Deformation as a Sub-volume Selection Method in Virtual Reality

Thomas van Coller
Department of Computer Science
University of Cape Town

ABSTRACT

Virtual reality (VR) has the potential to expand the uses of computing due to it immersing one in an environment with the same dimensions as the natural world. Most software developed is for desktop environments, but not all tasks necessarily lend themselves to a two-dimensional screen. Consider data analysis as an example, not all data is two-dimensional and hence visualising this data on a screen can be challenging. Moreover, selecting out subsets of the data can be difficult if the dimensions cannot be represented in an intuitive manner. Volumetric data, or data with three dimensions, has traditionally been visualised and analysed on a desktop, but virtual reality may offer a more intuitive space for this kind of data. Volumetric data naturally lends itself to a three-dimensional environment. Hence, visualising and interacting with the data in a two-dimensional desktop environment may not be ideal. More specifically, selecting out sub-regions of the data can prove to be tedious. Making use of VR allows one to be immersed in the data and view it within an environment with three dimensions. VR also allows for bi-manual interaction of the space with six degrees of freedom. This paper presents a technique for selecting out sub-regions of volumetric data in virtual reality. Selecting out a sub-region of volumetric data in VR can be approached in many different ways; in this paper a deformable or "mould-able" sphere is explored. The user is able to mould this sphere, as one might do with clay, around the region of interest in the data and then select out that region of the data. Moreover, a desktop selection method was developed as a baseline for comparison. These selection methods were integrated into the iDaVIE-v software developed by Marchetti et al. [7] to allow experiments to be conducted. Thirty users were recruited for the experiments in which they were trained with each method before being required to select out a sub-region of volumetric data using the methods. The goal of these experiments, and this paper as a whole, is to determine whether VR allows for more accurate selections as well as a more user-friendly environment for selection. These experiments indicated that there were no improvements in accuracy, usability, and workload with VR compared to a desktop environment. However, it was observed that the desktop method was more efficient than the deformation method.

KEYWORDS

Volumetric selection, object selection, volumetric data, virtual reality, data cube, voxel

1 INTRODUCTION

Volumetric data is produced by many different fields of study with two major fields being astronomy and medicine. For example, astronomical data is generally collected in two spatial dimensions with an additional spectral dimension [11]. IDaVIE-v [7] was developed by the astronomy department at the University of Cape Town to allow astronomers to view and analyze volumetric data in VR and is a basis for the implementation of the selection methods discussed in this paper. Astronomical data often consists of one or more regions of interest that the astronomers need to isolate. These regions of interest do not usually have a regular shape like a cuboid or sphere. Hence, making use of a sphere that can be sculpted into the desired shape can make the selection of these regions more accurate and intuitive. Figure 1 gives a simple example of how a sphere may be deformed.

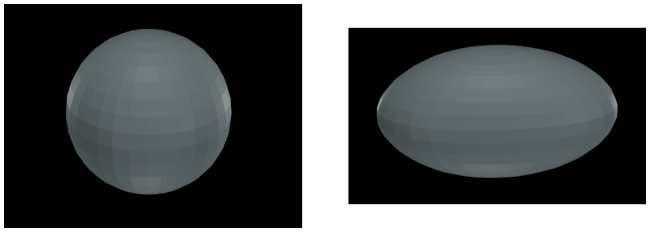


Figure 1: An example of how a sphere may be deformed

However, interacting with data in a 3D setting has been shown to be challenging; the user enters an environment that mimics reality, but cannot rely on haptic feedback or other senses that one is used to [20]. Thus, creating a mechanism that feels intuitive to the user was a priority. The basis of this deformation model is derived from the paper written by Sederberg and Parry [23]. This paper presents a process for deforming solid geometric models in a free form manner; the details of which will be explored in a sections 2, 3, and 4. However, this model did not account for direct manipulation of the geometric object by the user. Gain [12] extends the free form deformation model to include direct manipulation. This was followed closely to replicate the model for iDaVIE-v. Moreover, to allow for the sub-region to be selected an inside-outside test for the sphere's mesh was implemented to determine which data points fell inside and outside the data. In addition to the deformation method, a desktop selection method was developed as a basis for comparison. Volumetric data is represented as a data cube in iDaVIE-v, where the cube is made up of smaller cuboid units called voxels. A voxel represents a single data point within the volumetric dataset. The desktop method allows users to look through the data in slices and isolate regions of interest in each slice, where each slice is a single plane of voxels. The user can move through slices one by one in any of the three dimensions of the data cube. To compare the two selection environments, user experiments were conducted to gather data relating to the accuracy, usability, and workload of the methods. These experiments involved recruiting users to use the methods to perform sub-selections on volumetric data sets. Users were trained with each method to then use that method to select out a desired sub-region of a particular data cube. Ultimately, the experiment aimed to test whether VR provides a better environment for sub-volume selection of volumetric data than a desktop

environment. This paper will cover the implementation details of both selection methods, followed by the experimental design of the user tests. Following this, the results of the experiments will be discussed and analyzed, taking into account any and all issues from the experimental procedure. Finally, the paper will conclude whether VR allows for more accurate selections as well as a more user-friendly environment for sub-volume selections of volumetric data compared to a desktop environment.

2 BACKGROUND

This section outlines all the relevant information needed to implement the deformable sphere. It outlines free-form deformation, the use of hyperpatches to build up the model, and finally how the model can be extended to allow for direct manipulation.

2.1 Free-Form Deformation

Solid geometric models in computing comprise a set of vertices, each of which have a position in space relative to the model's local coordinate system. These vertices often contain more information than just position such as normal vectors, texture coordinates etc. These vertices are then connected by defining triangles of sets of three vertices, whereby these vertices connect to form a triangle. The basis of deforming a geometric model lies in passing the vertices of the model through a function that produces new positions for the vertices relative to the constraints of the deformation model. Gain [12] defines spatial deformation as the following: Spatial Deformation can be formulated as a mapping $f: \mathbb{R}^3 \longmapsto \mathbb{R}^m \longmapsto \mathbb{R}^3$, from world space $X = (x, y, z)$, through a local parameter space, U of dimension m, to deformed world space $\tilde{X} = (\tilde{x}, \tilde{y}, \tilde{z})$.
Free-Form Deformation (FFD) warps the space around an object and transforms this object in an indirect manner [2, 8, 12]. Sederberg and Parry [8] make the analogy that deforming an object can be seen as embedding the object in a flexible jelly with the same consistency as the object. Any change or moulding of the jelly will then distort the embedded object in a similar manner. This idea forms the basis for the hyperpatch deformation model used in this paper.

2.2 Hyperpatch Based Deformation

A hyperpatch, or Bézier patch [2] comprises a set of control points. These control points are used to generate a surface whereby the positions of the control points "pull" the surface toward them. Figure 2 illustrates this idea: This idea of the hyperpatch can be extended to three dimensions; instead of a 4x4 set of control points that produce a surface it could be thought of as having a 4x4x4 set of control points forming a volume. This is necessary for deformation as we need to embed a three dimensional geometric object into an enclosed space, which itself would need to have three dimensions. The set of control points will be referred to as the control lattice. The control lattice does not need to be 4x4x4, it can be larger than that in one, more, or all dimensions. However, it cannot be smaller than that, as a set of 4x4x4 control points make up a single cell for the deformation model. If we imagine a lattice of 4x4x4 control points equally spaced between 0 and 1 on each axis of a Cartesian coordinate system, we can think of the cell as the inner-most cube, with corners represented by the four inner-most control points. As

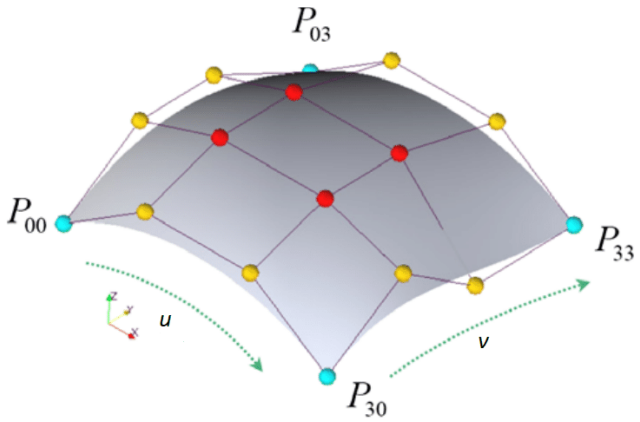


Figure 2: Hyperpatch with a set of 4x4 control points. (Source: [19])

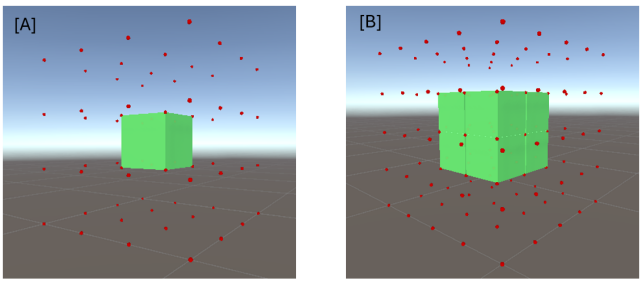


Figure 3: [A]: 4x4x4 Lattice with one cell. [B]: 5x5x5 lattice with 8 cells. Cells are shown in green.

mentioned previously, we can have as many control points as we like in each dimension as long as there are at least 4 per dimension. This allows us to have multiple cells for deformation, where each cell comprises the set of 64 control points surrounding it. Figure 3 shows a lattice with a single cell [A] and a 5x5x5 lattice with eight cells [B]. To allow an object to be distorted relative to the patch, it needs to be embedded in this cell whereby all the vertices need to be reassigned new positions relative to the cell. The vertices with coordinates (x, y, z) get assigned parametric coordinates (u, v, w) under this embedding. We consider this an embedding function whereby $E(x, y, z) \longmapsto (u, v, w)$. Once embedded, points inside the patch are able to be distorted relative to the patch by passing their parametric coordinates through the function defined in equation (1). This function calculates the new position with a weighted sum of the control points surrounding the cell the vertex is in. The control points are weighted with polynomial basis functions. The formula for this function is as follows:

$$H(u, v, w) = \sum_{i=0}^{3} \sum_{j=0}^{3} \sum_{k=0}^{3} N_i^3(u) \cdot N_j^3(v) \cdot N_k^3(w) \cdot P_{i,j,k} \qquad (1)$$

Whereby N_i^3 is the i'th polynomial basis function of degree 3, and $P_{i,j,k}$ is the control point at index (i, j, k) from the set of 4x4x4 control points surrounding the cell. While there are a variety of polynomial basis functions, the model used in this paper makes use

of uniform B-Spline basis functions defined as follows:

$$N_0^3(u) = \frac{(1-u)^3}{6}$$
$$N_1^3(u) = \frac{3u^3 - 6u^2 + 4}{6}$$
$$N_2^3(u) = \frac{-3u^3 + 3u^2 + 3u + 1}{6}$$
$$N_3^3(u) = \frac{u^3}{6}$$

To allow the embedded object to be deformed with respect to the lattice, all its vertices are passed through equation (1), giving them all new positions in space, given that the control points have been moved relative to the base lattice. However, this does not account for how the lattice can be altered based on user input, i.e. user manipulation of the lattice. This is vital to the system as users need to be able to interactively deform the sphere.

2.3 Direct Manipulation

Through the work by Hsu et al [15], Gain [12] extends this deformation model to allow direct manipulation of the embedded object. This works by allowing the user to specify a set of constraints that will be used to shift the control points in the lattice. A constraint is made up of C_i, the position at which the user would like to deform on the surface of the embedded object, as well as ΔC_i, a vector representing the direction of motion of the deformation. Figure 4 displays this idea with a 2D equivalent. To produce the new lattice control point positions, the following equation is produced for each constraint:

$$C_i + \Delta C_i = \sum_{i=0}^{3}\sum_{j=0}^{3}\sum_{k=0}^{3} N_{i,j,k}^{3,3,3}(E(C_i)) \cdot (P_i + \Delta P_i) \quad (2)$$

Where $N_{i,j,k}^{3,3,3}(E(C_i))$ represents the same basis functions in equation (1) applied to the embedded constraint. All terms are known except ΔP_i, the direction of movement of each control point. This allows a system of linear equations to be produced in which we can solve to find each ΔP_i. Moreover, equation (2) can be simplified into the following: $\Delta C = B\Delta P$. Here ΔC is a column vector containing all the constraints, B is a matrix where each row contains the basis functions used in each iteration of equation (2) for each constraint, and ΔP is what we are solving for. However, since each row of the matrix B will have 64 entries the system of equations will always be under-determined. This is because the user is unable to apply more than two constraints at any given time as the deformation will be conducted with controllers used for VR, this limits the user to using one or both hands, and hence can only deform in at most two directions at one time. In order to get a solution to the system of equations, a pseudo-inverse solution needs to be generated [12]. To calculate the pseudo-inverse solution, the method of normal equations is adopted [13]. This method allows us to solve for ΔP explicitly with the following solution:

$$\Delta P = B^T(BB^T)^{-1}\Delta C \quad (3)$$

Gain [12] presents the observation that the matrix $A = BB^T$ meets all the conditions for Choleski Decomposition [21]. Namely, that A is symmetric ($A = A^T$), and non-negative definite ($x^T Ax \geq 0$). This allows us to invert the matrix A and solve for ΔP. This produces the vectors that each control point needs to shift by. Adding these vectors to the respective control point produces the new position for that point. Using these new positions, the embedded geometric object can be warped by passing its vertices through equation (1).

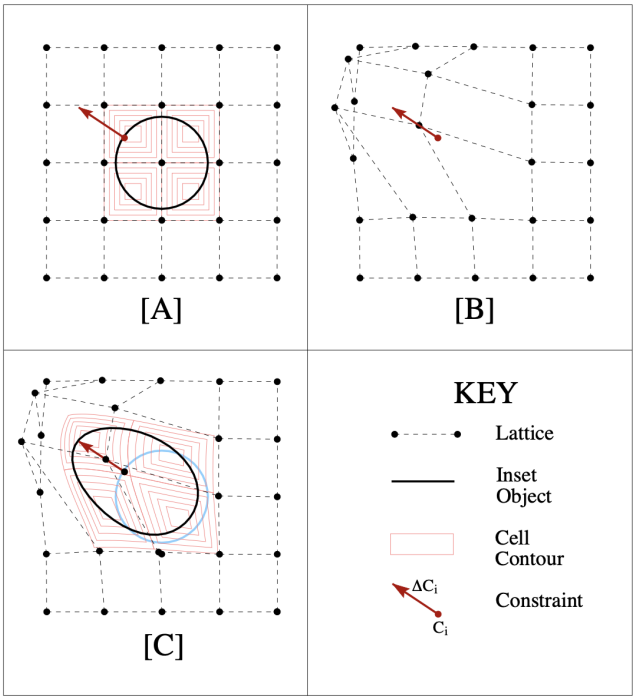


Figure 4: Stages of Directly Manipulated FFD. [A]: Constraint defined by user. [B]: Lattice moved based on constraint. [C]: Embedded circle warped based on new lattice points. (Source: [12])

3 RELATED WORK

3.1 Methods of Deformation

The Free-Form deformation model introduced by Sederberg and Parry [8] presents a method for indirectly deforming a 3D model based on changes to a volume enclosing that object. This method allows for intuitive deformation of simple objects by manipulating a set of control points. However, this method can be limited when deformations of complex 3D models are required. This is because the 3D model may not fit into the lattice of control points in a sensible manner. Think of animating/deforming the model of a 3D animal; a cuboidal lattice does not fit the shape of this model well, leading to unusual deformations. Hence, alternative methods of deformation have since been developed. Cage-based deformation [6] provides a method for deforming by surrounding a detailed 3D model with a simpler, low-resolution mesh, often referred to as a "cage." The cage acts as a control structure that influences the shape and position of the model contained within it. By deforming or manipulating the vertices of the cage, the underlying model deforms accordingly. This cage often closely replicates the object it

encompasses, and hence lends a better environment for deforming complex 3D models.
However, this method still provides a broad control over the mesh of the 3D model it is trying to deform as the cage is not tightly bound to the object. A popular alternative to the cage-based model is a skeletal-based method of deformation [9]. Instead of encompassing the 3D model within a cage, this method constructs a set of 'joints' and 'bones', similar to a skeleton. The vertices of the model are then attached to this skeleton and any transformations to the skeleton lead to subsequent transformations of the model's vertices.
Although both the cage and skeleton based methods provide more flexibility than free-form deformation, the complexity of their implementation made them infeasible for this project's timeline. Moreover, only deformation of a sphere is needed in this project, for which free-form deformation is well suited.

3.2 Methods of Selection

Although this paper makes use of a deformable sphere to make selections in virtual reality, there are endless possibilities for going about selection in a three-dimensional environment. This sub-section outlines some alternative methods of selection.

3.2.1 Cuboid Selection. A simple method for selecting out regions of 3D space is with a simple cube or rectangular prism. Ulinski et al. [5] proposed a methodology for doing exactly that, using both hands to generate a cuboid for selection. The idea behind this method is that each hand is assigned to a different part of the cube where one hand controls the position of the cube and the other controls the size. This method is simple and intuitive to use, but may cause difficulty in making precise selections. Montano et al. [20] proposes a hybrid cuboid selection method whereby the user interacts with both VR and real tangible hardware to complete a selection. This method made use of special controllers, one having a pen and the other a tablet. The idea is that a cuboid is used for selection in VR, this is then mapped to the tablet where more refined selections can be made.

3.2.2 Lasso Selection. Although using a cuboid to select is simple and intuitive, many objects may not be shaped in a way such that it is optimal for selection. Lucas et al. [17] present a method for selection in VR using a lasso called Cylindrical Selection. The user has a virtual tablet attached to their hand in which they can see a desired 2D view of the 3D space. They can then draw a loop onto this to project a cone-like mesh into the 3D space in the shape of the loop they drew. Anything within this cone will then be selected. However, this method does not take into account the spatial structure of the objects within the cone. Yu and Efstathiou extended Cylindrical Selection using density calculations within the lasso to select out objects. They developed two different methods: Cloud Lasso and Teddy Selection. The extension these methods provide allowed for smooth, accurate selections of objects in the 3D space.

3.2.3 Automatic Selection. If there are patterns in the system for which selections are being made, selections can be automated. IDaVIE-v currently makes use of SoFiA developed by P. Serra et al. [10]. SoFiA is a general source finding software for 3D spectral line data. It was developed with the purpose of detecting sources in astrophysical data, more specifically H1 emission data. SoFiA provides many source-finding techniques, combining many algorithms to optimise the selection of sources. SoFiA has been found to make a good basis for selection which can then be refined, but is often not accurate enough to be considered a reliable selection method on its own, and is often used in conjunction with manual selection methods.

4 METHOD DESIGN AND IMPLEMENTATION

The deformation model described above is used to develop an interactive deformable sphere in iDaVIE-v. IDaVIE-v was developed using the Unity Engine, and as such, the selections methods were also developed in Unity with C# as the programming language.

4.1 System Flow & Requirements

Within iDaVIE-v, controllers are used to navigate and interact with various components of the software. The same controllers are used to interact with the deformable sphere. The flow of the deformation system starts when the user selects the deformation mode from the standard menu in the iDaVIE-v system. This places a slightly transparent white sphere in front of the user and swaps them to the deformation menu. This menu is attached to the controller of the primary hand of the user. This ensures the user has control of where the menu is placed in the system, supported by Schneiderman's 7th Golden Rule of interface design *Support Internal Locus of Control* [18]. The user can then choose between two modes: move mode and deform mode. The user is initially placed in move mode whereby they can move the sphere to a desired location. This is done by placing a controller inside the sphere, upon doing so the sphere changes to green to indicate it is ready to be interacted with. This design choice is supported by Schneiderman's 3rd Golden Rule *Offer Informative Feedback*. Once a hand is inside the sphere the user can press the trigger on the controller to grab it, move it, then release the trigger to stop moving the sphere. The user can then swap to deform mode either on the menu, or with a button click on the controller. Once in deform mode, the user can deform the sphere. Prior to deforming, the user can adjust their area of effect setting, indicated by a slider bar on the menu. The area of effect represents how much of the sphere's surface is deformed given a particular constraint. Figure 5 displays this effect. Once set, the user can deform by placing a hand on the sphere's surface, holding a button and moving their hand in the desired direction of deformation. Once the button is released the sphere will stop deforming. If the user is unhappy with their deformation, they can undo this action or reset the sphere to its base state, supported by Schneiderman's 6th Golden Rule *Permit Easy Reversal of Actions*. If the user is happy with their shape, they can press a button on the menu to select all voxels within the shape they have created. The requirements for the deformable sphere are summarised as follows:

- The sphere should be able to have multiple deformations performed on it.
- A deformation should be able to be undone and redone.
- The sphere should be able to be reset to its initial state.
- The sphere should be able to be moved around by the user and positioned anywhere in the VR space.

- The sphere should be able to be scaled up and down.
- The user should be able to select out all the data contained within the deformed sphere.
- The user can adjust the area of effect of deformation.

4.2 Control Lattice Structure

The structure of the lattice is vital in allowing the functionality listed above. Most importantly, the structure of the lattice is what allows for the area of effect (AOE) of deformation to be adjusted. This is because with more control points, there are more cells, meaning the control points of one cell may not even be effected by a deformation in another cell. The lattice is set up depending on the AOE setting set by the user. This sets the base number of control points in each dimension. This ranges from 6 to 13. Six was chosen as the minimum because four and five control point lattices produced strange deformations and often broke the mesh of the sphere. Depending on this value, control points (implemented with Vector3 objects in Unity) are equally spaced from the origin to 1 on the x, y and z axes. Control points are also added such that the lattice forms a cube-like structure. When the deformation calculations take place the sphere is embedded into this lattice, more specifically among all the cells in the lattice. The top row of figure 5 shows some examples lattices, the middle row shows how increasing the control points in a lattice changes the area of effect of deformation. Moreover, since the user can change the AOE at any time, the structure of the lattice can be changed at runtime. After each deformation the control points are reset to their initial positions. This means that the deformed sphere may now fall into a region of the lattice that is not a cell (does not have 64 control points surrounding it). In this event, the lattice is extended in the relevant dimensions by the same spacing as the rest of the control points. The bottom row of Figure 5 displays this with a deformation that causes extra control points to be added in the y-direction.

4.3 Deformation Pipeline

The following sub-section assumes the sphere is positioned as the user wants it before commencing a deformation task and has selected their desired AOE such that the lattice is setup as desired. As the user begins deforming, constraints are recorded at each frame by measuring the vector resulting from the direction of the controller's movement between frames. This vector, along with the starting position of the constraint relative to the sphere's local coordinate frame making up a single constraint C_i. This is then passed into the pipeline of deformation calculations. After the constraint is measured the sphere is transformed from its position in world space into the lattice so that the deformation calculations can be applied to the sphere. Since the lattice is constructed from the origin along the positive direction of each axis, the relevant cell for deformation using the starting point from C_i is calculated as follows:

$$Cell = (\lfloor \frac{x}{l} \rfloor, \lfloor \frac{y}{l} \rfloor, \lfloor \frac{z}{l} \rfloor) \quad (4)$$

Where the starting point is (x, y, z) and l is the step size between control points in the lattice. Using this, the 64 control points surrounding that cell can be accessed. Using these control points, as well as C_i, the new control point positions are calculated using the calculations from section 2.3.

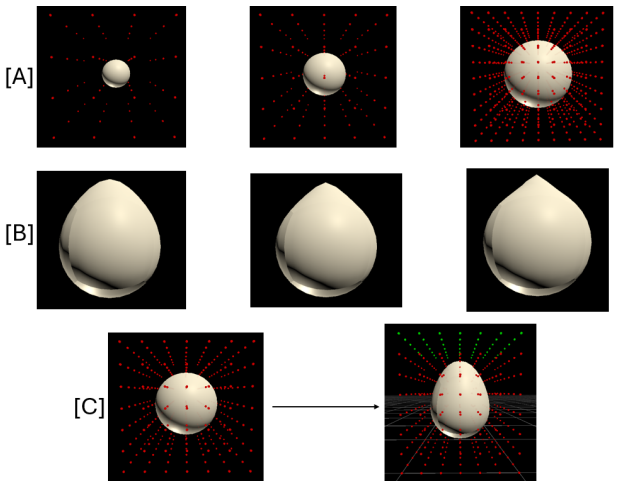


Figure 5: [A]: Lattice with 4, 5, and 9 control points in each dimension. [B]: Area of Effect, shown with the same constraint applied to lattices with an increasing number of control points. [C]: Lattice extended in Y axis after deformation, extension shown in green.

Following this, the sphere's vertices are passed through equation (1) where the cell each vertex lies in is calculated as in equation (4). If the cell is found to be one that does not have enough surrounding control points, the lattice is extended to accommodate this as described above. The vertices are then assigned their new positions and the deformed sphere is produced and transformed back to its initial position. This is happening so quickly the user can see their sphere deforming in front of them in its original position. To quantify this, I performed the same deformation with each AOE setting on the base sphere and averaged the time a single deformation calculation took for each. This is displayed in figure 6. The pipeline is summarised in figure 7.

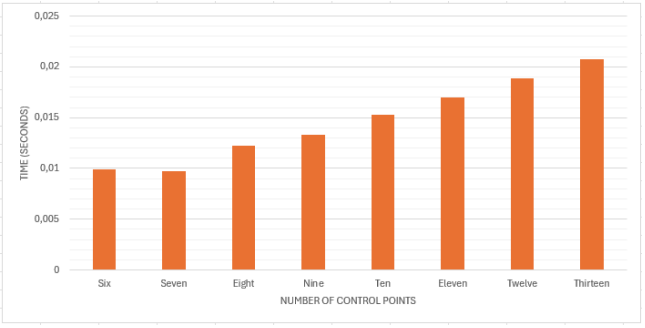


Figure 6: Average time to complete a single deformation calculation for an increasing number of control points in the control lattice

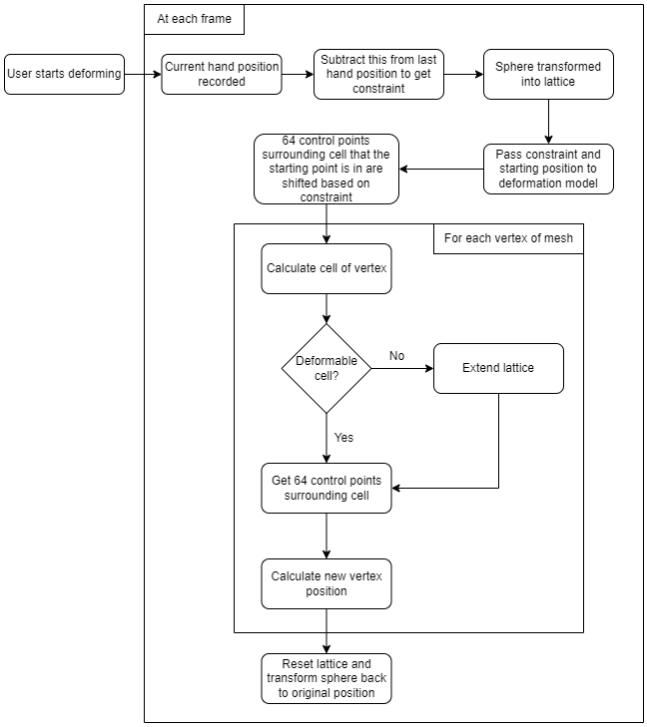


Figure 7: Flow Diagram displaying the Deformation Pipeline while a User Deforms.

4.4 Inside-Outside Test

To allow users to make sub-selections of the volumetric data in iDaVIE-v, simply developing a deformable sphere is not sufficient. A test to determine which data points fall within the sphere and which do not is needed. The volumetric data used for analysis in iDaVIE-v are called data cubes. Hence, the software displays the data as a cube with all the data contained within the cube. Figure 9 presents some example data cubes. The units of data that the cube comprises of are represented as voxels. Voxels are like pixels but with three dimensions. In effect, the cube is made up of much smaller cuboids. Hence, individual voxels need to be tested with the inside-outside test. However, testing every single voxel in the data is inefficient as the sub-regions that are selected are often much smaller than the entire cube. Keeping this in mind, a bounding-box is created around the sphere that updates as the sphere is scaled or deformed. Bounding boxes are commonly used in computer graphics to make approximations to the space around an object. A bounding box can be thought of as a cuboid that is as small as possible such that it encompasses a particular object. [22] In this case the bounding box encompasses the entire deformed sphere, so that only voxels in between the bounds of the box are tested.

The actual test makes use of ray casting from the position of the voxels toward the centre of the sphere's mesh. If a collision is detected with the sphere's surface, the normal vector at the point of intersection is dotted with the ray. If the result of this is positive, the voxel is inside the sphere, otherwise it is outside. This test works due to the mesh of the sphere always being enclosed, i.e. there are no holes in the mesh. This method is only full-proof for convex geometric objects.

4.5 Desktop Method

As mentioned previously, a desktop method for selecting out sub-regions of the volumetric data was developed in addition to the deformable sphere. This was to allow an experiment to be conducted to compare which, if any, environment was better. The desktop method allows users to move through the data in slices. In effect, the data is split up into planes along the axis desired by the user. The user can them move between these planes, and in each plane use their mouse pointer to draw loops around areas of interest. Figure 8 displays the steps taking to make a selection on a slice of a data cube using the desktop method.

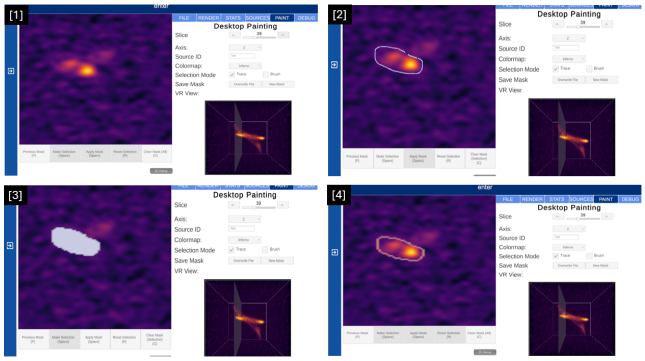


Figure 8: Steps for making a selection using the desktop method. 1: Screen before starting a selection. 2. Outline of desired selection made. 3: Outline confirmed, screen shows temporary area that will be selected if confirmed. 4: Selection confirmed

5 EXPERIMENTAL DESIGN

To test whether VR allows for more accurate, and a more usable environment for sub-volume selection of volumetric data, user experiments were conducted. Thirty users from the science faculty at the University of Cape Town were recruited to undergo the tests. We chose to have thirty participants so that our group could have a range of different people testing the software. In addition to this, it ensures that a few outliers could be removed in the case that this is needed. Moreover, we ran two pilot experiments to outline any issues in the experimental procedure and make adjustments.

5.1 Outline of the Experiment

A script was prepared for the experiment to ensure all users received the same information. The experiment proceeded as follows:

(1) Each user began by filling out a participation form in which they were presented with all potential risks of the experiment and gave consent to undergo the experiment. They also provided their email and name to be entered into a raffle.
(2) Following this a short five minute navigation training was conducted to allow the users to get familiar with the controls of iDaVIE-v. This involved showing users how to move, scale,

and rotate the data cube as well as how to select options within the menu of the software.

(3) Once completed, users tested four different selection methods. These methods were the desktop method, the deformation method, a lasso selection method, and a shape selection method. The latter two methods are not discussed in this paper and were developed by two colleagues.
(4) Since each user would be testing four methods in each session, they would get more comfortable with the system with each method, causing a potential for skewed results, this is called a learning effect. More formally, a learning effect is the cost reduction effect due to technological experience, where the cumulative experience with a technology leads to improved performance over time [16]. To combat this, we ensured that each user had a random permutation order of the four methods. To ensure consistency, we made sure that the permutations allowed for each method to be in each position in the order an equal number of times.
(5) For each method the user was first trained with that method for five minutes.
(6) Following this, they were presented with a data cube, the desired sub-region they were required to select out of the cube, and the ideal selection of that sub-region for which their selection would be compared against.
(7) The experiment had five different data cubes of varying difficulty. Each selection method during the experiment was conducted with a different cube and there was one training cube used for all the training. The assignment of data cubes to selection methods was also permuted for each experiment, the training cube remained constant. Figure 9 displays these cubes.
(8) The user then had ten minutes to select the region of interest with the method they had just been trained with. Once completed, the user filled out a questionnaire - the metrics of which are discussed in section 5.2.

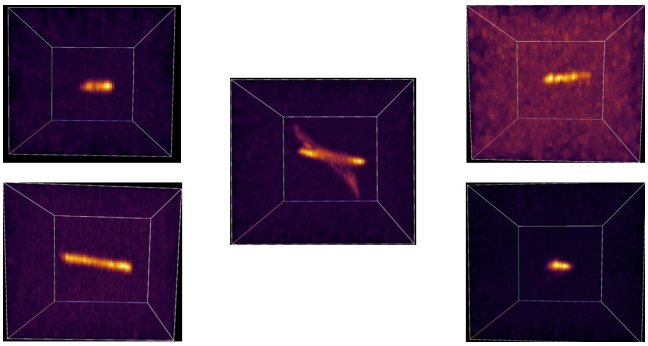


Figure 9: Data Cubes used to conduct the experiment

5.2 Metrics

5.2.1 Completion Time. To gauge how efficient the selection methods were the time taken to complete the selection with each respective method was measured.

5.2.2 Selection Accuracy. Selection accuracy involves observing the degree to which the intended selections are made against the actual selection performed. There are two common metrics used to measure the accuracy of a selection method: F_1 Score and the Matthews Correlation Coefficient (MCC). The following table outlines the terms used in these metrics:

Table 1: Variables for Selection Accuracy

Variable	Abbreviation	Description
True Positives	*TP*	Correctly selected points
False Positives	*FP*	Incorrectly selected points
False Negatives	*FN*	Missing points that had to be selected
True Negatives	*TN*	Correctly non-selected points

F_1 Score: Yu et al. [27] states that *the F_1 score calculates the harmonic mean of precision and recall and is often used in information retrieval to measure query classification performance.* It is defined as $F_1 = P \cdot R/(P + R)$ where P is precision and is calculated by $P = TP/(TP+FP)$ and R is recall (the fraction of particles that were selected) and is calculated by $R = TP/(TP + FN)$.

The F_1 score does not take TN into consideration, MCC provides a metric in which it is. MCC is defined as follows:

$$MCC = \frac{TP \cdot TN - FP \cdot FN}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}}$$

MCC is most commonly used as a performance metric for binary classifiers in machine learning. It provides an unbiased performance metric since it accounts for true negatives (TN). Yao [25] even goes as far to say that F_1 score should be deprecated due to its bias. We opted to use both metrics to obtain more data for analysis. Moreover, both of these metrics are commonly used in conjunction with one another when evaluating the selection accuracy of a selection technique. A "perfect" selection in each cube was created prior to the experiment, users were shown this and asked to replicate it. Their selection was the measured against this one whereby these two accuracy metrics were used.

5.2.3 Task Load Index (TLX). The task load index is a tool for measuring and assessing the mental workload a task has on the user performing that task. TLX was developed by NASA and is sometimes referred to as NASA-TLX [14]. TLX is evaluated on a few sub-metrics that are then weighted and combined to form the overall TLX value. These sub-metrics are mental demand, physical demand, temporal demand, effort, performance and frustration. After each selection method was tested, the user assigned a rating of 1-20 for each of those fields.

5.2.4 System Usability Scale (SUS). SUS is a scale that measures the usability of a system [1]. This scale requires users to answer various questions, these questions are answered via a rating from 1-5. These values are then used in a calculation to provide the overall usability on a scale from 0-100. The average SUS value is 68, meaning a score above that indicates a good degree of usability. Users answered these questions for each selection method after using it.

5.3 Ethics

To ensure our experiment maintained ethical practices, we obtained ethics approval from the ethics committee at the University of Cape Town. We also obtained explicit approval to use UCT students as users for our experiments. Our participation form ensured that we got informed consent from our users where they were made aware of the procedures of the experiment, the data collected in the experiment, as well as risks presented by the experiment. Our experiment only had one major risk, that being simulation sickness. Dużmańska et al. [4] defines simulation sickness as *a syndrome similar to motion sickness and can be experienced as a side effect during and after exposure to different virtual reality environments.* Symptoms include discomfort, drowsiness, nausea, disorientation, fatigue, nausea, and in some cases vomiting. Users were allowed to tap out of the experiment at any point without consequence if they experienced any of the symptoms above. There were no cases of simulation sickness recorded within the experiment.

6 RESULTS

This section outlines the results of the experiment discussed in section 5. The accuracy, efficiency, and usability of both the desktop and VR (deformation) methods were compared for significance. Prior to testing for significance, the data needed to be tested to see if it follows a normal distribution, as this will determine which statistical test we could use. Two tests were used to analyze whether the data followed a normal distribution: Shapiro-Wilk [24] and D'Agostino's K^2 tests [3]. Due to our small sample size of 25, the Shapiro-Wilk test holds more weight as it has been shown to provide better results for smaller sample sizes than D'Agostino's K^2 test [26]. D'Agostino's K^2 test compares the skewness and kurtosis of the data to that of a normal distribution, whereas Shapiro-Wilk observes whether the data follows a normal distribution by comparing the order statistics to the expected values of a normal distribution. These tests indicated that not all of four of the selection methods tested in the experiment followed a normal distribution for all the metrics. Hence, we ran a Friedman Test for the data as a whole, followed by Wilcoxen Signed-Rank test to test for significance between pairs of selection methods. Finally, the Cohen's d effect size was determined for each pair to determine to what degree there was significance between the two methods. This is conducted for each metric.

6.1 Accuracy

The summarised accuracy results given by the F1 Score and MCC values are shown in Figure 10,11 and Figure 12,13 respectively. Deformation shows to have slightly better, but negligible, means for both metrics. Moreover, the Wilcoxen Signed-Rank p-value is significantly greater than 0.05, and hence we can accept that there is no significance between the two methods for either metric. This is further supported by the small effect sizes.

	Desktop	Deformation (VR)
Mean	60.32	61.39
SD	16.14	17.82
Shapiro-Wilk p-value	0.016	0.0302
D'Agostino's K^2 p-value	0.0053	0.0021
Is normally distributed?	No	No
Wilcoxen Signed-Rank p-value	0.6837	
Cohen's d effect size	0.0590	

Figure 10: F1 Score results for Desktop and Deformation (VR) methods

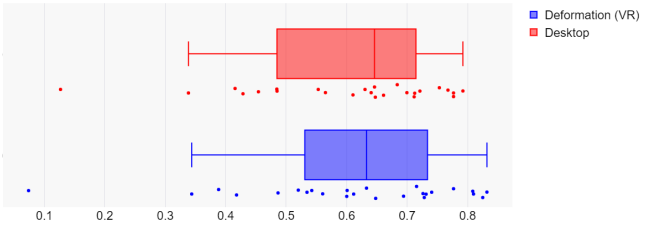


Figure 11: Comparison of F1 Score between Desktop and Deformation (VR) methods

	Desktop	Deformation (VR)
Mean	63.65	63.68
SD	13.80	16.55
Shapiro-Wilk p-value	0.0105	0.0035
D'Agostino's K^2 p-value	0.0164	0
Is normally distributed?	No	No
Wilcoxen Signed-Rank p-value	0.8462	
Cohen's d effect size	0.0288	

Figure 12: MCC results for Desktop and Deformation (VR) methods

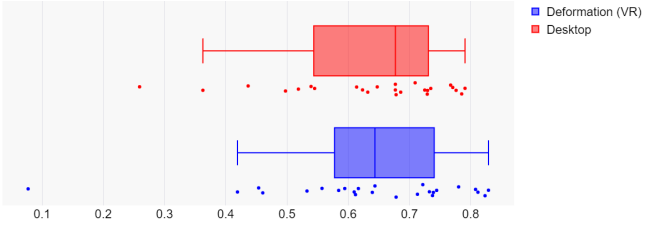


Figure 13: Comparison of MCC between Desktop and Deformation (VR) methods

6.2 Completion Time

The summarised completion time results are displayed in Figure 14 and 15. Completion time was measured in seconds. The mean completion time for desktop is better than the deformation method. This significance between the two methods is further supported by the Wilcoxen Signed-Rank p-value which is smaller than 0.05. However, the effect size is shown to be small indicating that the significance is not large between the two methods.

	Desktop	Deformation (VR)
Mean	333.76	413.16
SD	130.90	128.05
Shapiro-Wilk p-value	0.532	0.3559
D'Agostino's K^2 p-value	0.6316	0.4777
Is normally distributed?	Yes	Yes
Wilcoxen Signed-Rank p-value	0.0344	
Cohen's d effect size	0.301	

Figure 14: Completion Time results for Desktop and Deformation (VR) methods

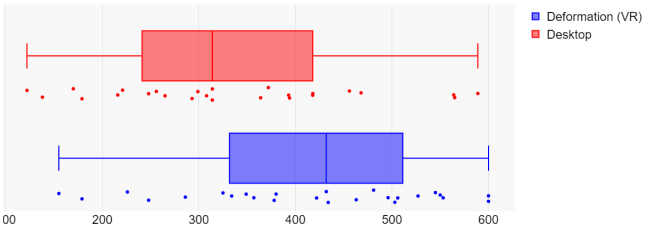


Figure 15: Comparison of Completion Time between Desktop and Deformation (VR) methods

6.3 System Usability Scale (SUS)

The summarised SUS results are displayed in Figure 16 and 17. The mean value for desktop shows to be slightly better than the deformation method. However the Wilcoxen Signed-Rank p-value is much larger than 0.05, and hence this difference is not significant. This is further supported by the small effect size.

	Desktop	Deformation (VR)
Mean	75.4	71.3
SD	16.45	19.61
Shapiro-Wilk p-value	0.01775	0.4175
D'Agostino's K^2 p-value	0.2587	0.6260
Is normally distributed?	No	Yes
Wilcoxen Signed-Rank p-value	0.4840	
Cohen's d effect size	0.1	

Figure 16: SUS results for Desktop and Deformation (VR) methods

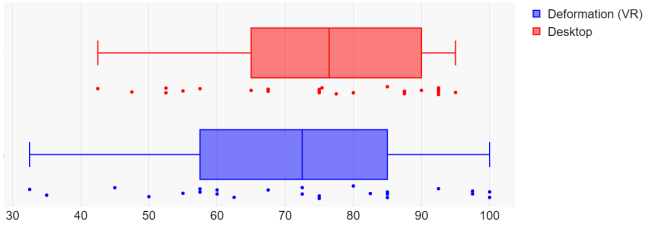


Figure 17: Comparison of SUS between Desktop and Deformation (VR) methods

6.4 Task-Load Index (TLX)

As mentioned in section 5.3.3, TLX is composed of six sub-metrics. Since it was decided to use the non-weighted TLX the analysis of these six metrics has been done separately. The results for each are almost identical, showing that the desktop method has slightly better means for each metric. However, the Wilcoxen Signed-Rank p-value and effect sizes for each metric indicate that these were not significant differences. The tables and box diagrams displaying the data have been moved into the Appendix in the interest of space and redundancy.

7 DISCUSSION

In this section the results presented in the previous section are discussed. Generally, it appears there is no significant difference between the desktop and deformation selection methods. Of all the metrics we assessed the methods on, only the completion time of selection tasks showed to have significance. Even so, the effect size of this significance was small. The other metrics (accuracy, usability, workload) all showed no significance between the two methods. The accuracy results showed that both desktop and the deformation method had similar selection accuracy. The deformation method had a slight improvement in the mean for both metrics; 1.8% for F1 Score and 0.05% for MCC. The deformation method also had a greater variability in selection accuracy than desktop due to the greater standard deviation for both metrics. These slight differences are not enough to draw significance between the two methods with respect to selection accuracy; the Wilcoxen Signed-Rank p-values for F1 score (0.6837) and MCC (0.8462) are indicative of this. It should be noted that the deformation method had a bug that had a minor effect on the accuracy results. The mesh used for the deformable sphere in Unity would sometimes break and have holes causing a few incorrect voxels to be selected. This happened only in around 10% of user tests, and with these only caused at most a 1% error in the accuracy value.

The SUS results indicate that the desktop method showed a slight improvement in the mean SUS score (5.8%). Moreover, deformation showed more variability with a standard deviation of 19.61 compared to desktop with 16.45. Again, the Wilcoxen Signed-Rank p-value (0.4840) and effect size (0.1) indicate that this is not a significant difference between the two. The slight improvement in the desktop method could be attributed to a lower familiarity with VR than a desktop environment amongst the users.

The TLX results indicated that the desktop provided slightly less of a workload for the users compared to the deformation method in each sub-metric. For all metrics except mental demand, the desktop method had a larger standard deviation, indicating a larger variability for the desktop method. The Wilcoxen Signed-Rank p-values and effect sizes for all six metrics indicate no significance between the two methods. These values are summarised below in Table 2: The only metric to show significance was completion time. The results showed that on average users completed selections 79.4 seconds faster with the desktop method compared to the deformation method, meaning the deformation method is 24% less efficient than the desktop method. The two methods had roughly the same variability. The Wilcoxen Signed-Rank p-value (0.0344) indicates that the difference between them is significant. The effect size (0.301)

Table 2: The Wilcoxen Signed-Rank p-values and effect sizes for TLX sub-metrics

Metric	Wilcoxen p-value	Effect size
Physical Demand	0.2346	0.451
Mental Demand	0.2346	0.169
Temporal Demand	0.1947	0.185
Effort	0.496	0.0978
Performance	0.2410	0.167
Frustration	0.0972	0.236

indicates that this is a small significance between the two. This difference in completion time could be due to a couple of things. Firstly, as stated previously, users likely have more experience with a desktop environment than VR. Secondly, we had very limited time to train users with the VR system without making the user experiments too long. With the timeline of this project it was not feasible to allow users more time to get accustomed to the VR space, as well as the deformation method.

In addition to the results presented above, a power analysis was conducted to determine if it is feasible for us to draw conclusions from the data we recorded. The power analysis uses the significance level desired (0.05 in this case), sample size, and effect size of the data to produce a power value. This value is a percentage which indicates to what degree your sample and effect sizes were large enough to accurately draw conclusions about the significance of the data. A value of 80% + is usually desired. The power for each metric is displayed in Table 3. As can be seen in the table, these values are well below 80%. This means that for us to reliably draw conclusions from the data either our sample size or effect sizes between the methods needed to be larger. Moreover, five outliers from the data were removed reducing our sample size to 25. We removed three participants as the system bugged during their tests causing them to not complete their experiment. One participant was removed as their data was overwritten. The final outlier that was removed was a user who had struggled getting accustomed to the system and resulted in their data being significantly different from the rest of the users. They had close to 0% accuracy for all the methods, and hence their data affected the results heavily prior to removal.

Table 3: Power values for each metric

Metric	Power (%)
F1 Score	34.21
MCC	40.17
Completion Time	34.84
SUS	28.11
Physical Demand	25.32
Mental Demand	37.14
Temporal Demand	28.69
Effort	12.75
Performance	13.88
Frustration	29.57

8 CONCLUSIONS

This paper aimed to determine whether virtual reality offers a better environment for sub-volume selection of volumetric data than a desktop environment. More specifically, whether VR allows for more accurate selections, a more usable environment, and creates a lower workload for the user. To test this, two selection methods were developed: a desktop method in which the user could pass through the volumetric data slice by slice, and a VR method, where the user could deform a sphere to select out regions of interest. User experiments were then conducted where users were trained with the methods, and then used them to select out a region of interest from a data cube. These experiments measured the time it took to complete the selection task, the accuracy of the selection task, the usability of the selection method, and the workload the selection method created for the user. The results indicated that there was no difference in either method in every metric apart from completion time, for which the desktop method performed better. Hence, it cannot be concluded that virtual reality offers a better environment for sub-volume selection of volumetric data compared to a desktop environment. Furthermore, it cannot be reasonably concluded that the desktop environment is better than VR as only one of the metrics were indicative of this.

The timeline of this project made it infeasible to run more than one set of user tests. Future work could include rerunning these experiments, with more participants and longer training to gather a better set of results to draw conclusions from. Moreover, the issues faced with the deformation method could be resolved to ensure no bugs occur. In addition to these improvements, the deformation method could have extended functionality to make it more intuitive to the user. This could include two-handed deformation, whereby the user could deform the sphere with both hands simultaneously. The model could also be extended to prevent self-intersection of the sphere during deformation. Finally, the sphere could make better use of the many degrees of freedom the VR controllers offer to incorporate twisting of the controllers to create deformations. Different starting shapes could also be included so that the user can deform an array of different objects.

Overall, this project can be deemed a success. Although the results did not indicate that VR provides a clear improvement to a desktop environment, the methods developed and the experiments conducted provided insight that can be used to further test this idea in the future. Moreover, it provided new features to the iDaVIE-v software that may be used to help astronomers analyze data around the world.

REFERENCES

[1] Aaron Bangor, Philip T. Kortum, and James T. Miller. 2008. An Empirical Evaluation of the System Usability Scale. *International Journal of Human–Computer Interaction* 24, 6 (2008), 574–594. https://doi.org/10.1080/10447310802205776

[2] Pierre Bézier. 1978. General distortion of an ensemble of biparametric surfaces. *Computer-Aided Design* 10, 2 (1978), 116–120. https://doi.org/10.1016/0010-4485(78)90088-X

[3] Ralph B. D'Agostino. 1972. Small Sample Probability Points for the D Test of Normality. *Biometrika* 59, 1 (1972), 219–221. https://doi.org/10.2307/2334638

[4] Natalia Dużmańska, Paweł Strojny, and Agnieszka Strojny. 2018. Can Simulator Sickness Be Avoided? A Review on Temporal Aspects of Simulator Sickness. *Frontiers in Psychology* 9 (2018). https://doi.org/10.3389/fpsyg.2018.02132

[5] Amy Ulinski et al. 2007. Two Handed Selection Techniques for Volumetric Data. *IEEE Symposium on 3D User Interface* (2007). https://doi.org/10.1109/3DUI.2007.340782

[6] Daniel Ströter et al. 2024. A Survey on Cage-based Deformation of 3D Models. *Computer Graphics Forum* 43, 2 (2024), e15060. https://doi.org/10.1111/cgf.15060 arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1111/cgf.15060

[7] Lucia Marchetti et al. 2021. iDaVIE-v: immersive Data Visualisation Interactive Explorer for volumetric rendering. *Zenodo* (2021). https://doi.org/10.48550/arXiv.2012.11553

[8] Michael J. McGuffin et al. 2003. Using deformations for browsing volumetric data. (2003), 401–408. https://doi.org/10.1109/VISUAL.2003.1250400

[9] Nadia Magnenat-Thalmann et al. 1988. Joint-dependent local deformations for hand animation and object grasping. *Proceedings of Graphics Interface '88* (1988), 26–33. https://doi.org/handle/20.500.14299/238891

[10] Paolo Serra et al. 2015. SoFiA: a flexible source finder for 3D spectral line data. *Monthly Notices of the Royal Astronomical Society* 448, 2 (02 2015), 1922–1929. https://doi.org/10.1093/mnras/stv079

[11] Thomas Jarrett et al. 2021. Exploring and Interrogating astrophysical data in virtual reality. *Astronomy and Computing* 37 (2021). https://doi.org/10.1016/j.ascom.2021.100502

[12] James Gain. 2000. Enhancing Spatial Deformation for Virtual Sculpting. *St. John's College, University of Cambridge* (09 2000).

[13] Gene Golub and Charles Van Loan. 1990. Matrix computations (2nd edition). *The Mathematical Gazette* 74, 469 (1990), 322–324. https://doi.org/10.2307/3619868

[14] Sandra Hart and Loell Staveland. 1988. Developing the NASA-tlx (Task Load Index): Results of theoretical and empirical research. *Advances in Psychology* 52 (1988), 139–183. https://doi.org/10.1016/S0166-4115(08)62386-9

[15] William M. Hsu, John F. Hughes, and Henry Kaufman. 1992. Direct manipulation of free-form deformations. *SIGGRAPH Comput. Graph.* 26, 2 (jul 1992), 177–184. https://doi.org/10.1145/142920.134036

[16] Seunghyok Kim, Jamin Koo, and En Sup Yoon. 2011. Optimization of sustainable energy planning with consideration of uncertainties in learning rates and external cost factors. In *21st European Symposium on Computer Aided Process Engineering*, E.N. Pistikopoulos, M.C. Georgiadis, and A.C. Kokossis (Eds.). Computer Aided Chemical Engineering, Vol. 29. Elsevier, 1914–1918. https://doi.org/10.1016/B978-0-444-54298-4.50161-6

[17] John Lucas, Doug Bowman, Jian Chen, and Chad Wingrave. 2005. Design and Evaluation of 3D Multiple Object Selection Techniques. *Virgina Tech* (2005). https://doi.org/10919/31769

[18] Fourcan Karim Mazumder and Utpal Kanti Das. 2014. Usability guidelines for usable user interface. *International Journal of Research in Engineering and Technology* 3, 9 (2014), 79–82. https://doi.org/10.15623/ijret.2014.0309011

[19] Amandine Menasria, Pierre Brenner, and Paola Cinnella. 2018. Wall treatment of multiple correction k-exact schemes. *International Conference on Computational Fluid Dynamics* (07 2018).

[20] Robert Montano, Cuong Nguyen, Rubaiat Habab Kazi, and Sriram Subramanian. 2020. Slicing-Volume: Hybrid 3D/2D Multi-target Selection Technique for Dense Virtual Environments. *2020 IEEE Conference on Virtual Reality and 3D User Interfaces (VR)* (2020), 53–62. https://doi.org/10.1109/VR46266.2020.1581198507712

[21] William H. Press, Saul A. Teukolsky, William T. Vetterling, and Brian P. Flannery. 1992. *Numerical recipes in C (2nd ed.): the art of scientific computing.* Cambridge University Press, USA.

[22] Philip Schneider and David H Eberly. 2002. *Geometric tools for computer graphics.* Elsevier.

[23] Thomas W. Sederberg and Scott R. Parry. 1986. Free-form deformation of solid geometric models. *SIGGRAPH Comput. Graph.* 20, 4 (aug 1986), 151–160. https://doi.org/10.1145/15886.15903

[24] Samuel S. Shapiro and Martin B. Wilk. 1965. An Analysis of Variance Test for Normality (Complete Samples). *Biometrika* 52, 3/4 (1965), 591–611. https://doi.org/10.2307/2333709

[25] Jingxiu Yao and Martin Shepperd. 2020. Assessing software defection prediction performance: why using the Matthews correlation coefficient matters. In *Proceedings of the 24th International Conference on Evaluation and Assessment in Software Engineering* (Trondheim, Norway) *(EASE '20)*. Association for Computing Machinery, New York, NY, USA, 120–129. https://doi.org/10.1145/3383219.3383232

[26] Berna Yazici and Senay Asma. 2007. A comparison of various tests of normality. *Journal of Statistical Computation and Simulation - J STAT COMPUT SIM* 77 (02 2007), 175–183. https://doi.org/10.1080/10629360600678310

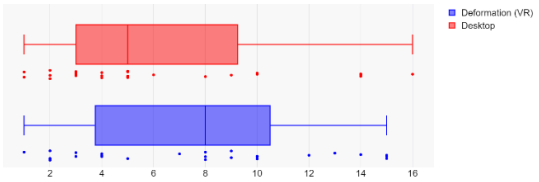
[27] Lingyun Yu and Konstantinos Efstathious. 2012. Efficient Structure-Aware Selection Techniques for 3D Point Cloud Visualizations with 2DOF Input. *IEEE Transactions on Visualization and Computer Graphics* (2012), 2245–2254. https://doi.org/10.1109/TVCG.2012.217

APPENDIX

A1 – TLX Results

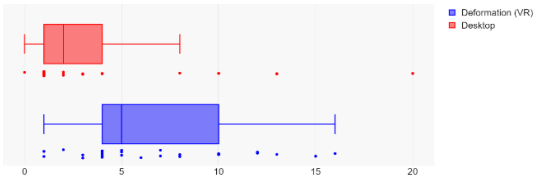
Mental Demand

	Desktop	Deformation (VR)
Mean	6.12	7.72
SD	4.50	4.57
Shapiro-Wilk p-value	0.003306	0.0731
D'Agostino's K^2 p-value	0.11	0.3772
Is normally distributed?	No	Yes
Wilcoxen Signed-Rank p-value	0.2346	
Cohen's d effect size	0.169	



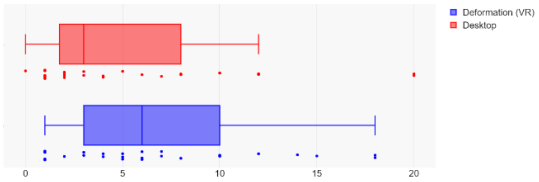
Physical Demand

	Desktop	Deformation (VR)
Mean	3.88	6.76
SD	5.01	4.26
Shapiro-Wilk p-value	0	0.0638
D'Agostino's K^2 p-value	0	0.2705
Is normally distributed?	No	Yes
Wilcoxen Signed-Rank p-value	0.2346	
Cohen's d effect size	0.451	



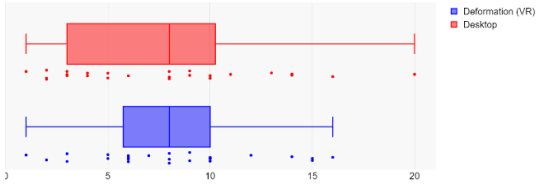
Temporal Demand

	Desktop	Deformation (VR)
Mean	5.48	6.88
SD	5.59	5.13
Shapiro-Wilk p-value	0.0002	0.015
D'Agostino's K^2 p-value	0.0005	0.1442
Is normally distributed?	No	No
Wilcoxen Signed-Rank p-value	0.1947	
Cohen's d effect size	0.185	



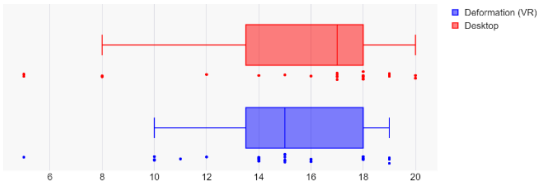
Effort

	Desktop	Deformation (VR)
Mean	7.6	8.16
SD	5.02	4.09
Shapiro-Wilk p-value	0.1055	0.4858
D'Agostino's K^2 p-value	0.2990	0.7247
Is normally distributed?	Yes	Yes
Wilcoxen Signed-Rank p-value	0.496	
Cohen's d effect size	0.0978	



Performance

	Desktop	Deformation (VR)
Mean	15.12	14.64
SD	4.92	3.5
Shapiro-Wilk p-value	0	0.0326
D'Agostino's K^2 p-value	0.0264	0.1087
Is normally distributed?	No	No
Wilcoxen Signed-Rank p-value	0.2410	
Cohen's d effect size	0.167	



Frustration

	Desktop	Deformation (VR)
Mean	3.96	4.84
SD	5.61	4.53
Shapiro-Wilk p-value	0	0.0004
D'Agostino's K^2 p-value	0	0.0044
Is normally distributed?	No	No
Wilcoxen Signed-Rank p-value	0.0972	
Cohen's d effect size	0.236