Technical Proposal
Extending Stochastic Neighbour Embedding (SNE) Algorithms
Prof. Matthew Chalmers and Dr. Alistair Morrison
School of Computing Science, University of Glasgow, UK

1. Technical Approach and Justification

Since the publication of [6] in 2008, t-distributed stochastic network embedding (t-SNE) has become a commonly used—perhaps the most commonly used—method for visualising high dimensional data, especially among researchers and analysts with a statistical or machine learning background. Its core aim is to take a high dimensional data set, and create a lower dimensional (usually 2D) layout that visually conveys useful structures and patterns within the data set. This layout is often called an *embedding* in the lower-dimensional space.

Wattenberg, Viégas and Johnson, of Google Cloud, recently published an interactive overview of t-SNE, the predominant SNE technique, that is by far the most accessible and insightful overview of the method we have come across [9]. For the reader less familiar with the details of t-SNE, it serves as an ideal introduction, as it points out (and graphically demonstrates) a number of the method's characteristics, strengths and weaknesses. In the next section, we step through some of the weaknesses of t-SNE and its variants, and outline our plans for improvement that focus on addressing these problematic points. We direct this toward discussion of two issues that we understand to be of interest to ONR, namely making t-SNE incremental and making it produce better overviews of data sets. Later subsections deal more directly with future naval relevance, and the form of deliverables for this project.

1.1. Technical approach

In this section, we will set out the context and basis of our approach. Although it may fairly be said that t-SNE has a number of different shortcomings that we might address, initial discussions with ONR have raised two above others. The first is addressing the way that t-SNE, although it finds and separates clusters of related data, does not offer intuitive information about higher-level structure, for example, the size of those clusters. This also extends to global structure, e.g. the relative position of clusters cannot be relied upon to mean anything in t-SNE. The second is finding better ways to modify the visualisation on the fly so as to append new data, without the need to start the process from scratch. Here our target is an incremental t-SNE algorithm that is much more efficient than previously available.

1.1.1 Showing higher-level structure

At the core of the first of these is the way that t-SNE, like many other embedding techniques, only tries to accurately model small high-dimensional pairwise distances in the low-dimensional layout or map. This is primarily because only small pairwise distances are reliable in high-dimensional spaces—although rougher relative distances may still be useful. t-SNE focuses on minimising Kullback–Leibler (KL) divergence between the joint distributions P (describing

object similarity in high-dimensional space) and Q (object proximity in the low-dimensional embedding space). In this way, t-SNE centres on minimising an objective function that focuses on modelling high values of p_{ij} (similar objects) by high values of q_{ij} (nearby points in the embedding space).

KL divergence has an asymmetry with regard to the distribution functions it compares, and so different types of error in the pairwise distances in the low-dimensional map are not weighted equally. In particular, it imposes a large cost for using widely separated map points to represent nearby data points. This is why t-SNE focuses its effort on accurately modelling small pairwise distances, i.e., on preserving local data structure in the low-dimensional embedding. Beyond choosing KL divergence as an objective function, and as introduced in [7], t-SNE also sets the number of neighbours used for the i^{th} object — i.e. the number of objects for which p_{ij} is not assumed to be zero — to$\lfloor 3u \rfloor$, for a small and fixed value of u. If p_{ij} is zero (or infinitesimal) then x_j is not a neighbour, and so x_j has no effect on x_i in terms of establishing x_i's place in the layout (and vice versa: x_i has no effect on x_j in terms of establishing x_j's place in the layout.) These design choices add to the t-SNE's de-emphasis or avoidance of global structure.

t-SNE papers display an implicit assumption that global relationships emerge from the many local ones chained or linked together—which perhaps is the basis for the original t-SNE paper's rather thin claim that "t-SNE is capable of capturing much of the local structure of the high-dimensional data very well, while also revealing global structure such as the presence of clusters at several scales". The latter part of this claim is true but weak, in that t-SNE may show the *presence* of clusters but does not reliably show the size, separation or inter-relationship of clusters.

t-SNE sometimes focuses so much on trying to establish local structure that it can create structure in a layout when none exists in the original data. In particular, to once again use the admirably pithy phrasing of [9], when using t-SNE, "random noise doesn't always look random". [9] points out this as an issue of concern, in that "the t-SNE algorithm adapts its notion of 'distance' to regional density variations in the data set. As a result, it expands dense clusters, and contracts sparse ones, evening out cluster sizes. To be clear, this is a different effect than the run-of-the-mill fact that any dimensionality reduction technique will distort distances. […] Rather, density equalization happens by design and is a predictable feature of t-SNE. The bottom line, however, is that you cannot see relative sizes of clusters in a t-SNE plot. Moreover, since the middles of the clusters have less empty space around them than the ends, the algorithm magnifies them." In such ways, an algorithmic design choice was made by t-SNE's creators. In trying to create more space to accurately model (small) pairwise distances between similar objects, and address what they call the 'crowding problem', t-SNE emphasises or magnifies fine–grained local structure, de-emphasises and distorts slightly coarser local structure, and ignores global structure almost completely.

This stands in contrast to 'spring models' for embedding high-dimensional data, which are more often used in Information Visualisation (InfoVis) than in machine learning and statistics, and which have roots going back to several decades before t-SNE was developed. Spring models are designed to have a more even balance between local and global structure, as they rely on different metrics and methods. Simple/naïve spring models, in every iteration of the layout process, model every distance between N objects (i.e. $(N \times N-1)/2$ distances), which is generally prohibitively expensive. Each iteration therefore means a number of distance (and force) calculations that is quadratic in N.

It is common for people to choose a number of iterations that is limited to a constant maximum, making the algorithm $O(N^2)$ overall. Another common choice is to let the algorithm run until either this maximum is reached or a metric—such as the average force or velocity of the objects—drops below some threshold value. The expected run time then becomes more data–dependent, but is still basically limited to $O(N^2)$. Wary analysts may give a lot more time for the system to run, e.g. giving it a number of iterations proportional to N. In this (possibly pessimistic) case, the run time is then $O(N^3)$. We note here that standard t-SNE runs in $O(N^2)$ time, rather like a simple spring model, but more recent variants (based on Barnes-Hut methods) have reduced t-SNE to $O(N \log(N))$ overall. The latter thus outperform simple spring models, but InfoVis researchers have also explored ways to speed spring models up.

More refined spring models—including our own, e.g. [1, 2, 3]—use lightweight sampling and interpolation approaches so as to be selective about the effort put into modelling global structure. They can reduce run times significantly without sacrificing layout quality. For example, in Chalmers' 'neighbour and sampling' method [1], each object has a 'neighbour set' of fixed max size that consists of the other objects that are closest in high-dimensional space (much like those used in t-SNE). The method combines this set with a (fixed max) number of samples of non-neighbours, i.e. of the rest of the data set, in each iteration. This means that the algorithm is $O(N)$ per iteration instead of $O(N^2)$. The overall run time depends (as mentioned above) on the analyst's choice about the number of iterations. The 1996 paper introducing the 'neighbour and sampling' method was conservative in the way mentioned above, and so we claim only quadratic speed—although, in practice, we use a force/velocity threshold that assists greatly in managing run times. Overall, we'd expect a hybrid of tSNE and the 1996 algorithm, as above, to run more quickly than tSNE.

Our later refinements of the 1996 algorithm, described in [2] and [3], reduced runtimes even further, using a multistage layout process: taking a sample of a data set, using the 1996 algorithm to lay out the sample, using that layout to interpolate the rest of the data set, and then finally doing a small number of iterations on the full set so as to refine the layout. We suggest, though, that the relative simplicity of the 1996 algorithm makes it a better (in the sense of more

adaptable) starting point for the proposed work. We will improve t-SNE's handling of global structure, by using both a neighbour set (much as per t-SNE) as well as a global sample for each object (unlike t-SNE). We will also base the spring model's force calculations on KL divergence instead of using spring models' traditional Hooke's Law. This hybrid will be our initial platform for development.

We will then be able to assess different modelling parameters' effects—such as set sizes and force models—in both qualitative and quantitative terms. Depending on the success of these initial experiments, we may then go on to explore slightly more complex heuristics. For example, we could explore additional ways to productively bias the sampling, e.g. working out from an object via its neighbours, to its neighbours' neighbours, so as to and using objects within that wider neighbourhood. Another possible line of experimentation involves using local geometry metrics to visually encode different information about clusters than that shown previously in t-SNE (or in spring models, for that matter). For example, when clusters in the layout are created, we could relate those clusters back to the local geometry in the high dimensional space, and find means to communicate that local geometry. For example, a 'hypersphere' of data that fills all dimensions might be shown with one particular shape or colour in the t-SNE layout, but a flat hyperplane of data would be rendered differently. This could be done as part of the main layout process, or interactively—e.g. as lenses that a user places over a cluster. Interactive encoding may be preferable, especially if encoding of local high-dimensional geometry uses shape or position, as such as encoding will change the relative positions of objects in ways that an analyst may find clashes with the patterns of proximity of objects.

Turning back to metrics of similarity and distance, we note that t-SNE defines joint probabilities p_{ij} that measure the pairwise similarity between objects x_i and x_j via Gaussian kernels (of variances or bandwidths σ_i and σ_j respectively) that approximate the data densities around x_i and x_j respectively. As a result, in this model, probabilities p_{ij} that correspond to dissimilar input objects i and j are infinitesimal. The bandwidth σ_i is set via a binary search, such that the perplexity of the conditional distribution (which increases monotonically with σ_i) equals a predefined perplexity u. Therefore, in regions of the data space with a higher data density, σ_i will be smaller than in regions of the data space with lower density. The perplexity is interpreted as a smooth measure of the effective number of neighbours of an object, and this parameter is applied to all objects.

Wattenberg, Viégas and Johnson note that "real-world data would probably have multiple clusters with different numbers of elements. There may not be one perplexity value that will capture distances across all clusters—and sadly perplexity is a global parameter. Fixing this problem might be an interesting area for future research. […] The basic message is that distances between well-separated clusters in a t-SNE plot may mean nothing." Similarly, cluster sizes in a t-SNE plot mean nothing, and slightly different choices of u can lead to very different outputs.

Again, these effects are graphically demonstrated in [9].

Our approach to this problem stems from the observation that there is no inherent reason why the neighbour set size has to be the same for all data points, all the time—as in t-SNE—as long as varied set sizes are weighted appropriately. Therefore, we propose to economically adjust neighbour set size in an adaptive way, per object, so that neighbour set size should be higher in dense areas and/or areas of high density variation. More particularly, the ongoing sampling used to assist with global structure (mentioned above) gives us most of the information needed to do this. As iterations progress, objects' neighbour sets become progressively more accurate. Calculating the density (in high-dimensional space) of each object's neighbour set will therefore let us roughly but cheaply estimate the actual local density, thus replacing the costly binary search used as mentioned above to calculate a variance σ_i. We can then use this to adjust the neighbour set size.

We will experiment to see how this 'quick and dirty' estimate compares to the (also ultimately simple) abstraction of t-SNE's Gaussian kernel, as well as comparing it to an exhaustively calculated density estimate. If we find that this estimate of local density is insufficiently accurate, we will explore alternatives, e.g. using neighbours of neighbours, and so forth. Another approach is to make more use of the ongoing sampling of the global set, e.g. to maintain a set akin to (but larger than) the neighbor set, but used only for density estimation.

Using any of these sampling processes will mean the density estimate will gradually stabilise. Based on this estimate, we can adjust the neighbour set size—and hence only adjust the number of costly force calculations used in the optimization process where it is necessary to do so. If an object is in a dense region of the high-dimensional data set, then the neighbor set size can be higher. If, however, the object is in a sparse region (e.g. an outlier) then the neighbour set size can be lower. We can also experiment with run-time tests that, for example, check if a slight change in an object's neighbour set size leads to a major change in the forces on that object. If the estimate is not robust in this way, then the set size can be raised again.

Outliers are worthy of note when considering tSNE. Since there is only a small cost for using nearby map points to represent widely separated data points, outliers can get stuck as 'strays' among other objects. Objects nearby in the layout won't 'push away' a dissimilar data point, as the outlier is not a neighbor (and so no force calculations are done). If the outlier's neighbours are widely distributed in the layout and/or are not very similar to it—i.e. they may be the *most* similar to the stray but nevertheless are not *very* similar to it—then there will be little force applied to pull the outlier away into a good part of the layout. Some later papers refer to a generally applied repulsive force, to keep objects from being too close to each other, but this is rather like Brownian motion, i.e. there is little to directly encourage an outlier to move to the most suitable part of the layout. We expect check the randomness of the combined 'neighbour

and sampling' approach proposed above will help us such outliers economically. The chances are that an outlier that has strayed into a cluster of dissimilar objects will occasionally be sampled by a cluster member, and then get a strong 'kick'. The bigger the cluster, the more often the stray will get kicked. This should help such a stray get out of an inappropriate cluster, and make it easier for it to move to a better part of the layout.

In summary, we propose to combine the sampling approaches of our spring model algorithms with t-SNE, so as to address the first goal mentioned at the beginning of this section: improving the representation of global structure by combining the 'neighbour and sampling' spring model with t-SNE, to sample more widely but economically, to sidestep tSNE's expensive and simplistic approach to local density, and to improve upon its poor treatment of outliers.

1.1.2 Incremental layout methods

Standard t-SNE is not incremental in terms of the data set, i.e. if a data set is changed slightly, then a layout that reflects that change must start the computation from scratch. A major part of this cost occurs at the start of each run, in that the neighbour sets N_i are found in $O(uN \log N)$ time by building a vantage-point tree on the input data in the initial stages of the algorithm, and then performing exact nearest-neighbour searches with the help of the resulting tree. For a large data set, building this tree is a major cost.

Quite recently, A-tSNE—an adaptive or incremental variant of t-SNE—was published in [4], that supports gradual and steerable refinement of the embedding. It starts up with an approximation of neighbour sets, using approximated k-nearest neighbour techniques, and then increases the accuracy in areas selected by the user interactively. A-tSNE thus starts up much more quickly than t-SNE.

[4] also presents an incremental layout algorithm, as it uses this neighbour set approximation to speed up the addition, deletion and modification of objects—the latter done by removing an object, and then adding in a modified version of the object. The cost of each removal and each addition is relatively high—$O(N)$—because Pezzotti et al. check an object coming in (or going out) with regard to every other data point's neighbour set. The tree used to find neighbours is also updated, at a lower cost—$O(\log N)$—and then the overall iterative layout algorithm (basically, gradient descent) is run again in its approximate form. The suggestion is then that the analyst can see whether a new exact layout is likely to be very different from the old one. If so, the cost of that exact layout isn't changed. The method of [4] therefore does not offer a layout method that is both accurate and incremental. It offers a quicker means to see whether the cost of a full layout is justified.

We propose to use the ongoing sampling approach of Chalmers' 1996 algorithm, together with the method of controlling runtimes by cheaply tracking forces and velocities in the layout

process, to decide when to stop a layout process—both of which were described above. Given an initial data set, we can run a layout until it stabilises. Then, instead of terminating it, we can make it sleep for a short time before checking to see if new data objects are to be integrated into the layout. (The number of data objects to take in, and the time between imports of new data, would be dependent on the rate of data arrival.) For each such new object, we can then do some neighbour sampling so as to find where in the layout might be a good place to introduce the new object into the spring model. We can also initialize its neighbour set, and (if required) update the neighbours' own neighbour sets. Then we can let the spring model iterate again. The newly introduced object(s) are unlikely to be in their optimal positions, and neighbour sets may not yet be optimal overall, and so the new objects will introduce some energy into the spring model. However, we can constrain the system so that the whole layout is not unnecessarily shaken apart (which would necessitate a great deal of layout work again). Instead, the targeted injection of the new objects will inject some localised energy into the spring model, assisting each new object in finding its way to a good layout position. Once that new stable layout is achieved, the process can check for more new data objects again (or sleep for a while if none are found).

A simpler process can be used to remove objects, as long as we a make very minor addition to the 'neighbour and sampling' algorithm mentioned above. As the algorithm proceeds, neighbour sets are built up and refined via sampling. Here we need to keep a record for each object, of which other objects it is a neighbour of. Then, when an object is being taken out of the layout, we use this record to remove it from the appropriate neighbour sets. Those sets will then have a gap which can be filled as per the normal ongoing algorithm. (We could also experiment with attempts to speed that gap-filling process up, by searching the remaining neighbours' neighbour sets for a suitable object to fill the gap with.)

If an object is modified after some time of being in the layout process, in that its high dimensional data values are changed, then we could also accommodate this. A crude way would be to clear its neighbour set, and remove it from all other neighbour sets, and then treat it as though it was an insertion. However, if the high-D data change was relatively small, then it might be more efficient to check these other neighbour sets—to see if the changed object should still be in them—and only remove it if need be. It might then be that the overall adjustment of the layout suffers less disruption and re-stabilisation happens more quickly.

1.1.3 Overarching requirements and resources

We suggest that the following overarching algorithmic aims or requirements would be useful elements of this project, to help assure quality:

Structure should always be preserved when mapping data to a space of same dimension. Beyond translational, rotational, and scaling differences, relative pairwise distances should always be maintained in an N to M embedding when N=M. The examples of [9] show that t-SNE often

does not achieve this. We note that it could be that this requirement introduces difficulties when N > M, i.e. maybe good N=M performance implies bad N>M performance, but we should investigate this empirically and theoretically.

Statistical properties should be preserved, up to analytical projections. The random noise example of [9] is illuminating, but (as noted in that article) there are nuances here due to randomness in high dimensions behaving differently to what one might expect. That being said, for some distributions (e.g. Gaussian), we know analytically what the distribution should look like in a lower dimensional space, so we should be able to check that statistical structure has been retained by the mapping.

Structure should be preserved over runs. Having zero variability of layout quality over repeated runs of the algorithms is a hard requirement to satisfy, given that we are using stochastic sampling and related methods, but it's natural to desire that the degree of variability should be low. This requirement could be assessed not only with regard to repeated runs with the same parameters (e.g. neighbour set size), but repeated runs with slight variations to such parameters.

Such tests could simply compare the outputs over multiple runs, to assess whether the variance is too high, but one interesting avenue of research would be to quantify the variation over runs using the probabilistic numerics methods now being advanced in statistical research. Either way, when someone views a layout, they might have more idea of how much confidence to put into what they are seeing.

In terms of experimental resources, we propose to use the same data sets used by the t-SNE community, e.g. the MNIST dataset of handwritten letters (60k data-points, 784 dimensions), the NORB object recognition dataset (24300 data-points, 9216 dimensions), the CIFAR-10 object recognition dataset (50k points, 1024 dimensions) and the TIMIT acoustic/phonetic speech dataset (1M data-points, 39 dimensions). In addition, we would aim for a number of other larger data sets, as we suspect that sizes of 50k and 60k are well below the limits for the proposed algorithms. For example, we might then move the full NIST dataset of handwritten letters (810,000 data points) from which MNIST was taken. We also aim to use our data from recent research on app usage analysis (see http://www.softwarepopulations.com). Furthermore, we welcome suggestions from ONR as to other data sets—or kind of data sets (e.g. cardinality, dimensionality, mix of nominal/ordinal/quantitative)—that might be of interest.

We expect to use 'regular' laptops (and graphics software packages such as D3) for this kind of work, rather than more specialized computers (e.g. clusters) or specialized computer components (e.g. GPUs). Therefore, our findings will be indicative of what can reasonably be done on an everyday personal computer.

1.2. Future Naval Relevance of Basic Research

This relevance of this project stems from its fit within an area outlined within the most recent Naval S&T Strategic Plan, namely *Information Dominance – Cyber*. It fits with *decision-making superiority*, being an example of *rapid accurate decision-making for C2/CS/ISR in Big Data environments*, and *data science involving the use of analytics and reducing information down to its critical element*.

By scaling up t-SNE to larger numbers of data objects (and/or making it handle existing numbers more quickly), we respond to one of the identified drivers—*data volume, voracity and velocity necessitate improved management and analysis techniques*—and we make such visualisation techniques more suitable for "mission areas that […] contain many objects, events and activities".

Our proposals will improve the representational accuracy of the high-level structure of data sets in layouts (e.g. the relationships between clusters). Furthermore, using incremental approaches to layout instead of having to restart analysis (as per mainstream t-SNE) when some new data arrives, some data changes, or some data becomes out of date. This will allow for rapid ongoing analysis, i.e. to better support "continuous analyses of intelligence" and "persistent surveillance" with regard to streams of high-dimensional data. Overall, we see this as assisting with *decision-making superiority* based on better overview of mission areas.

1.3. Deliverables

We propose the primary deliverable be an in-depth report that presents the visualisation methods we develop, and the evidence for their effectiveness and efficiency gained from our testing and evaluation of those methods. An interim version of this report will be sent to ONR after 12 months, to serve as a year end report—and also to allow for ONR feedback that will assist in our creation of the final report soon after. The reports will primarily be a combination of the various publications that we have developed over the course of the project, with the addition of budget information, etc., but we will also outline the process of development and evaluation, so as to convey the context of the research choices taken (and not taken) with regard to the methods and their evaluation. In this way, the algorithms and their implementations will be described in detail (in terms of algorithmic design and computation complexity, and languages, libraries and deployment platforms used), and their evaluation (in terms of data sets and parameters used, and resultant metrics of layout quality and performance times) will similarly be described comprehensively.

We consider the work that we will do to be significant in an academic sense, and we aim to produce academic papers that report on the work. We will aim for at least one journal publication (such as J. Machine Learning Research or IEEE Trans. Vision and Computer Graphics) as well as conference papers—which tend to be more highly valued in this technical domain. The

premier information visualization venue is IEEE Visualization, usually held in Oct/Nov each year, but we will also look to, for example, the EuroVis conference (which takes place in Brno, Czech Republic, in June 2018) and ML-centred conferences such as ICML, with the latter reflecting the domain in which most prior (premier level) t-SNE papers have previously been published in.

Our understanding is that no direct transfer of code is acceptable to ONR, and so the report will have references to a web repository on which we will make publicly available open source software for the visualisation methods, along with example data sets, visualisation parameters, and outputs. The most likely candidate for this repository is, at present, Github—but we are happy to consider other options.

2. **Management Approach**

Two personnel are involved in this project, namely Prof. Matthew Chalmers and Dr. Alistair Morrison—both from the School of Computing Science of the University of Glasgow, and with 13 years of experience of working together on research projects.

Given such a small co-located team, the management approach is lightweight when compared to some of the larger multi-site and interdisciplinary projects the pair have been involved in previously. In work to date of similar scale and exploratory style, we have generally taken an adaptive iterative approach, undergoing short cycles of developing a new version (or versions) of our system, trying it out with appropriate data sets as quickly as possible, and then getting together to talk through the results and the consequent choices for system design refinements and further evaluation work. We meet at least twice a week to discuss ongoing plans, priorities and problems related to our analytic research, with smaller ongoing issues also discussed and resolved via email, Skype and the like. We generally use Dropbox and Github as our shared file spaces.

The given project has two major technical foci, as per §1.1.1 and §1.1.2, and we propose to work on them in parallel. After roughly two weeks of assembling data sets, and performing benchmark tests on our older visualization tools on those data sets, we will begin implementing the features previously outlined. This will initially involve changing parts of those existing tools, i.e. a long lead time is not necessary before new work starts. Our aim will then be to develop new versions of our visualization tools roughly once a week, and to be continually running instances of those tools on example data sets so as to be always gathering evidence of our prototypes' performance characteristics. We will be happy to share these ongoing results (and the discussions around them) with ONR, should ONR staff wish it, e.g. giving regular Skype discussions at a frequency of ONR's choosing. It has also been suggested by ONR, that a visit to ONR's facilities to discuss this work might be of benefit. Again, we are happy to schedule that to suit ONR.

3. **Current and Pending Project and Proposal Submissions**

The applicants are wrapping up a UK EPSRC programme grant, *A Population Approach to Ubicomp System Design*. Chalmers leads this £4M project on combining new forms of statistical, formal and visual analytics, with UX/software design for mobile applications. We are working on several funding proposals, but this one to ONR is the most advanced.

Second in the list is one on systems for diagnosis of emotional state and mental health among large cohorts of students, combining statistical modelling of temporal patterns in data from phones and sensors, along with new forms of ethical design for such systems based on ongoing consent processes. This proposal is at the outline stage, but we are aiming for a full (8 page) proposal to EPSRC at the end of October 2017. A smaller pilot of this work is the target of an intra-university funding proposal that is also at the outline stage.

Fourth in our list is one on visualisation methods related to t-SNE and spring models, but not requested by ONR from those listed in the white paper that preceded this application. This focuses on exploring alternative types of computational hardware for such methods (e.g. multicore CPUs and FPGAs), and also exploring the space of metrics beyond KL divergence, localised addition of energy to shake loose local 'tangles' in the layout, and dynamic adaptation of parameters (such as stopping criteria). As yet, this proposal only exists in draft form.

4. **Qualifications**

The personnel have many years of experience of working together as a team, most recently on the aforementioned *Population Approach* project.

Prof. Matthew Chalmers (Principal Investigator) is a computer scientist, with experience in a range of areas including information visualization, ubiquitous and mobile computing, HCI theory, and software for distributed memory multiprocessors. He has an h-Index of 39 and 6801 citations, according to Google Scholar, and has been on several committees for (among others) top flight conferences and journals relevant to this application, e.g. the paper committee of IEEE Information Visualization (later absorbed into IEEE Visualization) and the editorial board of the journal *Information Visualization*. His work on using stochastic methods to speed up spring models, reaching back to the early 1990s (some of which is referenced below), is a core part of his qualification for this project.

Dr. Alistair Morrison is a computer scientist, specialising in information visualization, mobile and ubiquitous computing, and ethical design. His PhD work, done with Chalmers, was on fast spring models that took the expected computation time down to $O(N^{1.25})$, as described in papers such as [2] and [3].

The key *qualification* of this team is therefore its specialist skills in analytics and visualization,

most particularly algorithmic optimisation and heuristic methods that let us speed up and scale up visualization methods, and yet set among a broader experience of application and infrastructure development.

In terms of *resources*, we note that the planned work is set to be carried out on everyday personal computers—in our case, the laptops we are already equipped with—and with the support of the regular IT and estates infrastructure of our university. We also have two blade servers that we use for background execution of computationally expensive tasks (and for presenting externally services, e.g. demo versions of our algorithms, on a web site devoted to this topic).

We have no existing or prospective commitments that inhibit our *compliance* with the grant conditions.

All the participants have a clean record of *integrity and business ethics*, with no criminal or legal events ever having taken place. This is shaped perhaps by our work being mostly within top flight universities—so that our activities are guided and regulated by those organisations' processes. We are therefore *qualified and eligible* to receive an award such as this.

In terms of *performance history,* we refer to Chalmers' CV which details his leadership of a series of research projects over the past decades, the largest of which are/were the £4M *Population Approach* programme grant from UK EPSRC (Chalmers was sole PI), and the £11M *Equator* Interdisciplinary Research Collaboration grant from UK EPSRC (Chalmers was Glasgow's PI). All projects were executed in accord with funder/university regulations, stayed within budget, and produced research outputs in premier publication venues. Equator was perhaps the project that required the most of Chalmers in terms of organization, as he led the *City* subproject that involved roughly 15 people spread across 5 universities. He also led Equator's cross-project theory theme. Greater detail of these past projects is given on Chalmers' university web page, and his CV. Based on this, we suggest that this project is well within Chalmers' and the University of Glasgow's capabilities with regard to *organization, experience, accounting and operational controls, and technical skills*.

5. References

1) M. Chalmers, "A Linear Iteration Time Layout Algorithm for Visualising High-Dimensional Data", *Proc. IEEE Visualization*, San Francisco, Oct.-Nov. 1996, pp. 127-132.
2) A. Morrison, G. Ross and M. Chalmers, "A Hybrid Layout Algorithm for Sub-Quadratic Multidimensional Scaling", *Proc. IEEE Information Visualisation*, Boston, 2002, pp. 152-160.
3) A. Morrison and M. Chalmers, "A Pivot-Based Routine for Improved Parent-Finding in Hybrid MDS", *Information Visualization* 3(2), 109-12, 2004.
4) N. Pezzotti, B. Lelieveldt, L. van der Maaten, T. Höllt, E. Eisenmann, and A. Vilanova, "Approximated and user steerable t-SNE for progressive visual analytics," IEEE Trans. Visualization and Computer Graphics, May 2016.

5) N. Pezzotti, T. Höllt, B. Lelieveldt, E. Eisenmann, and A. Vilanova, "Hierarchical stochastic neighbor embedding," Proc. EuroVis, 2016.

6) L. van der Maaten and G. Hinton, "Visualizing data using t-SNE," *Journal of Machine Learning Research*, vol. 9, pp. 2579–2605, 2008.

7) L. van der Maaten, "Accelerating t-SNE using tree-based algorithms," *Journal of Machine Learning Research*, vol. 15, pp. 3221– 3245, 2014.

8) C. Stolper, A. Perer, and D. Gotz, "Progressive visual analytics: User-driven visual exploration of in-progress analytics," IEEE Trans. Visualization and Computer Graphics, vol. 20, no. 12, pp. 1653–1662, Dec 2014.

9) M. Wattenberg, F. Viégas and I. Johnson, "How to Use t-SNE Effectively", *Distill*, 2016. http://doi.org/10.23915/distill.00002