https://doi.org/10.2352/EI.2022.34.1.VDA-414

Visualizing Semantic 3D Object Clouds

Bola Okesanjo and Stephen Brooks; Dalhousie University; Halifax, Nova Scotia, Canada

Abstract

3D object clouds, first introduced by Hong and Brooks, visualize the pairwise similarity between a set of objects and a central object of interest. This similarity is used to determine the position of each object within the cloud. However, this does not capture the semantic relationship of all the objects and the lack of consistency may reduce the expectation of finding an object when performing visual search. To generate a semantic 3D object cloud, we define and subsequently minimize an energy function that captures the pairwise similarity amongst all objects within the cloud. The energy is minimized using several statistical machine learning techniques and we show that the generated layouts from such techniques outperform those of other algorithms on a variety of metrics for evaluating layouts.

INTRODUCTION

When querying Computer-Aided Design (CAD) models from commercial search engines results they are typically presented as a few dozen objects in a grid of discrete rows and columns. This arrangement of search results however can make it tedious to find and compare different sets of result. An alternative to representing such results is as a cloud, where all the models are clustered onto the screen. Such a cluster can take on arbitrary shapes and allow for a preferable arrangement of the search results. Tightly packing the cluster also maximizes the use of the limited display space and it has been shown to facilitate the faster recognition of 3D models [21].

Object clouds, as their name suggests, are analogous to word clouds in that both are a compact visual summary of various items. Like word clouds, the items in an object cloud also have varying sizes which indicates their degree of similarity to a given object of interest. This similarity is based on the how many visual features are shared between a pair of objects. Objects with the highest degree of similarity to an object of interest are larger and objects with lesser degree of similarity are smaller.

To facilitate faster recognition of objects, the object of interest or the object with the most degree of similarity is placed at the center of the cloud. This however need not be the case all the time. Current algorithms for visualizing object clouds attempt to place objects at a distance that also reflects their similarity to the central object of interest. Objects that are most similar to the object of interest are placed closer to the center of the cloud and objects that are less similar are placed farther away. This ordering has been shown to facilitate faster recognition of objects than a random or a grid-based ordering [21]. However, utilizing the pairwise distance between each object and the object of interest does not create a semantically accurate ordering within the entire cloud. To create a semantically accurate ordering, the pairwise distance amongst *all* the objects in the cloud needs to be used instead.

To create semantically accurate and compact clouds, we approach the problem of constructing an object cloud from an optimization perspective. To do so, we formulate an objective function that represents the energy within an object cloud and encapsulates the requirements of a semantic object cloud:

- Objects should be compactly placed together without overlaps
- Similar objects should be closer and dissimilar objects should be farther

The objective function is then optimized using several strategies such as gradient descent and majorize-minimize which result in several algorithms for visualizing object clouds. Using the energy function, we can monitor the performance of our algorithms and for some, terminate them when they cease to minimize the energy within the cloud. This however contrasts with other cloud visualization algorithms that rely on a set number of iterations from the user which may be insufficient to properly minimize the energy.

In order to fulfil the above objectives, this work proposes a real-valued function to describe and quantify the aesthetic of object clouds. We also propose algorithms for the generation of object clouds based on the optimization of the proposed function. Furthermore, we evaluate and contextualize the performance of our proposed algorithms in relation to existing algorithms for the generation of object clouds. Recalling that our target application is to browse a similar number of objects that are typically found on commercial webpages (which generally present a few dozen objects), we made a comparison of clouds of up to 100 objects. For the same reason, we focussed on layout quality rather than computation speed, since we are only browsing a few dozen objects in practice.

We begin with a discussion of related work, including graph drawing dimensionality reduction, word clouds and the original 3D object cloud paper. We then introduce semantic 3D object clouds which incorporates dimensional reduction, a graph structure and energy minimization strategies. We then discuss our experimental evaluation, followed by conclusions and limitations.

RELATED WORK

Graph Drawing

Graph drawing algorithms use the information contained within a graph to generate suitable layouts. But graph drawing is a very large area of research that incorporates many types of graphs such as hierarchical graphs and orthogonal layouts [49]. We will restrict our background discussion to force based and energy-based graphs. Force-directed graphs first applies a set of spring forces to move the edges and vertices of the graph, while energy-based graphs optimize an energy function to move the vertices of the graph into place.

These graphs are typically undirected graphs with a 2D or multi-dimensional layout. A layout can be described as suitable if it has minimal energy or if it both exhibits some symmetry and the pairwise vertex distances are close to some constant [24]. These graph drawing algorithms start with an initial layout of the graph - this can be random, circular or any other layout. From the initial layout, both the vertices and edges of the graph are moved until a stop criterion is achieved. This criterion includes the number of iterations, net change in forces or net change in the energy of the graph.

While drawing an undirected graph is thought of as matching the pairwise vertex distances to some constant, the matching process itself can also be thought of as a form of energy minimization whereby the energy corresponds to the discrepancy between the geometric pairwise distances and said constant. Based on this principle, energy-based graph drawing algorithms generate their layout by optimizing a given energy function for a graph. This energy is defined over the vertices of a graph and the optimization can be achieved using local methods such as gradient descent or global methods like simulated annealing [12][40].

Force-directed graph drawing algorithms, as the name suggests, are a class of algorithms that use a set of forces to generate undirected graphs. These forces are typically modelled after spring forces and as such are either attractive or repulsive nature. The Fruchterman-Reingold method is a well-known force-directed algorithm that generates an undirected graph, using a set of spring forces that are modeled after Hooke's law [15].

Dimensionality Reduction

Dimensionality reduction is the process of representing high dimensional data into a lower dimension such that important patterns within the data are preserved. There are typically 2 types of patterns that are preserved: global and local patterns [17]. These patterns are typically found without the aid of human labels in what is known as unsupervised learning and as such it makes dimensionality reduction an indispensable tool in information visualization. Examples of dimensionality reduction techniques include non-linear approaches, such as t-SNE and UMAP, as well as Principal Component Analysis (PCA) and Multi-Dimensional Scaling (MDS) [50][6]. One problem with classical MDS is that it places too much emphasis on larger pairwise distances at the expense of smaller pairwise distances [17]. This means that larger distances are mostly accurate whereas smaller distances tend to be inaccurate. In fact it does not use as much as a third of all small distances [10][19].

In an attempt to solve the shortcomings of classical MDS and place equal emphasis on small pairwise distances, metric MDS adds weights to the MDS objective. These weights are typically the inverse of the original pairwise distances so that as much focus is given to matching smaller distances as is given to matching larger distances. Matching smaller distances allows us to capture patterns between higher dimensional points that are close together. Unlike classical MDS where the objective is solved analytically, the objective in metric MDS cannot be solved analytically because the weights perform a non-linear transformation of the points, Q. Instead, gradient descent or an iterative process known as majorization is used to optimize the stress objective [17][13].

Scaling by MAjorizing a COmplicated Function (SMACOF) is a popular majorization algorithm that is used to optimize a stress objective Majorization itself is an optimization technique that is used to solve a complicated function by solving a simpler surrogate function. Using the Cauchy-Schwartz inequality, we can create a simple surrogate function for the stress objective [14].

Word Clouds

3D object clouds were inspired by word clouds which is a visualization of a set of words whereby the font size represents some weighting like their frequency within a given text corpus. Word clouds became popular as a form of visualization when social sites such as flickr and del.icio.us started associating web resources with keyword metadata. These words summarized the content and allowed users to navigate other resources on the sites [36][35]. Recently, word clouds provide a visual statistical summary of a text corpus and are generated using specialized algorithms and well-known software such as Wordles [20].

There are several types of word cloud generation algorithms. They include both the random and semantic word cloud algorithms [2]. Random word cloud algorithms generate their layout without any emphasis on the semantic relationship between the words. Semantic word cloud algorithms however generate their layout based on these relationships. Having such structure along with the frequency of the words improves the statistical summary provided by the word cloud. A well known random algorithm is Wordle and well known semantic algorithms are the Context Preserving Word Cloud (CPWC) and seam carving [11][45].

Wordle is a popular algorithm that generates random word clouds using an Archimedean spiral. After extracting the relevant words that summarize a text, each word is successively placed at a random position on the canvas. During placement, if a word collides with any other word on the canvas, it is moved along an ever increasing spiral until it no longer collides with another word or until it is no longer on the canvas [37].

In order to generate semantic word clouds, most algorithms first utilize some kind of dimensionality reduction to capture the semantic relationship between high level representations for a set of words. Such relationship exists when the text in a corpus can be faithfully represented in some form such as vector embeddings. Most algorithms use classical MDS for dimensionality reduction but other algorithms like t-SNE can be used as well [2][33].

Classical MDS is used to generate an initial semantic layout for most algorithms. However the 2D positions that are generated by MDS does not take into account the size and geometry of the words. As a result, when the words are initially placed, they are not compact and in most cases may overlap. The crux of semantic word cloud algorithms is in how they refine this initial layout. Various algorithms take different approaches to refining the layout of the cloud while preserving its semantic order.

The context preserving algorithm is a semantic word cloud visualization algorithm that uses a set of forces to generate a word cloud. These forces are: attractive, repulsive and planar forces. The algorithm is a force-directed algorithm similar to the Fruchterman-Reingold algorithm. A major drawback of this method is the use of the Delaunay graph. By flipping the position of the words in order to create each triangle within the graph, some semantic order is lost. This in turn results in vastly different word clouds that get generated with the insertion or removal of new words.

As the name suggest, the seam carving algorithm is inspired by seam carving which is used to resize an image while maintaining the important parts of the image [45]. Seam carving operates by removing or adding connected rows or columns of pixels that have little importance in the image [1]. For word cloud generation, such pixels correspond to empty spaces within the initial layout. As with the context preserving algorithm, classical MDS is used to generate an initial layout and a repulsive force-directed algorithm is then applied in order to separate overlapping words. After separating the words, the seam carving algorithm strips the empty spaces between them to generate a compact word cloud. Unfortunately, in many cases, there are no connected paths between words that run through one end of the canvas to another. In such cases, the algorithm leaves empty regions between the words and a sub-optimal word cloud is generated.

3D Object Clouds

3D object clouds [21] and other pseudo-random placement algorithms are algorithms that generate a seemingly random layout. In most cases the layout is generated from a breadth-first search of discrete positions within the layout. These positions correspond to cells in a row-column grid that overlays the canvas for the object cloud.

In order to generate a cloud, the vector embedding for each object is first generated and its distance from that of the object of interest is computed. Next the 2D layout for the cloud is divided into a grid and the central object of interest is placed at the center of the grid. Every other object is then sorted according to their vector distance from the object of interest and placed at the next available square in the grid. The next available square is found by performing a breadth-first search of all the discrete positions with the center square as the root. If an object is placed in a square but it collides with an already placed object, it is rotated along an Archimedean spiral until it no longer collides with any object [21]. The new grid position for the object is marked as occupied and the process is repeated until all the objects are placed within the layout.

But as discussed previously, emphasizing *only* the pairwise relationship between all objects within the cloud and the object of interest does not create a proper semantic order within the entire cloud. To generate a more complete semantic order, the pairwise relationship amongst all the objects in the cloud needs to be computed, which is the focus of this work.

SEMANTIC OBJECT CLOUDS

Energy-based Clouds

As discussed in the previous chapter, using dimensionality reduction methods such as MDS to visualize object clouds is restricted to projecting high dimensional points unto a 2D canvas without accounting for the geometry and size of the items that said points represent [31].

We can modify the MDS algorithm so that performing dimensionality reduction simultaneously determines the proper position of various items within the cloud. To do this, we re-express the metric MDS objective for dimensionality reduction as a graph-drawing energy function. When such functions are minimized, we obtain a resulting force-based graph drawing algorithm. In this section we discuss the construction of this energy function, as well as how it serves as a qualitative measure for an object cloud. We also discuss how gradient descent and majorization can be used to optimize it, as well as provide an algorithmic implementation for generating such a cloud.

Metric Multi-Dimensional Scaling

One problem with classical MDS is that it places too much emphasis on larger pairwise distances at the expense of smaller pairwise distances [17]. This means that larger distances are mostly accurate whereas smaller distances tend to be inaccurate. In fact, it does not use as much as a third of all small distances [10][19].

In an attempt to solve the shortcomings of classical MDS and place equal emphasis on small pairwise distances, metric MDS adds weights to the MDS objective. These weights are typically the inverse of the original pairwise distances so that as much focus is given to matching smaller distances as is given to matching larger distances. Matching smaller distances allows us to capture patterns between higher dimensional points that are close together. The new objective formed is referred to as Stress, *S*, and it is expressed as

$$S = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} w_{ij}(d_{ij} - \|p_i - p_j\|_2)^2 \tag{1}$$

where $d_{ij} = \|q_i - q_j\|_2$ and $w_{ij} = 1/d_{ij}$.

The pairwise distances, d_{ij}, however need not be represented by Euclidean distances. They can be represented by any distance function. In order for MDS to capture local patterns within the neighbourhood of a point q_i, the Euclidean distance has to be replaced with the geodesic distance. This is because Euclidean distance corresponds to a distance on a straight line and on curved manifolds such as the one illustrated in figure 1, this results in smaller pairwise distances for far away points on the manifold.

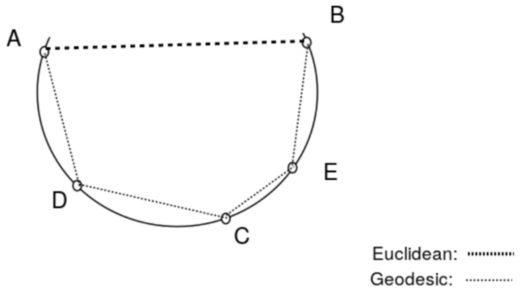


Figure 1. Geodesic & Euclidean distances between points A and B on a curve

The geodesic distance on the other hand is the shortest-path distance between any pair of points, q_i and q_j, on a graph. If a manifold is represented as a graph, the geodesic distance can find the appropriate distance between any pair of points on the manifold. This means that on curved manifolds, far away points will always have a larger pairwise distance than neighbouring points. Since the shortest-path distance computation relies on the distance between neighbouring points, MDS is able to capture local patterns when using the geodesic distance [39].

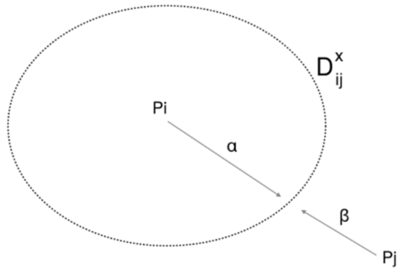


Figure 2. Direction of α and β on a given point p_j

Energy Formulation

In order to express the energy in equation 1 as a force-based graph-drawing algorithm, we change the weighting, w_{ij}, into a force-based weighting and replace the original pairwise distances, d_{ij}, with the sum of the radii for a pair of objects. In order to replace the weight, w_{ij}, in equation 1 we decompose it into

$$w_{ij} = (1 - \delta_{ij})w_{ij} + \delta_{ij}w_{ij} = (1 - \delta_{ij})\beta_{ij} + \delta_{ij}\alpha_{ij} \tag{2}$$

where β_{ij} is an attractive weighting, α_{ij} is a repulsive weighting and δ_{ij} is an indicator function. We also express the pairwise distances, d_{ij}, as

$$d_{ij} = r_i + r_j \tag{3}$$

where r_i and r_j are the radii for a pair of objects. The new pairwise distance indicates that we want all the objects to be adjacent each other. Normally this will cause all the objects within the cloud to collapse on each other. However, the force-based weighting will prevent this from happening and instead results in a tight packing of the objects within the cloud. From the above, the energy in equation 1 becomes

$$E = \sum_{i<j}^{n} ((1-\delta_{ij})\beta_{ij} + \delta_{ij}\alpha_{ij})(d_{ij} - \|p_i - p_j\|_2)^2 \quad (4)$$

where the indicator function, δ_{ij}, evaluates to

$$\delta_{ij} = \begin{cases} 1 & \text{if } \|p_i - p_j\| < d_{ij} \\ 0 & \text{otherwise} \end{cases} \quad (5)$$

The first weighting, β_{ij}, can be thought of as an attractive weighting between points, p_i and p_j while the second weighting, α_{ij} , is the repulsive weighting between the pair of points. Since p_i and p_j represent the center of a pair of objects and d_{ij} represents the distance at which the pair of objects become adjacent without overlapping, the weightings control how fast or slow different pairs of objects become adjacent to one another. When $\|p_i - p_j\| > d_{ij}$, the objects represented by p_i and p_j are far apart and either is attracted to the other at a speed of β_{ij} until $\|p_i - p_j\| = d_{ij}$. Conversely when $\|p_i - p_j\| < d_{ij}$, the objects represented by p_i and p_j are too close and either is repelled away at a speed of α_{ij} until $\|p_i - p_j\| = d_{ij}$. When $\|p_i - p_j\| = d_{ij}$, the weightings have no effect and there is no change in the position of either p_i or p_j.

Under the typical MDS objective both weightings are equal and cancel themselves. In such a scenario the pairwise distances, d_{ij}, must reflect actual distances between all the objects or the objects will collapse unto themselves. However, when $\alpha > \beta$, a given layout has a lot of space with a few overlapping objects and when $\alpha < \beta$, the layout is more compact but with a lot of occluding objects.

After decomposing the weights, we construct a graph over the high dimensional points. This graph allows us to establish a pairwise ordering of the objects which is lost after replacing the original pairwise distances, d_{ij}. In the next section we will discuss what kind of graph is needed to create a semantic object cloud. Using the high dimensional graph, we split the objective into adjacent pairs of points, N_1, and non-adjacent pairs of points, N_2, such that we have

$$E = \sum_{i<j}^{N_1} ((1-\delta_{ij})\beta_1 + \delta_{ij}\alpha_1)(d_{ij} - \|p_i - p_j\|_2)^2 + \sum_{k<l}^{N_2} ((1-\delta_{kl})\beta_2 + \delta_{kl}\alpha_2)(d_{kl} - \|p_k - p_l\|_2)^2 \quad (6)$$

By setting attraction, $\beta_2 = 0$ for the non-adjacent pairs of objects and making the repulsion $\alpha_1 = \alpha_2$ for both adjacent and non-adjacent pairs of objects, we end up with an energy function that when differentiated, results in a force-based graph-drawing algorithm. The function can succinctly be expressed as

$$E = \sum_{i<j}^{N} (A_{ij}(1-\delta_{ij})\beta + \delta_{ij}\alpha)(d_{ij} - \|p_i - p_j\|_2)^2 \quad (7)$$

where A_{ij} is another indicator function that represents the adjacency of any pair of points. In order to create non-overlapping object clouds, especially as the number of objects increases, we find it important to set $\beta < \alpha$. When the number of objects is relatively small, $\beta = \alpha$ produces non-occluding objects but as the number of objects increases, so does the number of occlusions. Setting $\beta > \alpha$ on the other hand results in occluding objects regardless of the number of objects.

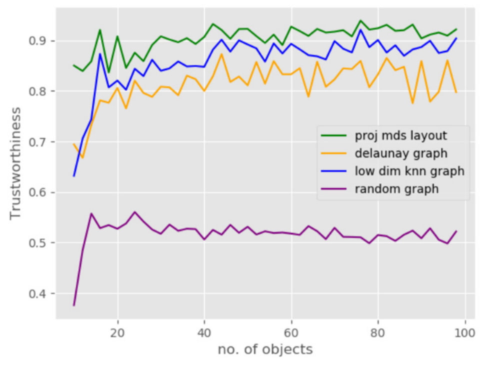
Graphical Structure

In order to create a semantic object cloud, we use a K-Nearest Neighbour (K-NN) graph. For any vector, v_i, in a given space, the K-NN graph finds the k closest vectors, $\{w_1, ..., w_k\}$ and constructs an edge between v_i and each of the vectors. This graph is however directed and unconnected whereas the graph for an object or a word cloud needs to be undirected and connected in order to exert the proper forces amongst the nodes and compact them all. An unconnected graph will have items that are not attracted by any other items nor repelled by all the other items, thus creating excess space within the cloud. To convert a graph into an undirected and connected graph, we sum the adjacency matrix of the graph with its transpose and take the non-zero entries as the edges.

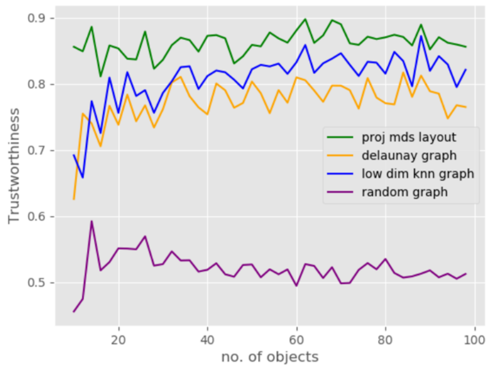
To determine the appropriate number of neighbours for the K-NN graph, we choose the smallest number of neighbours, k, that is necessary to form a connected graph. We find that the smaller the number of neighbours, the easier it is to compactly place an object with all its neighbours, thereby reducing the energy for the object. If the number of neighbours is high, an object cannot be placed with all its neighbors because there is a limit to the number of non-overlapping objects that can be placed around a given object. Hence the energy of the object in question will increase.

The K-NN graph is typically used in many dimensionality reduction algorithms because it preserves the local ordering amongst high dimensional points [39][4]. To that end, we find that it helps ensure a semantically accurate ordering amongst high dimensional points. However, we find that the Delaunay graph which is typically used to construct word clouds does not ensure a semantically accurate ordering due to the flipping of points that may be required when constructing a simplex for the triangulation. Furthermore, the Delaunay graph cannot be used for high dimensional points because in order to construct such a graph, the number of points needs to exceed their dimensionality. However, for both word and object clouds, the dimensionality of the vector embeddings can sometimes exceed the number of items to be visualized.

In figure 3, we illustrate the advantage the KNN-graph has over the Delaunay graph for both semantic accuracy and dimensionality reduction. To measure the semantic accuracy, we use the trustworthiness metric [18][41]. This measures the amount of high dimensional neighbouring points that are present in the neighbourhood of each 2D point. Neighbouring points from the high dimension that are missing from the neighbourhood of a 2D point reduces the trustworthiness of a projection. Conversely, neighbouring points from the high dimension that are present in the neighbourhood of a 2D point increases the trustworthiness. We compare the trustworthiness of a 2-dimensional Delaunay, K-NN and random graphs that are formed from an initial MDS projected layout. We also include the trustworthiness of the MDS layout as a baseline.



(a) ModelNet 40



(b) Princeton shape benchmark

Figure 3. Trustworthiness of 2D graphs and MDS on different datasets with different numbers of objects.

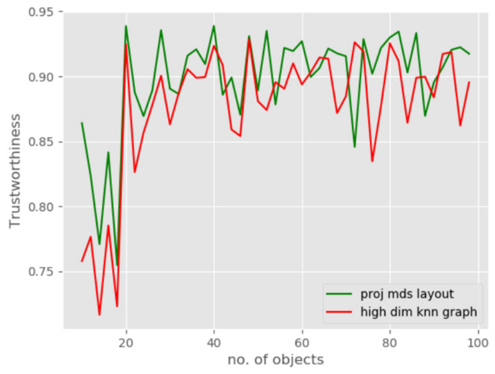
From figure 3, we can see that the classical MDS projected ordering has a high trustworthiness which indicates that the initial layout is quite similar to the high dimensional ordering of the points. However, by applying both the K-NN, Delaunay and random graphs to draw the layout, the trustworthiness decreases. This is due to the fact that some information about the actual closeness of neighbouring points is lost during the projection and graphs like the Delaunay graph change some semantically correct edges during its construction. In figure 4 we show how applying a KNN graph to the high dimensional points themselves simply circumvents these problems and increases the trustworthiness of an initial layout.

Measuring Object Clouds

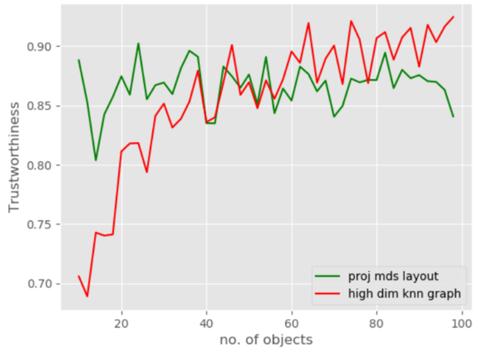
Formulating an objective function for object clouds allows us to measure the quality of a cloud. By defining a semantic or random graphical ordering over the set of objects, we can measure how well said objects are compactly placed relative to each other within a layout. We will briefly illustrate how the energy of a layout quantifies its appeal, as well as what to note when using such qualitative measure.

Consider the pair of objects in the figure 5. The farther apart their distance, L_{ij}, the larger the amount of energy, E, within the layout. Conversely suppose we have an overlapping pair of objects as in the figure 5. If we assume the energy of this layout is minimal because the distance, $l_{ij} < L_{ij}$, then the objective function will tend towards this configuration for a pair of objects. However, since this configuration is as equally undesirable as the previous configuration, we scale l_{ij} - using L_{ij}/l_{ij} - so that the value of the energy is as large as that of the previous configuration. Note that compared to an overlapping configuration, we do not need to scale a distant configuration. That being said, we are left with 2 configurations in which the distance, and subsequently the energy, E, is zero: when the objects are compactly placed side by side and when they completely overlap. The configuration in which the objects completely overlap is a local minimum that can only be escaped by adding noise to their positions. If however we ignore this case, then we see that the configuration with the most appeal - compact placement - has no energy.

In order to determine the proximity of a pair of objects, L_{ij} or l_{ij}, the energy in equation 7 calculates the position of their centers, $\|p_i - p_j\|$, relative to the size of the objects, $d_{ij} = r_i + r_j$ such that L_{ij}, $l_{ij} = d_{ij} - \|p_i - p_j\|$. When $\|p_i - p_j\| < d_{ij}$, the pair of objects are overlapping and a force $\alpha > \beta$ is applied to the overlapping distance, l_{ij} in order to compute the energy value. The indicator function in equation 5 determines which of the forces to apply and subsequently which distance, L_{ij} or l_{ij} is in effect. We can set the weight β to any positive real value but in order to scale l_{ij} and prevent the energy function from settling into an overlapping configuration, $\alpha > \beta$.



(a) ModelNet 40



(b) Princeton shape benchmark

Figure 4. Trustworthiness of MDS and a high-dimensional KNN on different datasets with different number of objects.

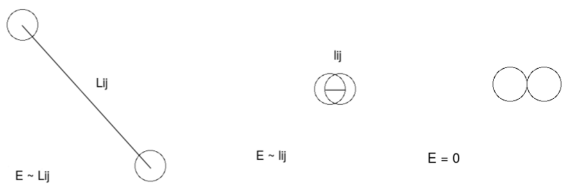


Figure 5. Distance in relation to the energy for a pair of objects.

Since computing the energy involves the squared distance between pairs of objects and the weightings are positive real numbers, for an object cloud the energy is always a positive real number. Ideally for every cloud, we would want this number to be as close to 0 as possible without pairs of objects completely overlapping but this is not always possible as the number of objects increases. This is because as the number of objects increases, so does the number of configurations. Many of these configurations are sub-optimal and the objective function may not be able to escape one of such local optima. Hence as the number of objects increases, we may achieve a state of minimal energy rather than a state with no energy.

Energy Minimization

In order to minimize the energy function of an object cloud, we consider several optimization methods that have been applied to multi-dimensional scaling and machine learning problems.

Gradient Descent

The algorithm represented by equation 7 can be converted into a force-based algorithm by taking the negative gradient of the energy, E. Taking the gradient with respect to p_i, we have

$$\frac{\partial S}{\partial p_i} = \sum_j (A_{ij}(1-\delta_{ij})\beta + \delta_{ij}\alpha)(\|p_i - p_j\|_2 - d_{ij})\frac{(p_i - p_j)}{\|p_i - p_j\|_2} \tag{8}$$

where $(p_i - p_j) / \|p_i - p_j\|_2$ indicates the direction in which the center, p_i, of an object should move and $(d_{ij} - \|p_i - p_j\|_2)$ indicates by how much the center should be moved for an object in question to be placed compactly amongst its neighbours. Using the gradient, we can update the position of p_i as

$$p_i^{(t+1)} = p_i^{(t)} - \eta\frac{\partial S}{\partial p_i} \tag{9}$$

However, the problem with this update algorithm is that it converges slowly relative to second-order optimization methods like the Newton-Raphson method and it has a higher likelihood of becoming stuck at a local minimum [23]. Applying the Newton-Raphson method as is done in the Kamada-Kawai algorithm requires the calculation of a Hessian which can be a tedious approach and in some cases, a semi-positive definite Hessian may not exist. In order to avoid this while guaranteeing faster convergence to a global minimum, we further adopt two optimization strategies to minimize the energy function: random reshuffling and majorization.

Random Reshuffling

Random reshuffling is a form of gradient descent in which the training set is shuffled at each iteration before taking the gradient. In our case, the training set, P, consists of pairs of objects whose initial order is permutated at each update iteration. We can express this as

$$p_i^{(t+1)} = p_i^{(t)} - \eta\frac{\partial S_{\sigma(P,t)}}{\partial p_i} \tag{10}$$

where $\sigma(P, t)$ represents the permutation of P at iteration t. While the convergence of the random reshuffle is chaotic in nature, its convergence rate is greater than that of the normal gradient descent and can be even close to quadratic in some cases [7]. Additionally, shuffling the order of the training set can allow us to escape local optima within the energy function.

Majorization

Given that the energy for object clouds can be expressed is a form of metric MDS, majorization can be used to minimize it. A significant advantage of majorization is that it guarantees a set of non-increasing energy values that allows us to stop the minimization process when there is little to no change in the energy values. In other words

$$f(x^t) - f(x^{t+1}) \leq \epsilon \tag{11}$$

At which point we can be certain that the energy of the object or word cloud is minimal and has the highest aesthetic value before ending the minimization process. This is particularly useful because most graph drawing algorithms such as the Fructerman-Reingold and the context preserving algorithm as well as the techniques mentioned above minimize the energy in a cloud within a set number of iterations. This may result in a sub-optimal configuration of objects within the cloud if the number of iterations is inadequate or if the number of iterations is too long that the energy values begin fluctuating at the valley of the energy function.

Scaling by MAjorizing a COmplicated Function (SMACOF) is a popular majorization algorithm that is used to optimize the stress objective in equation 1. Majorization itself is an optimization technique that is used to solve a complicated function, $f(.)$, by solving a simpler surrogate function, $g(., .)$. For a given point, y, and a minimizer, $x*$, the surrogate function needs to satisfy the following sandwich inequality

$$f(x^*) = g(x^*, x^*) \leq g(x^*, y) \leq g(y, y) = f(y) \tag{12}$$

Using the Cauchy-Schwartz inequality, we can create a simple surrogate function for the stress objective. To do this, we begin by rewriting equation 1 as:

$$S = \sum_{i<j}^{n} w_{ij}d_{ij}^2 + \sum_{i<j}^{n} w_{ij}\rho_{ij}^2 - 2\sum_{i<j}^{n} w_{ij}d_{ij}\rho_{ij} \tag{13}$$

where $\rho_{ij} = \|p_i - p_j\|_2$. Using the definition of,

$$\|p\| = \frac{p^\top p}{\|p\|} \tag{14}$$

we further express equation 13 in matrix form such that:

$$f(p) = \frac{n(n-1)}{2} + \text{tr } P^\top VP - 2 \text{ tr } P^\top B(P)P \tag{15}$$

where, P, are the coordinates of the points in the lower dimension and the matrices V and $B(P)$ are weighted Laplacian such that

$$V_{ij} = \begin{cases} -w_{ij} & \text{if } i \neq j \\ \sum_{i \neq j} w_{ij} & \text{if } i = j \end{cases} \quad B(P)_{ij} = \begin{cases} -w_{ij}d_{ij}/\rho_{ij} & \text{if } \rho_{ij} > 0 \\ 0 & \text{otherwise} \end{cases} \tag{16}$$

The last term in equation 15 is the what is most important for constructing the surrogate function. Using the Cauchy-Schwartz inequality

$$\|x\| \geq \frac{x^\top p}{\|p\|} \tag{17}$$

for some vectors, p and x [16]. We can construct a surrogate function, $g(x,p)$, that is expressed as

$$g(x,p) = \frac{n(n-1)}{2} + \text{tr } X^\top V X - 2 \text{ tr } X^\top B(P)P \tag{18}$$

From equation 17, the following inequality holds

$$X^\top B(X)X \geq X^\top B(P)P \tag{19}$$

Hence for a minimizer, x, $f(x) \leq g(x, p)$ and the sandwich inequality in equation 14 holds [14]. We can find this minimizer by setting the derivative of the $g(.,.)$ with respect to X as zero and then solving for X. This results in the solution

$$X = V^+ B(P)P \tag{20}$$

where V^+ is the pseudo-inverse of V.

This solution is also known as the Guttman transform of the points, P [14]. On each iteration of the majorization algorithm, the points P are replaced with X and the above transformation is repeated until $f(x) - f(p)$ is less than some tolerance value, ϵ.

In order to majorize a function, $f(.)$, we need to construct and minimize a surrogate function, $g(.)$, such that the sandwich inequality in equation 14 holds and the minimum of $g(.)$ is the same as that of $f(.)$. We adopt the surrogate function that is used in equation 14 with the minor modification that the weight, w_{ij}, is no longer constant and can be expressed as

$$w_{ij}(p) = A_{ij}(1 - \delta_{ij}(p))\beta + \delta_{ij}(p)\alpha \tag{21}$$

where $w_{ij}(p)$ are the weights for a particular configuration P such that $\delta_{ij}(p)$ is the activation function for a set of positions p_i and p_j that belong to said configuration.

Hence the surrogate $g(x,p)$ can then be expressed as

$$g(x,p) = \sum_{i<j}^{n} w_{ij}(p)d_{ij}^2 + \text{tr } X^\top V(W_p)X - 2 \text{ tr } X^\top B(P, W_P)P \tag{22}$$

where $V(W_p)$ and $B(P, W_p)$ are the Laplacian matrices computed from the weights $w_{ij}(p)$. Similarly, the energy function at a minimizer, x, can be expressed as

$$f(x) = g(x,x) = \sum_{i<j}^{n} w_{ij}(x)d_{ij}^2 + \text{tr } X^\top V(W_x)X - 2 \text{ tr } X^\top B(X, W_x)X \tag{23}$$

When $x = p$, $A_{ij} = 1$ and $\beta = \alpha$, $w_{ij}(p) = w_{ij}(x) = 1$, then we have the SMACOF equation 15. However, for object clouds, we recompute both the weight and the Laplacian matrices at every iteration since the value of w_{ij} depends on the amount of attracting and colliding objects at a given iteration. At each iteration, w_{ij} changes because the objects that are being attracted or separated changes. Therefore our weighting is not constant as in the original SMACOF algorithm and as such the Cauchy-Schwartz inequality in equation 19 does not always hold true.

Fluctuation in the value of, w_{ij}, however does not affect the convergence of the energy function. We still get a set of non-increasing stress values that obey the sandwich inequality. Under normal MDS, the equation 19 is necessary for a set of decreasing stress values because the first two terms in $f(x)$ and $g(x, y)$ are equal and constant when w_{ij} is constant. Therefore the difference between the first two terms and the third term in $f(x)$ is less than difference between the third term in $g(x,p)$.

Under the MDS formulation for object clouds, a similar dynamic comes into play when w_{ij} fluctuates. If tr $X^\top B(X,W_x)X <$ tr $X^\top B(X,W_p)P$ then the first two terms in equation 23 are also less than those in equation 22 and vice versa. The value of these terms is such that the difference between the first two terms and the third term in equation 23 is less than difference first two terms and the third term in equation 22. Hence despite fluctuations in the value of w_{ij}, the sandwich inequality still holds for energy-based object clouds and we get a set of non-increasing energy values.

EXPERIMENTAL EVALUATION

Datasets

To evaluate the energy formulation for object clouds, we used the Princeton Shape Benchmark (PSB) and the ModelNet40 datasets [34][46].

The PSB dataset is a collection of 1,814 3D CAD models. The PSB dataset is unique in that it has several hierarchies of classification. They range from general classes like "musical instruments" and "furniture" to more specific classes like "acoustic guitar" and "desk with hutch" respectively. These hierarchies reflect both the primary and secondary form of each model. In total there are 161 general and specific classes. The lowest classification level contains at least four 3D models and the largest class (general or specific) contains 100 3D models.

The ModelNet40 is a larger collection of about 151,128 3D CAD models within the dataset, each of which belong to roughly 660 object categories [46]. Although there are many object categories containing everyday objects, the ModelNet40 dataset specifies 40 classes.

Methods Under Comparison

We evaluate the performance of each of the three energy optimization techniques discuss above. Additionally, because our energy-based model is related to the context preserving algorithm, we also evaluate its performance for generating semantic object clouds. Finally, we include the breadth-first search algorithm for pseudo-random object clouds as a baseline for our comparisons.

Metrics

In order to compare the various object cloud algorithms, we utilize 4 different metrics: trustworthiness, realized adjacency, compactness and energy.

Trustworthiness

When performing dimensionality reduction, each point in the original manifold has a neighbourhood that contains a set of close points that should ideally be retained in a lower dimensional manifold. Trustworthiness measures how well such neighbouring points from the original dimension are preserved in the lower dimension. It is defined as follows

$$T(k) = 1 - \frac{2}{nk(2n-3k-1)} \sum_{i=1}^{n} \sum_{j \in \mathcal{N}_i^k} \max(0, r_{ij} - k) \tag{24}$$

where $\mathcal{N}_i^k$ are the K-Nearest neighbours in the lower dimension and r_{ij} is the rank of each neighbour in the original input space. Each neighbour that is unexpected in the lower dimensional space is penalized by its rank in the original dimension and the fractional term helps normalize the output. Trustworthiness has a value from 0 to 1 where 0 indicates that all the points have unexpected neighbours and 1 indicates that the neighbourhood for every point in the higher dimensional space is well preserved.

Realized Adjacency

The realized adjacencies, defined in equation 25 are the set of objects that are adjacent to each object [2]. An object is adjacent another object if its boundary from that object is within 0.01-0.05% of the size of the smaller object. This metric measures the level of similarity across adjacent objects within the cloud. Its value is between [0, 1] with a higher value indicating that there many adjacent objects within the cloud are similar and vice versa.

$$A = \frac{\sum_{i,j \in E_r} \text{sim}(p_i, p_j)}{\sum_{k,l \in E} \text{sim}(p_k, p_l)} \tag{25}$$

Compactness

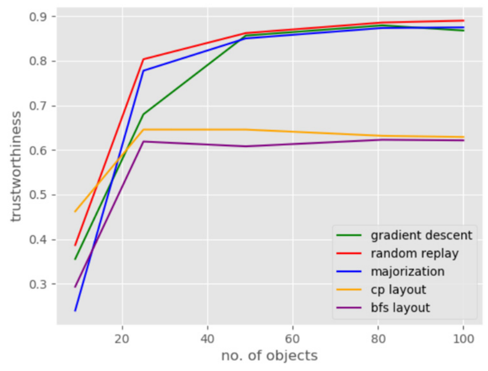
Compactness measures how tightly packed each of the words are within the given 2D layout of the cloud. It is calculated by dividing the total area by the used area. The total area refers to the tightest rectangular area bounding all the objects within the layout. It is measured by multiplying the difference between the X-axis of the leftmost and rightmost object or word with the difference between the Y-axis of the uppermost and lowermost object. The used area however is the sum of the area of each individual objects within the cloud. The metric can be expressed as follows

$$C = \min(1, \frac{\text{Used Area}}{\text{Total Area}}) \tag{26}$$

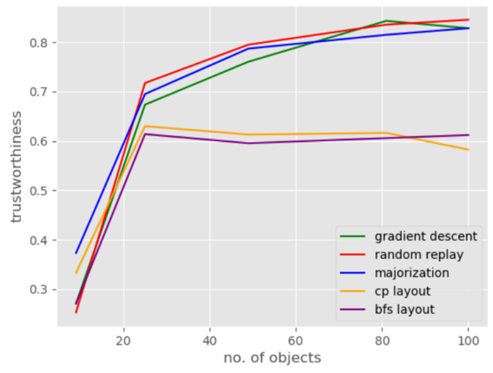
When the rectangular area bounding all the objects is larger than the sum of area of all the objects, $C < 1$ indicating that there is some space between the objects. As this space increases, $C \to 0$. Conversely, when all the objects are tightly packed, the rectangular area is equal to the sum of the area of all the objects and $C = 1$. However, if many of the objects begin to overlap each other, $C > 1$. Therefore, a good value for compactness is $0.5 \leq C \leq 1$.

Data Preprocessing

Each of the 3D models from the PSB and ModelNet40 datasets was passed into a Blender python script in order to extract varying views of the model. The various views are obtained from cameras that are positioned at an angle and ground elevation of 30° around the model. The python script generates 12 views which are then passed into an MVCNN that was pre-trained using the Resnet-18 neural network. Rather than use the MVCNN network as it is, we fine-tuned it to each dataset by training it over a dataset for 5 epochs. Each epoch was over 1000 iterations and at the end of training, the MVCNN had an accuracy of over 85% on each of dataset. After training, the dataset was passed into the network and the vector output from the penultimate layer was used as high-dimensional vector embedding for each of the algorithms to be evaluated.



(a) ModelNet 40



(a) Princeton shape benchmark

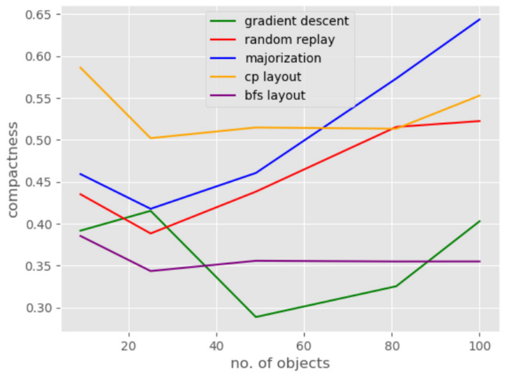
Figure 4. Trustworthiness of various algorithms on different shape datasets.

Results

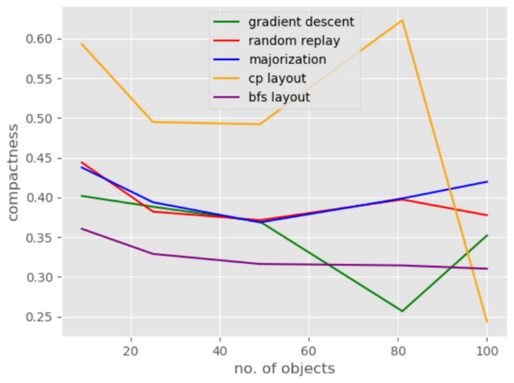
Recalling that our target application is to browse a similar number of objects that typically found on commercial webpages (a few dozen), we compare clouds of up to 100 objects. Likewise, we focussed on layout quality rather than computation speed, since we are not browsing a very large number of objects. We evaluated how well each of the optimization methods performed on the shape datasets for 9, 25, 49, 81, and 100 objects.

Firstly, to give the reader a sense of what the clouds look, consider the results shown in figures 8-12. These are generated from a single run with 49 models using the various layout algorithms (gradient decent, random replay, majorization, context preserving algorithm and breadth-first search). But individual runs are not very informative, so we also need to look at the results shown in the aggregated graphs.

In figure 4 we see that on both datasets, all three optimization layouts outperform the context preserving and breadth-first layouts. While the context preserving and breadth-first layouts manage to preserve some of the high-dimensional semantic structure of the objects, they do not do so to the degree of the optimization layouts. As discussed in the previous sections, this is due to the fact that the optimization layouts utilize a high-dimensional KNN graph in their operation as opposed to the lower dimensional graph. Therefore, they lose less information about the semantic structure of the object representations. This is illustrated in figures 8, 9, and 10 where objects like sofas and chairs are grouped separately but placed in such a way that they morph into each other. Whereas in the figures 11 and 12, there is some semantic order but those objects are not clearly grouped.

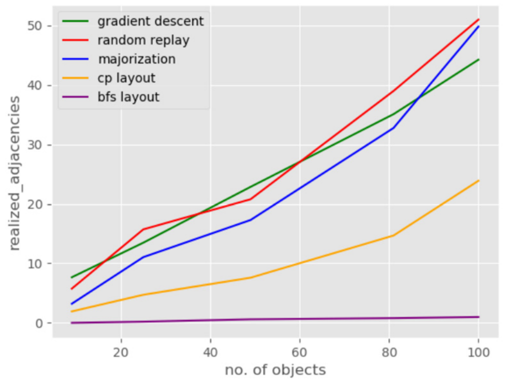


(a) ModelNet 40

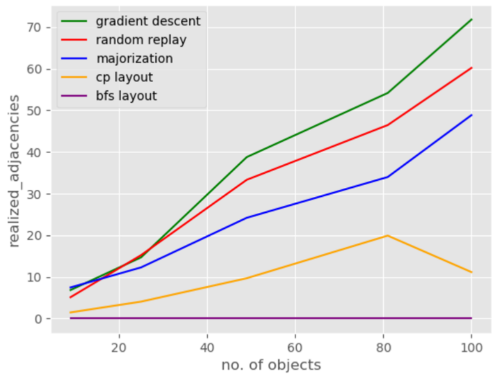


(b) Princeton shape benchmark

Figure 5. Compactness of the various algorithms on different shape datasets.



(a) ModelNet 40



(b) Princeton shape benchmark

Figure 6. Realized adjacency of various algorithms on the shape datasets.

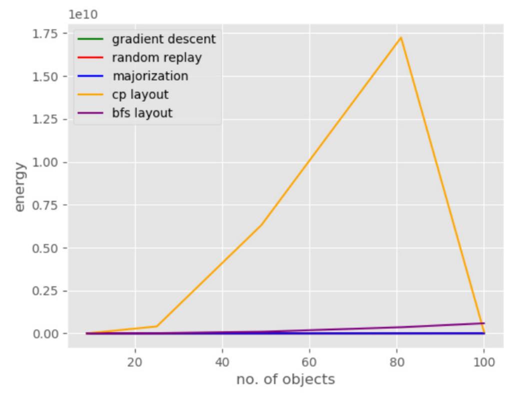
When we observe the compactness of the layouts from figure 5, we can see that the majorization, random replay and context preserving algorithms perform well on both datasets. The relatively poor performance of gradient descent is due to the way compactness is calculated. The figure 8 has a wider bounding box for its object cloud than those of figures 9 and 10. Hence the total area is much larger. The gradient descent layout also has relatively more overlaps which leads to a smaller used area than that of the other two optimization layouts. The smaller used area and larger total area therefore lead to much less compactness for the gradient descent layout. The breadth-first layout on the other hand makes a more uniform use of the layout as illustrated in figure 11. However, some of the objects are too small for the grid in which they have been placed in, leading to excess space among objects of various grids and thus a relatively low compactness for the layout.

From figure 6, we once again observe that the optimization algorithms out-performed both the context preserving and breadth-first layouts on the realized adjacency. This is likely due to the fact that the optimization layouts minimize the squared pairwise distance between neighbouring objects while maintaining a high degree of semantic similarity between said objects. By minimizing the distance, similar pairs of objects touch each other which in turn increases the realized adjacency of the layouts. The context preserving layout does the same thing but additionally it tries to maintain the planarity of its underlying Delaunay graph.

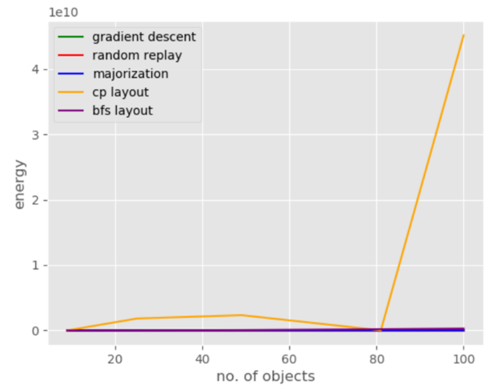
As discussed earlier, this graph has a lower degree of semantic similarity and so when neighbouring objects do touch each other, they might not be very similar which in turn reduces the realized adjacency of the layout. Furthermore, as can be seen in figure 12, the highly compact nature of the context preserving layout means that neighbours that are colliding do not get included in the realized adjacency sum thus leading to a lower value for the layout.

While a highly compact layout in which there are significant collisions may reduce the realized adjacency, a less compact layout such as that of the breadth-first layout also reduces the realized adjacency because fewer object are touching each other. Since many of the objects in the breadth-first layout are smaller than their grid, the space between neighbouring objects was too large for the objects to be considered as touching each other. Hence the very low realized adjacency for the layout.

In figure 7 we observe that except for the context-preserving layout, the layouts for the other algorithms minimize the energy of an object cloud. This may be due to the large amount of colliding objects within the context-preserving layout. As discussed in the previous sections, a larger repulsive weighting is needed to repel colliding objects. Therefore, when computing the energy of a cloud, colliding objects have higher energy. Since our experiment utilized a repulsive weighting that is 8 times that of the attractive weighting, this may account for the high energy within the context-preserving cloud.



(a) ModelNet 40



(b) Princeton shape benchmark

Figure 7. Energy of the various algorithms on different shape datasets.

CONCLUSION

We proposed an energy function for object clouds. By optimizing the energy function, we showed that we can create a semantic object cloud. This is due to the fact that we minimized the squared distance between similar objects in the K-NN graph while maximizing the squared distance between dissimilar objects in the graph. To that end, we explored 3 optimization strategies: gradient descent, random replay and majorization.

We discussed each of these strategies and their advantages over each other. Random replay is like gradient descent, but it randomizes the order in which the objects are adjusted thereby allowing it to escape local minima that gradient descent may get stuck in. Majorization however is a very different strategy in that all the objects are adjusted at the same time and the adjustment is based on the minimization of a surrogate for the proposed energy function. This allows it to construct object clouds faster and it decreases the energy function monotonically. Using the decreasing monotonicity, we can stop the construction of an object cloud whenever the energy function ceases to decrease rather than specifying a set number of iterations within which to minimize it.

From the minimization strategies, we proposed a set of algorithms for constructing object clouds. We then compared the layout from these algorithms against other the layout of algorithms that include the breadth-first search and context preserving algorithms. We used metrics such as trustworthiness, compactness, and realized adjacency to facilitate our evaluation.

The optimization algorithms outperformed the other algorithms on trustworthiness due to the fact that less semantic information is lost when using a high dimensional graph as opposed to a lower dimensional graph like the Delaunay graph. In terms of compactness, both the context preserving layout and the optimization layouts were compact. The breadth-first search layout was not as compact because of its uniform use of layout space. Finally in terms of realized adjacency, the optimized layouts did outperform the other algorithms due to a combination of high semantic order and compactness. The context preserving algorithm did not perform as well because the semantic order of the layout was not as high and some objects were too close to be considered as touching each other. The breadth-first search in a similar vein did not perform as well because the space between the objects was too large for them to be considered as touching.

Finally, we note that there are several important limitations of this work. Given our target application of presenting clouds of objects normally found in online webpages we only considered a maximum of 100 objects that could be browsed at a reasonable size. In addition, we compared 5 methods for computing the layouts of 3D clouds, but this is not exhaustive, and others could be tried. We also note that all the methods produced some degree of overlap due to competing constraints which is reflected in the adjacency metric scores. Moreover, a second user study on semantic 3D clouds may offer additional insights into user preferences.

References

[1] S. Avidan and A. Shamir. “Seam carving for content-aware image resizing”. ACM Trans. Graph, page 10. SIGGRAPH, 2007.

[2] L. Barth, S. Kobourov, and S. Pupyrev. “Experimental comparison of semantic word clouds”. Experimental Algorithms, 247–258. Springer, 2014.

[3] G. Begelman, P. Keller, F. Smadja, et al. “Automated tag clustering: Improving search and exploration in the tag space”. WWW2006, Edinburgh, Scotland, pages 15–33, 2006.

[4] M. Belkin and P. Niyogi. “Laplacian eigenmaps for dimensionality reduction and data representation”. Neural computation, 15(6):1373–1396, 2003.

[5] M. Billinghurst, A. Clark, and G. Lee. “A survey of augmented reality”. Foundations and Trends in Human-Computer Interaction, pages 73–272, 2015.

[6] C. Bishop. “Pattern recognition and machine learning”. Springer, 2006.

[7] L. Bottou. “Curiously fast convergence of some stochastic gradient descent algorithms”. Learning and Data Science, Paris, 2009.

[8] S. Chaudhuri. “Shape Descriptors - iii”, Indian Institute of Technology Bombay, May 2016.

[9] D. Chen, X. Tian, Y. Shen, and M. Ouhyoung. “On visual similarity based 3D model retrieval”. Computer graphics forum, volume 22, 223–232, 2003.

[10] L. Chen and A. Buja. “Local multidimensional scaling for nonlinear dimension reduction, graph drawing, and proximity analysis”. Journal of the American Statistical Association, 104(485):209–219, 2009.

[11] W. Cui, Y. Wu, S. Liu, F. Wei, M. Zhou, and H. Qu. “Context preserving dynamic word cloud visualization”. PacificVis, 121–128. IEEE, 2010

[12] R. Davidson and D. Harel. “Drawing graphs nicely using simulated annealing”. ACM Transactions on Graphics (TOG), 15(4):301–331, 1996.

[13] J. De Leeuw. “Convergence of the majorization method for multidimensional scaling”. J. of classification, 5(2):163–180, 1988.

[14] J. De Leeuw and Patrick Mair. “Multidimensional scaling using majorization: Smacof”. Journal of Statistical Software, 31(3), 2009.

[15] T. Fruchterman and E. Reingold. “Graph drawing by force directed placement”. Software: Practice and experience, 21(11):1129–1164, 1991.

[16] E. Gansner, Y. Koren, and S. North. “Graph drawing by stress majorization”. Graph Drawing, 239–250, 2004.

[17] H. Geoffrey. “Non-linear dimensionality reduction”. University of Toronto Dept of Computer Science, May 2013.

[18] A. Gracia, S. González, V. Robles, and E. Menasalvas. “A methodology to compare dimensionality reduction algorithms in terms of loss of quality”. Information Sciences, 270:1–27, 2014.

[19] J. Graef and I. Spence. “Using distance information in the design of large multidimensional scaling experiments”. Psychological Bulletin, 86(1):60, 1979.

[20] F. Heimerl, S. Lohmann, S. Lange, and T. Ertl. “Word cloud explorer: Text analytics based on word clouds”. System Sciences, pages 1833–1842. IEEE, 2014.

[21] X. Hong and S. Brooks. 2020. “3D Objects Clouds: Viewing Virtual Objects in Interactive Clouds”. IEEE Transactions on Visualization and Computer Graphics 26, 3, 1442–1453.

[22] Y. Hu. “Efficient, high-quality force-directed graph drawing”. Mathematica Journal, 10(1):37–71, 2005.

[23] T. Kamada, S. Kawai, et al. “An algorithm for drawing general undirected graphs”. Information processing letters, 31(1):7–15, 1989.

[24] S. Kobourov. “Spring embedders and force directed graph drawing algorithms”. arXiv preprint arXiv:1201.3011, 2012.

[25] A. Lambie. “Directing Attention in an Augmented Reality Environment: An Attentional Tunneling Evaluation”. PhD thesis, Rochester Institute of Technology, 2015.

[26] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. “Gradient-based learning applied to document recognition”. Proceedings of the IEEE, 86(11):2278–2324, 1998.

[27] M. Leginus, P. Dolog, R. Lage, and F. Durao. “Methodologies for improved tag cloud generation with clustering”. Web Engineering, 61–75. Springer, 2012.

[28] W. Lu, H. Duh, S. Feiner, and Q. Zhao. “Attributes of subtle cues for facilitating visual search in augmented reality”. IEEE transactions on visualization and computer graphics, 20(3):404–412, 2013.

[29] T. Mikolov, K. Chen, G. Corrado, and J. Dean. “Efficient estimation of word representations in vector space”. arXiv:1301.3781, 2013.

[30] K. Murphy. “A Probabilistic Perspective”. MIT Press, 2012.

[31] F. Paulovich, F. Toledo, G. Telles, Minghim, and L. Nonato. “Semantic wordification of document collections”. Computer Graphics Forum, volume 31, 1145–1153, 2012.

[32] K. Petersen, M. Pedersen, et al. “The matrix cookbook, vol. 7.” Technical University of Denmark, 15, 2008.

[33] E. Schubert, A. Spitz, M. Weiler, J. Geiß, and M. Gertz. “Semantic word clouds with background corpus normalization and tdistributed stochastic neighbor embedding”. arXiv preprint arXiv:1708.03569, 2017.

[34] P. Shilane, P. Min, M. Kazhdan, and T. Funkhouser. “The princeton shape benchmark”. Shape Modeling Applications, 167–178. 2004.

[35] J. Sinclair and M. Cardew-Hall. “The folksonomy tag cloud: when is it useful?”, Journal of Information Science, 34(1):15–29, 2008.

[36] G. Smith. “Tagging: people-powered metadata for the social web”, safari. New Riders, 2007.

[37] J. and N. Iliinsky. “Beautiful visualization: Looking at data through the eyes of experts”, ch. 3, 37–58. O’Reilly Media, 2010.

[38] H. Su, S. Maji, E. Kalogerakis, and E. Learned-Miller. “Multi-view convolutional neural networks for 3D shape recognition”. ICCV, 2015.

[39] J. Tenenbaum, V. De Silva, and J. Langford. “A global geometric framework for nonlinear dimensionality reduction”. Science, 290(5500):2319–2323, 2000.

[40] D. Tunkelang, D. Sleator, P. Heckbert, and B. Maggs. “A numerical optimization approach to general graph drawing”. Technical report, Carnegie-Mellon, 1999.

[41] J. Venna and S. Kaski. “Neighborhood preservation in nonlinear projection methods: An experimental study”. Artificial Neural Networks, pages 485–491. Springer, 2001.

[42] K. Vyshenska. “How to build a parameterized archimedean spiral geometry”, Jul 2016.

[43] C. Wickens, S. Gordon, Y. Liu, et al. “An introduction to human factors engineering”. Longman New York, 1998.

[44] J. M Wolfe. “Visual search”. The handbook of attention, 27–56, 2015.

[45] Y. Wu, T. Provan, F. Wei, S. Liu, and K. Ma. “Semantic preserving word clouds by seam carving”. Computer Graphics Forum, vol. 30, 741–750, 2011.

[46] Z. Wu, S. Song, A. Khosla, F. Yu, L. Zhang, X. Tang, and J. Xiao. “3D shapenets: A deep representation for volumetric shapes”. CVPR, pages 1912–1920, 2015.

[47] J. Xu, Y. Tao, and H. Lin. “Semantic word cloud generation based on word embeddings”. PacificVis, 239–243. IEEE, 2016.

[48] M. Yeh and C. Wickens. “Visual search and target cueing: A comparison of head-mounted versus hand-held displays on the allocation of visual attention”. Technical report, Army Research Lab Aberdeen, 1998.

[49] M. Kaufmann and D. Wagner, eds. "Drawing Graphs: Methods and Models", Lecture Notes in Computer Science, vol. 2025, Springer-Verlag, 2001.

[50] M. Espadoto, R. M. Martins, A. Kerren, N. S. T. Hirata and A. C. Telea, "Toward a Quantitative Survey of Dimension Reduction Techniques," in *IEEE Transactions on Visualization and Computer Graphics*, vol. 27, no. 3, pp. 2153-2173, 2021.

Author Biography

Bola Okesanjo received his BSc from the University of Toronto (2016), and his Master of Computer Science from Dalhousie University (2020). His interests include visualization and machine learning.

Stephen Brooks is a professor of computer science at Dalhousie University. He received his MSc from the University of British Columbia (2000), and his PhD in computer science from Cambridge (2005). His research interests include visualization, computer graphics, and interaction.

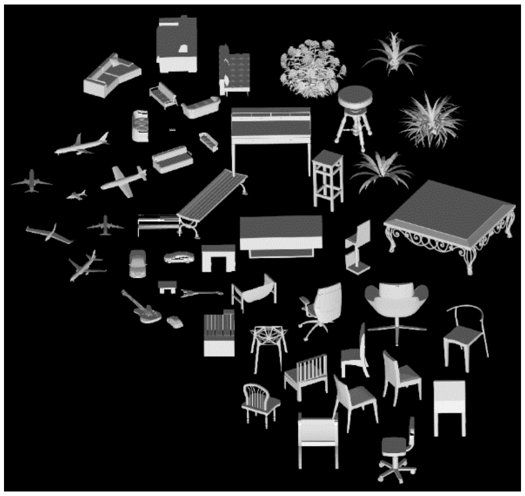


Figure 10: Majorization Layout on ModelNet40 dataset. (# objects = 49)

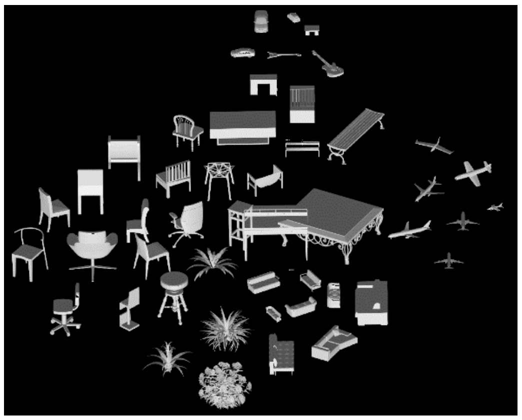


Figure 8: Gradient Descent Layout on ModelNet40 dataset. (# objects = 49)

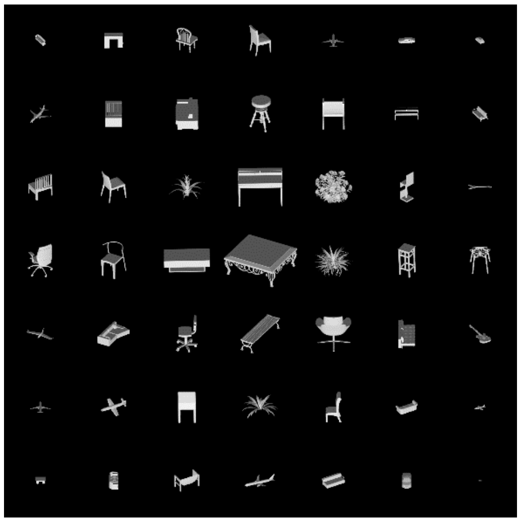


Figure 11: Breadth First Search on ModelNet40 dataset. (# objects = 49)

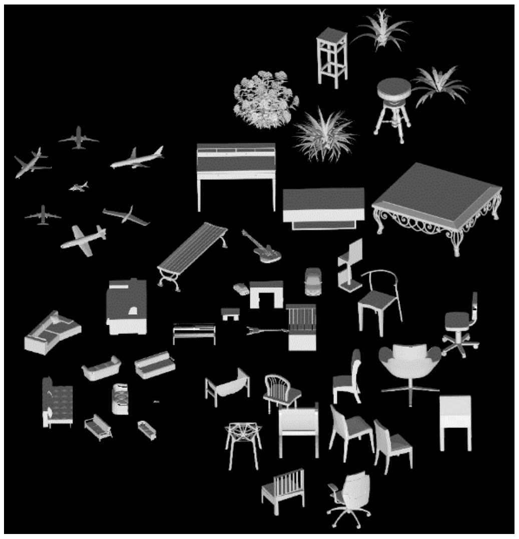


Figure 9: Random Replay Layout on ModelNet40 dataset. (# objects = 49)

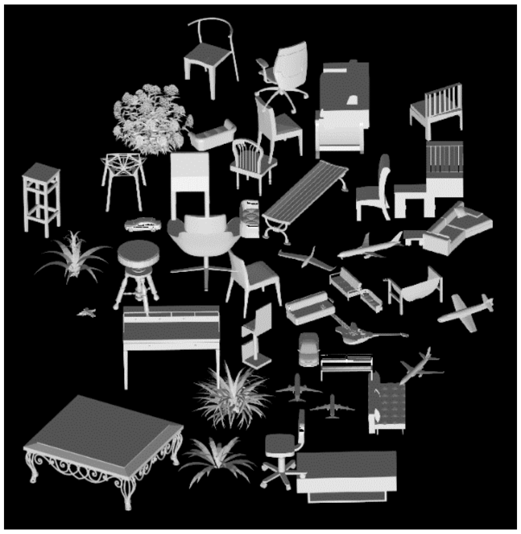


Figure 12: Context Preserving Layout on ModelNet40 dataset. (# objects = 49)