_f varying from 0 to 90 degrees for regular and irregular graphs for two values of r. For irregular graphs, no clear bias is observed. For regular rectangular graphs, flight is almost entirely confined to preferred directions that vary non-monotonically with r.

Additionally, this experiment confirmed that the neighbor-selection strategy in our Jellyfish implementation did not have a distorting effect on the selection of flight angle for irregular graphs, as mentioned in Section 2.4.

3.4 Simulation Four: Motion in Extra Dimensions

In order to ensure that the algorithm behavior remained consistent for graphs emulating different numbers of dimensions, we repeated the simulation in Section 3.2 for a range of three-dimensional graphs. The graphs contained 10^5 nodes as previously, but to compensate for the increased space that the nodes could occupy, larger linking radii were tested. We explored values from $0.04 < r < 0.12$. The number of runs per graph, along with the run lengths and delays, were kept the same as for Section 3.2. The values used for n were determined using the formula for optimal particle size that was derived for the two-dimensional case.

Using three-dimensional graphs once again showed straighter motion for higher values of r (see Figure 6). Increasing the number of dimensions appeared to have no significant effect on the behavior of the algorithm, except for requiring a larger number of nodes to test.

4. Exploration

Two further qualitative investigations were performed in order to test the potential of this approach for use in future work. URLs for video shorts that show the forms of motion described in this section are listed in the appendix. These were originally generated for the 2008 Midwest NKS Conference.

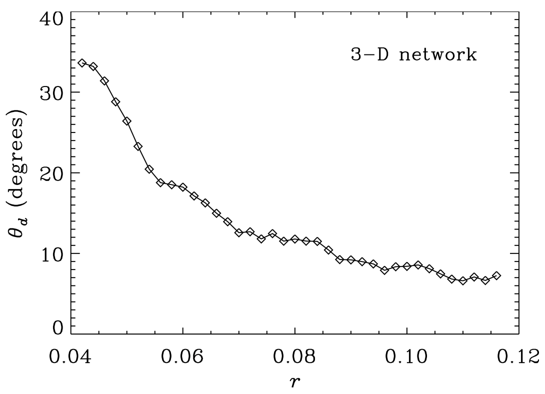

Figure 6. Change in angular deviation θ_d with linking radii r for ideal values of n. The same behavior as for two dimensions is witnessed for three dimensions, though with larger linking radii being required to compensate for the relative paucity of nodes in the sample.

4.1 Investigation One: Motion in Curved Spaces

In order for our approach to make use of the advantages of network-based systems described in Section 1, it is necessary that the Jellyfish algorithm behave consistently on graphs that do not conform to the Euclidean norm we use to test straightness. Specifically, in order to be useful for physical simulations, we require that the algorithm represent particles moving in a relativistically distorted space. However, research has yet to be done on how to best adapt dense, irregular graphs to correctly model relativistic curvature. For the purposes of illustrating the Jellyfish algorithm's potential, two simplified models are described, along with the observed flight paths that pseudo-particles take across them. In neither case do we attempt to accurately model the kind of curvature produced by gravity. Furthermore, we do not attempt to generalize the straightness metric described in Section 2.2 for these surfaces, because without a fully realized model of curvature, pseudo-particle performance cannot be adequately tested.

Arguably, the simplest way to adapt our graph generation system to model curvature is to distribute points with approximately even density on a manifold described by a smooth function and to connect them together using a linking radius as in the Euclidean case. Thus, for our first example, we produce graphs using this method to simulate a static potential well conforming to the relation $z = 1 \big/ r^2$, where r denotes radius and z denotes distance above the $x\,y$ plane.

In order to create a distribution of points on such a surface for which regions of equal area A are equally likely to contain a node, we generate random angle and area values and then apply the relation $A \propto 1 / r$ to calculate corresponding Cartesian coordinates in a three-dimensional space. For simplicity, these points are then connected by measuring their Euclidean distance in three-dimensional space, rather than measuring across the curved surface.

To test this approach, we used graphs of 100K nodes, with a scaling factor of 5 applied to the curvature function so that the effects of varying distance would be clearly visible. A linking radius of $r = 0.1$ was then used to connect the points. The mean position of a pseudo-particle with set size $n = 600$ was measured as it traversed the surface. The run lengths were of size 40, of which 10 were carried out. These simulation parameters were selected to provide clear, well-behaved examples demonstrating the possibility of orbital motion.

The trajectories traced by the pseudo-particle in each of the 10 runs are illustrated in Figure 7. In all the cases shown, smoothly curved, orbital motion can be seen. However, in order to produce results of this quality, a large pseudo-particle and a very densely connected graph were required, suggesting that this approach only yields the desired results at larger scales.

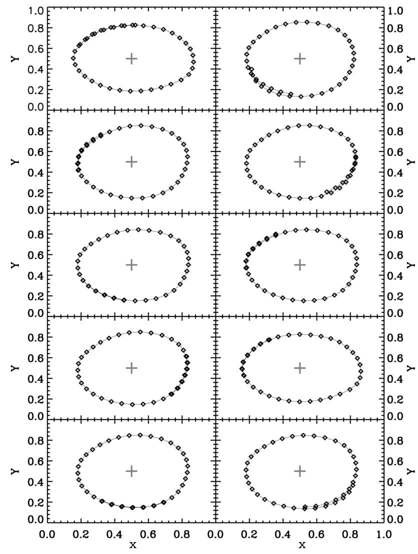

Figure 7. Sample of orbital flight paths for a Jellyfish pseudo-particle on graphs with spatial curvature generated using a linking radius applied to a surface in three-dimensional space. The center of gravity is marked with a cross for each case. Something approaching repeating elliptical motion can be seen.

While the demonstration in Figure 7 shows that well-behaved motion on curved surfaces is easy to produce, it does not depict the kind of curvature needed to model relativistic systems. Stationary objects placed in a gravity well accelerate toward the center of the well—a behavior that static curvature cannot capture. To address this, general relativity models curvature in spacetime rather than simply in space.

We can go a small way toward mimicking the kind of acceleration vector field required by general relativity by using *directed* graphs. In such a system, we simulate a potential well using a graph where the connection between nodes is dependent not only on a linking radius, but also on a "linking probability" P_l. Instead of always connecting a node to neighbors within the linking radius with probability one, each neighbor is selected for connection only with probability P_l. Given two closely spaced nodes p and q for which q is closer to the well center g, the probability that p will link to q is greater than the probability that q links to p. By then varying P_l, we can vary the degree to which a pseudo-particle will be drawn toward the center of the well.

For a given node p and neighbor q, we can determine a simple linking probability function using the following formula:

$$P_l(p, q) = \frac{1}{2} - \frac{1}{2} m \frac{|\vec{q} - \vec{p}|}{r}. \tag{4}$$

Here, r denotes the linking radius and m indicates a damping value used to moderate the scale of the effect. The symbols $\vec{p}$ and $\vec{q}$ represent the vectors from the well center g to the nodes p and q.

To test this approach, graphs of 100K nodes were used, with the linking radius set at 0.06. A damping value of 0.55 was chosen as preliminary investigations showed that it produced some of the largest stable orbits without causing the pseudo-particle to cross the graph's boundaries. The run lengths were of size 40, of which 10 were carried out.

The trajectories traced by the pseudo-particle in each of the 10 runs are illustrated in Figure 8. In all cases, smoothly curved, orbital motion can be seen, though for graphs on this scale, the orbits are not particularly stable as small changes of angle near the well center produced significant changes in orbital direction. It is anticipated that for tests on larger graphs, much smoother behavior will be visible.

It is worth noting that this method does not require changing the distribution of points from the flat Euclidean graphs described in Section 3. In terms of the placement of nodes upon a background manifold, the graphs used here are completely flat. Nevertheless, the effects of curvature are observed because the relationships between nodes have changed.

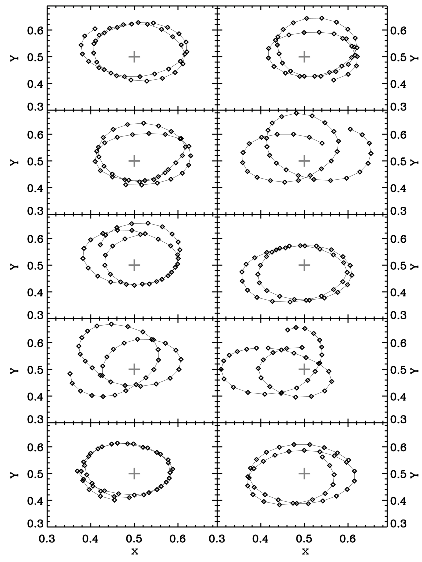

Figure 8. Sample of orbital flight paths for a Jellyfish pseudo-particle on graphs with spatial curvature generated using linking probabilities. The center of gravity is marked with a cross for each case. Though the graphs are too coarse at this scale of simulation to produce stable orbits, something approaching repeating elliptical motion can be seen.

4.2 Investigation Two: Relativistic Time Dilation

To test whether our approach could be extended to encompass motion conforming to the Lorentz metric, and therefore with special relativity, a simple technique was borrowed from Kaluza-Klein theory: the introduction of a rolled-up, or compactified, dimension. Kaluza-Klein theory was introduced in 1921 as an early attempt to reconcile electromagnetism with relativity and popularized the use of compact dimensions as possible features of physical space. The original version of Kaluza-Klein theory has since been abandoned as it failed to predict the relative strengths of the forces it sought to combine. However, the mechanisms it introduced are still in use in fields such as string theory.

Using such an extra dimension, converting the standard flight paths of pseudo-particles into relativistic flight paths is trivial. Particles are simply given freedom of motion in one extra direction. For an adaptation of the standard two-dimensional graph case, a pseudo-particle's velocity is measured by examining its motion in the x and y directions, while the "subjective time" it experiences is given by the distance traveled in the direction z. Under these conditions, the change in time-rate experienced by a pseudo-particle with varying velocity matches the requirements of the Lorentz metric exactly.

To see why this works, consider the formula for the spacetime interval given by special relativity (we assume units for which $c = 1$):

$$s^2 = t^2 - x^2 - y^2 - z^2. \tag{5}$$

By adding a dimension for s to our graph, it is easy to see that time can be described as a function of s and the dimensions of space:

$$t^2 = s^2 + x^2 + y^2 + z^2. \tag{6}$$

Each iteration of the simulation corresponds to a step on the t axis of unit length. As the Jellyfish pseudo-particle has constant velocity across the graph, some combination of the s and spatial directions that conforms to the required relativity relationship is traversed with each step. This method is not intended to capture all of special relativity, but was seen as the simplest way to produce the desired result.

It should be noted that the use of this method to encompass relativistic effects is only possible because the Jellyfish particle has rotationally invariant behavior. In this representation, frame of reference is encoded as orientation with respect to the extra dimension. Velocity can only vary smoothly if all possible angles of flight are allowed.

It should also be noted that compactifying the extra dimension in use here is not a requirement for this approach to produce the correct results, at least for particles in isolation. However, it is anticipated that future experiments designed to test relativistic behavior in conjunction with particle interaction may require it.

A basic proof-of-concept simulation employing this technique was produced and its results can be seen as a video file listed in the appendix. Subjective time experienced by a pseudo-particle is represented using pulses of brightness. Pseudo-particles traveling fast pulse slowly and those traveling slowly pulse with greater frequency.

No attempt was made here to fully model the effects of special relativity as this would have required modeled observers and was considered outside the remit of this paper. Similarly, no attempt was made to formally measure the change in subjective time as the relationship was all but guaranteed by the experimental set-up. The fact that this effect was so easily obtainable is not considered significant in isolation. It merely illustrates the potential for future work in this area.

5. Discussion

As anticipated, more densely connected irregular graphs provide a better analog for smooth surfaces than sparse ones. The observed trend suggests that perfectly straight motion should be expected as the continuum limit for the number of neighbors for each node is probed. More surprising is the discovery that for every linking radius, there appears to be an optimum particle size. Though the mechanism outlined in Section 3.1 suggests a reason for this, it does not tell us why the optimum size should be approximately the same as the average number of neighbors to a node. More work is needed to clarify what is happening here.

Observations of the Jellyfish algorithm suggest that, while well-behaved, it neither parallels the behavior of a classical particle nor a quantum-mechanical one. It cannot be considered classical as the particle is always “in many places at once”, being represented by a potentially disjoint set of nodes. Jellyfish is also not quantum-mechanical as it has no wavelength and only follows all possible paths for a distance of one arc before selecting which paths to follow and abandoning all other possibilities. In this respect, it might be said to experience wave-function collapse with every iteration. We therefore believe it should be considered a “test particle” rather than a direct analog of any natural phenomenon.

We also note the inability of Jellyfish to traverse coarse graphs in straight lines, even under apparently ideal conditions. This raises a question about the suitability of very coarse graphs to any digital physics model that employs pseudo-particles based on iterative functions over node sets. The limiting-case directed graph description of two inbound links and two outbound links per node is known to be able to emulate all more complex forms of graph. This is done by connecting nodes together in unidirectional cycles to produce rings that can be connected up to any number of other nodes. However, while such emulations are adequate for static systems, it is unclear that they provide sufficient flexibility for describing the motion of particles. Cycles of different sizes appear to disrupt the patterns of structures traversing the graph. However, more work is needed in this area before anything can be ruled out.

While there are few practical implications to this work as yet, it seems to strongly suggest that smooth-valued mathematics are not a requirement for modeling either spatial curvature or the motion of physical objects. Given that rotational invariance has been fairly straightforward to achieve, it seems reasonable to suspect that other forms of continuous symmetry might be simulated within the same descriptive paradigm.

One arena in which this work might eventually provide testable hypotheses is that of making estimates about the granularity of space. For Jellyfish, the optimum pseudo-particle size turns out to be determined by the connectedness of the graph it traverses. If this proves true for the entire class of such algorithms, it may have a bearing on the expected behavior of physical particles. For instance, the scale of electrons is well understood and their motion accurately modeled. We might be able to make an estimate of the required connectedness of the spatial graph that electrons traverse in order for the observed motion profile to be achieved. Conceivably, such a result might be used as the basis for ruling out or supporting the hypothesis that space is defined by the kind of graph described in this paper.

6. Conclusions

We have shown that dense, irregular graphs provide a viable discrete medium for modeling rotationally invariant motion for both two- and three-dimensional spaces. Regular graphs did not achieve the same flexibility, at least with respect to the class of algorithms on which our research has focused. Our investigations further suggest that dense, irregular graphs have the potential to model spatial curvature, and therefore also local spatial expansion and contraction.

We also showed that the Jellyfish algorithm provides an instance of a pseudo-particle algorithm that produces approximately straight-line motion. Though Jellyfish lacks the attributes of any naturally occurring phenomenon, it has proved extremely useful by providing us with a reliable and computationally economic test-particle for use in later studies.

Overall, these simulations suggest that we have an exciting new paradigm from which to explore the potential of the digital physics approach.

Acknowledgments

We would like to thank the following people for their support of this work.

- Genevieve Graves (UC Berkeley, Astronomy Dept.)
- Gabriel Hare (UC Santa Cruz, Physics Dept.)
- Lucas Winstrom (Cornell, Physics Dept.)
- Sean Halle (UC Santa Cruz, CS Dept.)
- Andrew Cotton-Clay (Harvard, Math Dept.)

This paper is based on a talk given at the 2008 Midwest NKS Conference.

Appendix

A. URLs for Video Samples

- Motion in two dimentions: `www.youtube.com/watch?v=Y_yCxcjYPmo`
- Motion in three dimentions: `www.youtube.com/watch?v=3w4A6m26WI4`
- Deflection by regular graphs: `www.youtube.com/watch?v=Me6K4weLS5c`
- Geodesic motion: `www.youtube.com/watch?v=n3jnKejhX-Q`

- Relativistic motion: `www.youtube.com/watch?v=ggd8Z1fZwTA`
- Quantum-like collapse: `www.youtube.com/watch?v=qUrqKhBwjGw`

Alternatively, visit the YouTube site and search for “Jellyfish Particle”.

References

[1] G. Hooft, “Quantum Gravity as a Dissipative Deterministic System,” *Classical and Quantum Gravity* **16**, 1999 pp. 3263-3279.

[2] L. Smolin, *Three Roads to Quantum Gravity*, New York: Basic Books, 2002.

[3] S. Wolfram, *A New Kind of Science*, Champaign, IL: Wolfram Media, Inc., 2002.

[4] K. Zuse, *Rechnender Raum*, Braunschweig: Vieweg, 1969.

[5] E. Fredkin, “Digital Mechanics: An Informational Process Based on Reversible Universal Cellular Automata,” *Physica D* **45**(1-3), 1990 pp. 254-270.

[6] J. Schmidhuber, “A Computer Scientist’s View of Life, the Universe, and Everything” in *Foundations of Computer Science: Potential—Theory—Cognition* (C. Freksa, M. Jantzen, and Rüdiger Valk, eds.), New York: Springer, 1997 pp. 201-208.

[7] G. Hooft, “Entangled Quantum States in a Local Deterministic Theory,” in *2nd Vienna Symposium on the Foundations of Modern Physics (2009)*, Vienna, Austria, preprint available as arXiv:0908.3408v1 [quant-ph].

[8] T. Bolognesi, “Planar Trinet Dynamics with Two Rewrite Rules,” *Complex Systems*, **18**(1), 2008 pp. 1-41.

https://doi.org/10.25088/ComplexSystems.18.4.439