Efficient Structure-Aware Selection Techniques for 3D Point Cloud Visualizations with 2DOF Input

Lingyun Yu, Konstantinos Efstathiou, Petra Isenberg, and Tobias Isenberg, *Member, IEEE*

Abstract—Data selection is a fundamental task in visualization because it serves as a pre-requisite to many follow-up interactions. Efficient *spatial* selection in 3D point cloud datasets consisting of thousands or millions of particles can be particularly challenging. We present two new techniques, TeddySelection and CloudLasso, that support the selection of subsets in large particle 3D datasets in an interactive and visually intuitive manner. Specifically, we describe how to *spatially* select a subset of a 3D particle cloud by simply encircling the target particles on screen using either the mouse or direct-touch input. Based on the drawn lasso, our techniques automatically determine a bounding selection surface around the encircled particles based on their density. This kind of selection technique can be applied to particle datasets in several application domains. TeddySelection and CloudLasso reduce, and in some cases even eliminate, the need for complex multi-step selection processes involving Boolean operations. This was confirmed in a formal, controlled user study in which we compared the more flexible CloudLasso technique to the standard cylinder-based selection technique. This study showed that the former is consistently more efficient than the latter—in several cases the CloudLasso selection time was half that of the corresponding cylinder-based selection.

Index Terms—3D interaction, spatial selection, direct-touch interaction.

1 Introduction

In scientific visualization, researchers are often interested in various physical properties of objects/regions to analyze their structure and context. To complete such analysis tasks it is essential to first select the objects of interest from their environment. While this is reasonably easy if one faces only a few objects which are also relatively large, the selection becomes a challenge if the dataset consists of thousands or millions of particles. While it is sometimes possible to filter particles according to some known data properties besides particle position, this may not always be the case because such properties may not yet be known or may not even exist. In such cases the only accessible possibility may be the interactive *spatial* selection of subsets of the particles by specifying a region in space in which the targeted particles are located. Therefore, interactive spatial selection is essential for many visualization applications and domains. A spatial selection also permits people to employ Boolean operations that involve several selection regions. However, when dealing with data in a three-dimensional domain, such a spatial selection often becomes a tedious multi-step process because it is difficult for people to specify the correct enclosing 3D surface for the spatial region of interest.

Our goal was thus to design new structure-aware selection techniques that facilitate the selection of multiple objects at a time by specifying *spatially* which regions are important, without a complicated multi-step process. A major design requirement for a new selection technique is that it should be as easy to use as freeform lassos for 2D particle selection or as raycasting-based methods for selecting single objects in 3D. It is important to realize that raycasting based techniques are unsuitable for our problem due to the impracticality of selecting thousands of particles in a serial manner. Instead, we are inspired by lasso-based methods such as Lucas and Bowman's [21] Tablet Freehand Lasso which selects 3D points within the lasso-specified generalized cylinder and we, therefore, refer to it as CylinderSelection. Such techniques are better suited for the selection of a large number of particles but have to be combined with subsequent selections from different angles through Boolean operations.

To facilitate an easy and intuitive selection in 3D particle datasets we present two new methods, TeddySelection and CloudLasso. For both techniques, the interaction is based on a 2D lasso that the user draws onto the 2D projection of the 3D space. The methods then derive an appropriate 3D selection geometry, taking the full spatial structure of the dataset into account. TeddySelection is inspired by interaction techniques from sketch-based modeling [11]. Based on a heuristic that takes into account the local particle density, the area enclosed by the user-drawn 2D lasso is inflated and fitted to the region of space that the user intended to select. CloudLasso uses the seminal Marching Cubes method [20, 37] to identify and select the regions inside the lasso where the density or the value of another scalar property is above a threshold. Both our techniques can be employed not only in traditional mouse-based interaction but are also, in particular, suitable for direct-touch visualization environments. In fact, our work was motivated by a direct need for an enhanced spatial selection mechanism using direct-touch input in the domain of astronomy for particle data such as numerical simulations of the gravitational processes of stars or galaxies. Nevertheless, the new techniques presented here are applicable to any 3D point cloud dataset such as 3D scatter plots in information visualizations or particle flow simulations in physics, and can also be extended to allow selection based on other scalar properties besides density. The created 3D selection surfaces can also be used in an off-line process to enable the processing of datasets whose sizes do not fit into main memory.

In the remainder of the paper we first review related work in Section 2. Then, we describe our own selection techniques in detail: TeddySelection in Section 3 and CloudLasso in Section 4. Next, we present the results of a formal, controlled user study in Section 5 where we compared CloudLasso to CylinderSelection. In Section 6 we compare all three methods, discuss possible extensions of TeddySelection and CloudLasso, and consider applications of these methods in general visualization contexts. We conclude the paper in Section 7.

2 Related Work

Several techniques for selecting objects in 2D environments have been developed in the past. These include selections by clicking on the target (picking), brushing where users pass over several target objects, spatial selections by clicking and dragging a lasso to create a selection area, or selections based on property masks such as color. Two-dimensional selection is considerably easier than selection in 3D because the 2D space is fully visible and accessible for interaction. Nevertheless,

- *Lingyun Yu, Konstantinos Efstathiou, and Tobias Isenberg are with the University of Groningen, the Netherlands. E-mail: {lingyun.yu@ | k.efstathiou@ | isenberg@cs.}rug.nl.*
- *Tobias Isenberg is also with DIGITEO/INRIA/CNRS, France.*
- *Petra Isenberg is with INRIA, France. E-mail: petra.isenberg@inria.fr.*

Manuscript received 31 March 2012; accepted 1 August 2012; posted online 14 October 2012; mailed on 5 October 2012.
For information on obtaining reprints of this article, please send e-mail to: tvcg@computer.org.

structure-aware selection that exploits the perceptual grouping of objects may outperform standard selection techniques even in 2D [6, 7].

For 3D data, raycasting selection techniques (e. g., [1, 18, 25, 36]) are common in virtual environments that range from desktop VR to CAVEs; for true volumetric displays similar techniques are used [10]. Raycasting is frequently employed because it can be operated at a distance, is fast, and is easily understandable [21]. Cone selection [19] is a related technique that selects the object whose location is closest to the cone center. To improve accuracy, shadow cone selection [31] selects groups of objects that lie within a cone projected from the interacting hand for the period of the selection interaction. This means that people need to move the input device to select a single object. While techniques generally based on ray casting are efficient for selecting single and large objects, they are less well suited for selecting small or occluded objects in cluttered environments. Thus, de Haan et al. designed IntenSelect [4] to assist users in selecting potentially moving objects in occluded and cluttered VR environments. Here, a constantly updated scoring function is calculated for all objects that fall within a user-controlled, conic selection volume. While interacting, a bending ray remains snapped to the highest ranking object to ease selection. To address the selection of small objects, Kopper et al. [17] developed SQUAD as a technique that is based on progressive refinement. 3D streamlines or tracts can be selected by drawing a 3D lasso around the objects of interest [15, 39] or using haptic input [13]. 3D picking also relates to 3D selection and is possible, e. g., via raycasting for distinct objects—even though object transparency may cause challenges with respect to what object was intended to be selected [23]. Without distinct objects, however, more advanced techniques like WYSIWYP are needed [33, 34], similar to our lasso-based selection.

Of course, there also exist techniques that enable 3D object selection based directly on a 3D position. Such techniques rely on rate controls such as 3D mice, tracked input hardware such as gloves, or non-wired 3D tracking such as Sixense's TrueMotion. Although they are easy to understand and manipulate, they are only feasible for object selection within the user's reach, unless combined with other non-linear mapping interaction techniques like the Go-Go technique [26].

While all these selection techniques—both based on raycasting and on direct 3D positions—are easy to understand and to operate, the time necessary for completing a selection increases with the number of targets. These methods are thus not suitable for datasets such as particle clouds where huge amounts of tiny objects often need to be selected. Selection-by-volume techniques like our own provide faster and more effective selection in these situations due to less repetition being necessary during the selection process. A related but sequential approach is taken by Elmqvist et al.'s [8] ScatterDice visualization system which selects multi-dimensional data through successive lasso selections in 2D scatterplots of the data. Ulinski et al. [32] use two-handed techniques for selecting the data in a subset of space. The two hands specify and manipulate a cuboidal selection volume. However, this technique may also include undesired objects because the desired structure in the 3D dataset typically does not have a cuboidal shape [21]. A more flexible technique is the Tablet Freehand Lasso method (CylinderSelection) by Lucas and Bowman [21] that lets people draw a lasso on a tracked physical canvas to extend a conical selection shape into the 3D space, as seen from the camera whose view is shown on the canvas. While this approach may still require complex multi-step Boolean operations to come to a final selection, it still serves as the foundation of our work: the lasso lets us take the dataset's 2D structure into account as visible in the projection. In our approach we significantly improve the original idea by considering the data's structure along the view direction. In the same spirit, Owada et al. [24] introduced Volume Catcher, a technique for unsegmented volume data that, based on a 2D stroke, selects a region of interest by applying a segmentation algorithm. Owada et al.'s technique, however, requires a precisely drawn stroke and is not directly applicable to particle datasets.

Our selection techniques target, in particular, applications in a directly-manipulative context such as visualization exploration on touch-sensitive displays. One reason for employing a direct-touch input metaphor lies in its performance improvements over mouse interaction, e. g., for target acquisition [16] and in form of more precise control over motions on the visualization display [38]. In this context also direct-touch interaction approaches with 3D environments play a role for which we refer to the overview by Isenberg and Hancock [12].

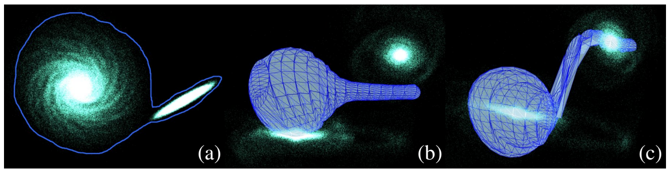


Fig. 1. Problem with Teddy-like placement of a selection mesh in the 3D data space: (a) lasso in the original view direction, drawn from a 'top-down' view; (b) selection mesh after a 90° rotation, placed at constant depth—the mesh does not wrap around the data; (c) depth adjusted to local average particle depth—the mesh still does not properly wrap around the data due to only being parameterized by the drawn lasso.

3 TeddySelection

To facilitate a lasso-based selection in 3D particle datasets, we need to find techniques that permit the creation of a 3D selection volume from a shape drawn in 2D. For our first technique, TeddySelection, we are thus inspired by Igarashi et al.'s [11] sketch-based Teddy modeling approach which, based on users' drawings of 3D shapes on a 2D surface, constructs a 3D mesh from the sketches. Teddy first triangulates the drawn shape and then extracts a *spine* in the triangulation's center. Then, the 2D shape is 'inflated' by determining the vertical depth of vertices on the spine from their distance to the drawn shape. Wide areas thus become fat, while narrow areas remain thin.

Relying on heuristics, we adjust the general Teddy approach to adapt the created mesh to the actual structure of the particles in the 3D point cloud dataset. More specifically, our selection algorithm consists of the following three main stages:

1. Input polygon triangulation: We compute a 2D triangulation between the spine and the drawn 2D outline, following [11].
2. Particle mapping to triangles: We determine which particles are located inside the 2D selection polygon and map each of them to their corresponding triangle in the triangulated mesh.
3. Construction of the selection mesh: We inflate the mesh by adjusting the vertices' depth based on the particle distribution.

3.1 Input Polygon Triangulation

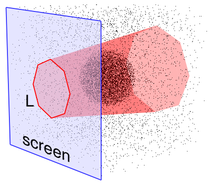
In the first step of our algorithm we follow Igarashi et al.'s [11] lead. We first convert the user-drawn input lasso into a closed stroke by connecting its start and end points (Fig. 2(a)). If the input stroke self-intersects we only consider its largest closed part, starting and ending at the intersection. To remove noise from the input device, we re-sample the input stroke with a uniform edge length before further processing which later-on also ensures a regular mesh polygon. Next, we determine the polygon's spine and create a complete 2D triangular mesh between the spine and the perimeter of the initial polygon.

3.2 Particle Mapping to Triangles

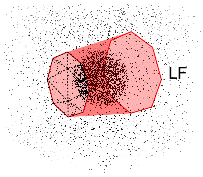
While the 'inflating' of the triangulated polygon in the original Teddy algorithm [11] is parameterized based on the local width of the drawn shape, we need to take both the location and distribution of the particles into account in this process. For this purpose we associate the particles with the generated 2D triangulation. Therefore, we render an ID-buffer of the triangulation and then check, for each particle's screen position, its association to a specific mesh triangle.

3.3 Construction of the Selection Mesh

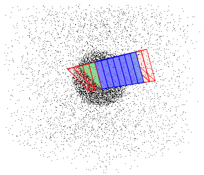
In the final step we need to inflate the 2D mesh and position it into the 3D space of the dataset. While Igarashi et al. [11] can simply use one constant depth for the entire mesh, this does not work well for structured datasets (see Fig. 1(b)). Also, a local adjustment of the depth of the mesh does not produce satisfying results. We can see in Fig. 1(c) that narrow regions still remain thin and wide regions



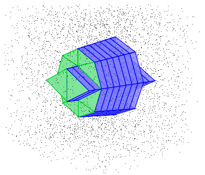
(a) Lasso is drawn on screen.



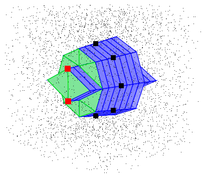
(b) First-level binning to obtain LF.



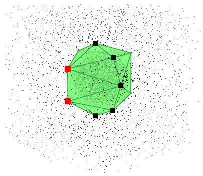
(c) Second-level binning for one triangle to obtain near & far planes.



(d) Near planes (green) for all triangles.



(e) Internal (red) and external (black) vertex inflation.



(f) Selection mesh inflation.

Fig. 2. Main steps of the TeddySelection algorithm.

become fat regardless of the particle distribution since the inflation parametrization solely relies on the drawn shape (compare Fig. 1(a) to Fig. 1(b)). Therefore, we perform a structural analysis by triangle of the produced 2D mesh. Generally speaking, our goal is to identify the closest and the furthest distance to the camera per triangle such that all dense clusters associated to the triangle lie between these two points. For this purpose we use a two-stage binning process to identify and remove sparsely populated regions.

The first binning stage (Fig. 2(b)) examines the whole generalized cone volume (i. e., only the part that is covered by the dataset) that is specified by the 2D lasso selection and splits it, at regular intervals in depth, into a certain number of bins (we use 100). We then calculate the number of particles within each of these bins. Next we threshold the bins and determine the closest and farthest bin with more than a given number of particles per volume unit. The specific particle number threshold depends on the dataset and can be adjusted. It is important to note that bins have different volumes, with bins closer to the viewer having a smaller volume. We call the section of the generalized selection cone between the front of the closest non-empty bin and the back of the farthest non-empty bin the *lasso frustum* (LF).

Then we perform a second binning (Fig. 2(c)), but this time by triangle of the 2D mesh to extract the local 3D structure of the dataset. Similar to the first binning, we split each generalized cone segment as defined by a mesh triangle into a number of bins (we use 60) and locate the closest and farthest *dense* bin. A bin is considered to be dense if its particle count is larger than a user-adjustable percentage of the expected particle count (we use a default of 400%, i. e., 4× the density average) if all particles were evenly distributed in the entire LF. This results in a depth range per triangle that includes all large clusters of particles between the front of the closest z_f and the back of the farthest z_b non-empty bin. An illustration of this result is shown in Fig. 2(d).

We now employ this second-level binning information to inflate the 2D mesh. First, we need to determine the exact z-depth of each 2D mesh vertex, both for the front and for the back part of the inflated mesh. Here we need to distinguish between internal vertices (the ones on the spine) and external vertices (the ones on the resampled sketched lines). For inflating the internal vertices we examined two approaches: using the average or using the most extreme 'non-empty' depth (front or back) of all adjacent triangles for a vertex. We experimented with both and found that both have advantages and disadvantages. By using the average depths of the adjacent triangles we obtain smoother 3D shapes but introduce errors in the form of some smaller clusters no longer being included in the selection volume due to the 'contraction' that happens because of the averaging. In contrast, the use of the most extreme depth values of all adjacent triangles results in a 3D shape that is not as smooth as the one based on averages but does ensure to include smaller clusters. We thus inflate the selection shape by moving all internal vertices to the extreme depths, while for each external vertex we average z_f and z_b of all its adjacent triangles to determine its location (Fig. 2(e)). Finally, the vertices are connected, resulting in a polygonal selection mesh that selects particles from the dataset spatially (Fig. 2(f)). In order to produce a smoother polygonal mesh we follow the original Teddy algorithm [11] and further subdivide fan triangles (triangles that connect the spine and the outside edge) to produce a gradual transition between spine and outside edge.

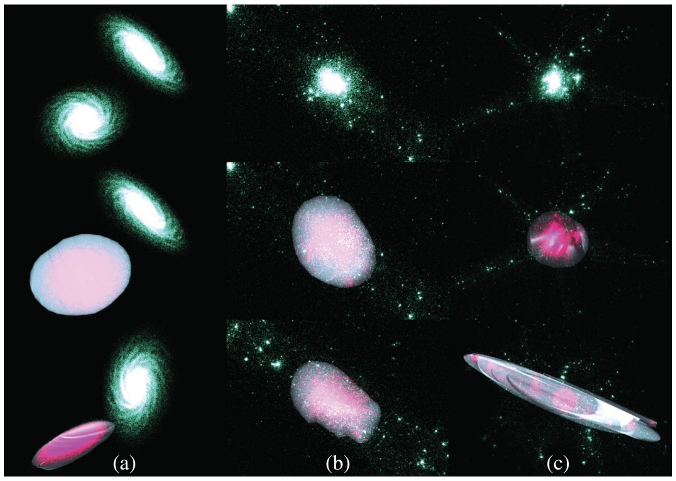


Fig. 3. Different dataset configurations (top), TeddySelections (middle), and selections viewed from another angle (bottom).

3.4 Example Results

Fig. 3 shows examples of applying TeddySelection to two astronomical datasets, a galaxy collision simulation in Fig. 3(a) and an N-body mass simulation in Fig. 3(b,c). The top row in Fig. 3 shows the dataset before the selection, the middle row shows the selection applied for this view, and the bottom row shows a different view of the selection to illustrate how the TeddySelection takes the 3D structure of the dataset into account. In Fig. 3(a) one of the galaxies was selected, resulting in a flat disk selection shape. Fig. 3(b) shows the selection of an almost spherical particle cluster. In the case of Fig. 3(c) it becomes clear after rotating the dataset (bottom), that the particle cluster is elongated and that the selection has taken this fact into account, creating a connected selection shape from the front to the back of the cluster. Note that in all cases the projection of the selection on screen is disk-shaped but TeddySelection takes into account the 3D structure of the selection, producing an appropriate selection geometry.

3.5 Performance

TeddySelection's processing time depends on the size of the dataset, the 2D size of the selection (number of triangles), and the particle count in the first-level selection volume. The selection on a 3.16 GHz Intel® Core™ 2 Duo CPU using a $1.6 \cdot 10^5$ particle dataset (Fig. 3(a)) takes 0.36 s. Selection from the $2.0 \cdot 10^5$ particle dataset for small (Fig. 3(b)) and larger clusters (Fig. 3(c)) takes 0.17 s and 0.21 s, resp.

3.6 User Feedback on TeddySelection

Our work was motivated by a direct need for an enhanced selection mechanism using direct-touch input in the domain of astronomy. To elicit feedback on our technique from expert users we interviewed three astronomers who regularly work with particle data such as numerical simulations of the gravitational processes of stars or galaxies. We demonstrated TeddySelection in individual sessions to the experts and discussed its benefits and drawbacks. We used both datasets shown in Fig. 3 (galaxy collision and cosmological N-body simulation) and asked the astronomers to experiment with the interactive selection.

For this purpose we presented the particle datasets on a 1920×1080 $52''$ touch-sensitive display. Our interface allowed the experts to explore the data by navigating through the 3D space [38] and facilitated fluid switching between navigation and selection.

Their feedback confirmed our motivation that spatial selections are essential for analyzing datasets, especially when properties about a subset of data such as the movement of stars in a particular region are of importance to the analysis. The astronomers' normal way of making selections is to rotate the dataset to find a good view, and then to use an unconstrained selection by means of a lasso or a selection rectangle. However, they reported that, generally, performing rotations to get a good view is tedious for them. Therefore, they much appreciated TeddySelection, commenting that our technique is helpful to find the prominent subset in depth without having to consider the view direction much. Similarly, the experts found it easy to select tails or arms of the collision galaxy dataset which was difficult to do previously. They also noted that the direct-touch interaction using the technique made it easy to draw precise lassos around the clusters of interest. As an addition they asked for a feature to make inverse selections (i. e., to select everything but the sketched regions) which is not yet possible with our prototype, but which can easily be included.

While the astronomers did not comment on further issues, TeddySelection has three limitations. One is that TeddySelection does not work well in generally sparse regions due to the noise contained therein. For example, if one zooms into a dataset too much the particles become sparsely distributed so that selections performed in such regions are less predictable. However, in such situations the structure of the data is less important so that cuboidal selections [32] work generally better. A second limitation is that the chosen view direction can have an effect if dense clusters lie visually behind the intended selection, and because regions between dense clusters in the front and in the back are always included. The need for structure in the data to constrain the selection—the third limitation—also means that the TeddySelection is not as useful in complex environments which contain many small, evenly distributed clusters. While the last two limitations can be addressed by a small change in view direction or by Boolean combinations of Freehand Lasso selections [21], the expert feedback indicated that a technique that addresses these issues would be extremely useful. We therefore discuss a new selection technique next that addresses the latter two mentioned limitations.

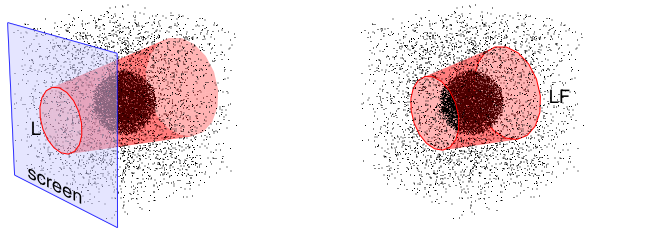
4 CloudLasso

Our second technique, CloudLasso, is based on the application of the Marching Cubes (MC) algorithm [20, 37] to the identification of dense parts of a particle dataset inside a user-drawn lasso; i. e., CloudLasso is a lasso-constrained Marching Cubes method. The MC method allows us to spatially select dense clusters within a lasso region individually even if these lie visually one behind another—without including the lower-density space in-between. To be able to apply the MC method, however, we need to first compute a continuous scalar field approximating the particle density. Using the scalar field we can then select the threshold for the intended selection. The CloudLasso algorithm, therefore, comprises the following three main steps:

1. Density estimation: The particle density is estimated inside a volume that contains all particles that project inside the lasso.
2. Volume selection: The subset of the volume where the density exceeds a threshold is computed using Marching Cubes.
3. Threshold tuning: Interactively adjusting the density threshold.

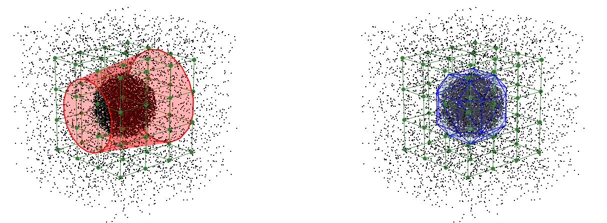
4.1 Density Estimation

In the first step of CloudLasso, we convert the user-drawn input lasso L (Fig. 4(a)) to a closed stroke, similar to TeddySelection, by connecting its start and end points, re-sampling it to remove noise, and addressing self-intersections. All subsequent computations and constructions in the algorithm are, unless otherwise noted, carried out in the view coordinate system. We, therefore, first transform all particle coordinates to



(a) Lasso is drawn on screen.

(b) First-level binning to obtain LF.



(c) Particle density is estimated on a grid containing LF.

(d) The volume with density above the threshold is computed using Marching Cubes.

Fig. 4. Main steps of the CloudLasso selection algorithm.

view coordinates by applying the graphics system's model-view transformation, while preserving volumes and densities. Then we perform the first-level binning stage as we have also used it for TeddySelection to obtain the lasso frustum (Fig. 4(b)).

Next, we determine the minimal rectangular box B such that it completely encloses LF and that its edges are parallel to the view coordinate axes. We construct a uniform rectangular grid G inside B such that the latter is split into 2^{18} cubes of equal volume ($64 \times 64 \times 64$; Fig. 4(c)). Then, we apply a kernel density estimation method in order to obtain a scalar density field from the particle data. Such methods effectively 'smear' each particle over a region and assign to each point in space a scalar value that approximates the local particle density. In our case, we compute a value for the scalar density field at each grid node using the modified Breiman kernel density estimation method (MBE) with a finite-support adaptive Epanechnikov kernel [9, 35]. Our reason for choosing this particular method is that it was shown by Ferdosi et al. [9] to be optimal with respect to speed and reliability when compared to the k-nearest neighbors, adaptive Gaussian kernel, and Delaunay tessellation field methods. Moreover, kernel methods (incl. Gaussian kernel methods) have two practical advantages over other density estimation methods. First, the Marching Cubes method that we use in later steps requires a grid-based density estimation, and kernel methods always compute this information. Second, kernel methods compute a continuous density field and, thus, are more suitable for usage in combination with Marching Cubes. The main benefit of the Epanechnikov kernel compared to Gaussian kernels, however, is its reduced computational effort: the number of grid points that have to be considered per particle is limited due to the Epanechnikov kernel's finite support.

We now give a brief description of the MBE method and we refer to Ferdosi et al.'s paper [9] for more details. First, for each direction $k = x, y, z$ we compute a smoothing length as

$$\ell_k = 2(P_k^{(80)} - P_k^{(20)}) / \log N,$$

where N is the number of particles in the rectangular box B that encloses LF and $P_j^{(q)}$ is the q-th percentile value for coordinate k. For each data particle i at position $\mathbf{r}^{(i)}$ and for each node n at position $\mathbf{r}^{(n)}$ we define the vector $\mathbf{r}^{(i;n)}$, the k-th component of which is given by

$$\mathbf{r}_k^{(i;n)} = (\mathbf{r}_k^{(i)} - \mathbf{r}_k^{(n)}) / \ell_k.$$

Then, we compute the *pilot density* $\rho_{\text{pilot}}(\mathbf{r}^{(n)})$ at the node n as

$$\rho_{\text{pilot}}(\mathbf{r}^{(n)}) = \frac{15}{8\pi N} \frac{1}{\ell_x \ell_y \ell_z} \sum_i 1 - \mathbf{r}^{(i;n)} \cdot \mathbf{r}^{(i;n)},$$

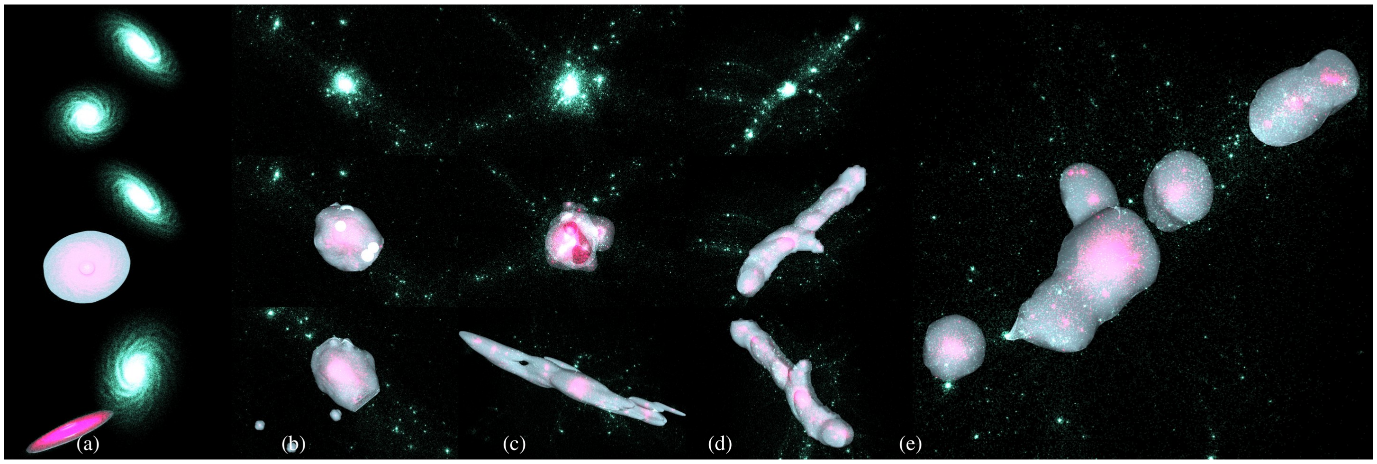


Fig. 5. Interactive CloudLasso selection: (a)–(d) different dataset configurations (top), CloudLasso selection (middle), and the selection viewed from another angle (bottom); (e) close-up of the selection in (d) with an interactively adjusted (higher) density threshold.

where only particles for which $\mathbf{r}^{(i;n)} \cdot \mathbf{r}^{(i;n)} \leq 1$ are considered in the sum. In other words, the only contribution to $\rho_{\text{pilot}}(\mathbf{r}^{(n)})$ comes from particles for which the node n is inside an ellipsoid with semi-axes ℓ_x, ℓ_y, ℓ_z centered at the particle.

We compute the pilot density $\rho_{\text{pilot}}(\mathbf{r}^{(i)})$ at the position of the i-th particle using multi-linear interpolation with respect to the nearby nodes. Next, we define the particle-specific smoothing lengths $\ell_k^{(i)}$ with $k = x, y, z$ for each particle i as

$$\ell_k^{(i)} = \min\{\ell_k(m/\rho_{\text{pilot}}(i))^{1/3}, 10\,s_k\},$$

where m is the arithmetic mean of $\rho_{\text{pilot}}(\mathbf{r}^{(i)})$ over all particles in the rectangular box B, and s_k is the distance between adjacent grid points in the k-th direction. Note that here we have introduced two modifications with respect to the standard method described by Ferdosi et al. [9]. First, we employ the arithmetic mean of the pilot densities instead of the geometric mean because the pilot density for a particle can often be zero, which would lead to a zero geometric mean. Second, we introduce a cut-off threshold for the value of $\ell_k^{(i)}$ because particles with a small pilot density define large corresponding ellipsoids with semi-axes $\ell_x^{(i)}$, $\ell_y^{(i)}$, $\ell_z^{(i)}$ and, thus, contribute to the final density of a large number of nodes. We found that due to this reason particles with large ellipsoids completely dominated the computational time and thus made the density estimation unsuitable for interactive applications. Now, we re-define the vectors $\mathbf{r}^{(i;n)}$ as

$$\mathbf{r}_k^{(i;n)} = (\mathbf{r}_k^{(i)} - \mathbf{r}_k^{(n)})/\ell_k^{(i)}.$$

Finally, we compute the density $\rho(\mathbf{r}^{(n)})$ at the node n as

$$\rho(\mathbf{r}^{(n)}) = \frac{15}{8\pi N}\sum_i \frac{1}{\ell_x^{(i)}\ell_y^{(i)}\ell_z^{(i)}}(1 - \mathbf{r}^{(i;n)} \cdot \mathbf{r}^{(i;n)}),$$

where only particles with $\mathbf{r}^{(i;n)} \cdot \mathbf{r}^{(i;n)} \leq 1$ are included in the sum.

4.2 Surface Extraction

To extract the selection shape surface based on the density estimation, we start by computing the average density for nodes of the grid G that lie inside LF and setting this average density as our initial *selection threshold* ρ_0. We could then naïvely apply the Marching Cubes algorithm to G to compute the selection region with $\rho \geq \rho_0$. However, we are actually interested in the region inside LF where $\rho \geq \rho_0$. Therefore, we first compute for each node n of G its projection n' on the screen and the distance $d(n')$ of n' from the lasso L. We define the corresponding *signed* distance $\delta(n)$ as $\delta(n) = d(n')$ if n' is inside L and as $\delta(n) = -d(n')$ if n' is outside L. This generalizes to arbitrary points: $\mathbf{r}$ in view coordinates projects on screen inside L if and only if $\delta(\mathbf{r}) \geq 0$. Hence, the point $\mathbf{r}$ is simultaneously inside the region $\rho \geq \rho_0$ and inside LF if

$$f(\mathbf{r}) = \min\{\rho(\mathbf{r}) - \rho_0, \delta(\mathbf{r})\} \geq 0.$$

Therefore, we apply the Marching Cubes method for the iso-surface $f(\mathbf{r}) = 0$ to obtain the bounding surface S of the required volume. The surface S might consist of more than one disconnected component. Furthermore, note that in order to ensure that the surface is closed we pad G with a layer of outer nodes where the value of f is defined to be `-DBL_MAX`. Also, after having constructed S, we mark particles inside this selection shape as being selected.

4.3 Threshold Adjustment

In the previous step we automatically set the density threshold to ρ_0. In our experience this setting already yields good selections but adjusting the threshold can improve the result. We thus provide means to interactively adjust the threshold at runtime in the range $[\rho_0/16, 16\rho_0]$ by mapping a linear parameter $s \in [-4, 4]$ to the threshold value using $\rho_s = 2^s\rho_0$. When s is adjusted, we thus recompute the scalar $f(\mathbf{r}) = \min\{\rho(\mathbf{r}) - \rho_s, \delta(\mathbf{r})\}$ for all grid nodes and obtain the iso-surface $f(\mathbf{r}) = 0$ using Marching Cubes. It is important to note that we only need to recompute $f(\mathbf{r})$ in this case because $\rho(\mathbf{r})$ and $\delta(\mathbf{r})$ remain constant, therefore adjusting the threshold is computationally much less expensive than the previous step and can be done interactively.

Furthermore, note that if ρ_s becomes smaller than the minimum density inside LF then the whole LF is selected. This means that by setting the threshold low enough the result of the CloudLasso method becomes identical to that of CylinderSelection, i. e., lasso selection using generalized cylinders. One could thus see CylinderSelection as a special case of CloudLasso selection.

4.4 Example Results

Fig. 5 shows a collection of results we generated by applying the CloudLasso to the same two datasets as discussed in Section 3.4 (colliding galaxies and cosmological N-body simulation). The top row shows the dataset before the selection, the middle row shows the result of selection applied for this view, and the bottom row shows a different view of the selection to illustrate how CloudLasso takes the 3D structure of the dataset into account. In Fig. 5(a) a compact cluster was selected with a circular lasso, however, after rotating the dataset (bottom) we see that the galaxy's compact disk was selected. In Fig. 5(b) an initially visually similar Fig. 5(b, top) compact cluster was selected, but this selection reflects the compact rounded shape of the cluster. Fig. 5(c) shows again a visually similar case where the cluster and consequently the selection also appear to be compact and rounded from

Table 1. CloudLasso performance. Times are in seconds.

Selection	Total Time	Density Estimation	Marching Cubes
Fig. 5(a)	6.38	4.89	0.46
Fig. 5(b)	2.13	0.83	0.39
Fig. 5(c)	4.14	2.87	0.39
Fig. 5(d)	4.50	1.95	0.39
Fig. 5(e)	0.30	0	0.28

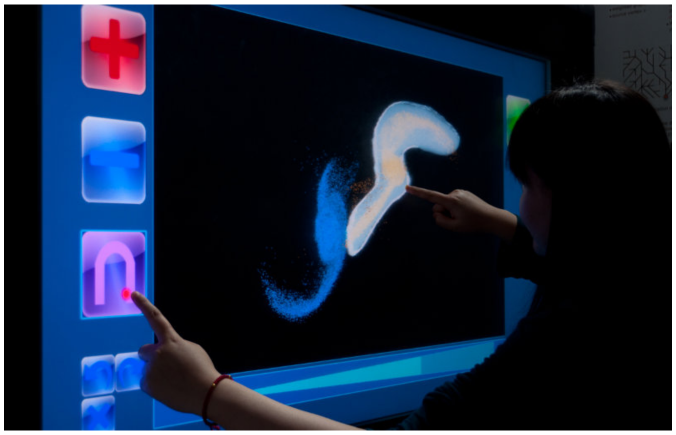


Fig. 6. A selection being performed using our direct-touch interface.

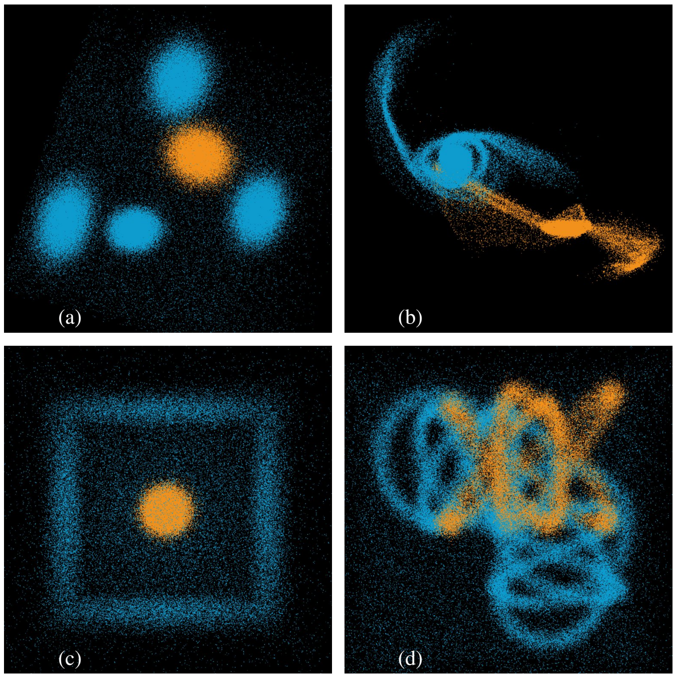


Fig. 7. Four tasks: (a) five simple particle clusters, (b) two galaxies, (c) cubic shell and central core, and (d) three intertwined figure-eight knots.

the initial view Fig. 5(c, top). However, after rotating the dataset (bottom) we can see that the cluster is elongated and that CloudLasso has taken this fact into account for the selection shape (including holes in the selection where appropriate). Fig. 5(d) shows a more complex cluster where the 3D structure of the 'arms' is captured. Fig. 5(e) shows a close-up view of Fig. 5(d) after changing the density threshold. As can be seen, only high density clusters remain selected.

4.5 Performance

The computationally most expensive part of CloudLasso is the density estimation. The density estimation processing time increases with the number of particles inside the rectangular box B. The processing time also heavily depends on the distribution of the particles within B because, as outlined in Section 4.1, particles in sparse areas can dominate the computation. After density estimation, the second-most expensive part is the selection geometry construction with Marching Cubes.

We have measured the performance of CloudLasso on a 3.16 GHz Intel® Core™ 2 Duo CPU for the selections shown in Fig. 5 and the results are summarized in Table 1. While the time required for density estimation is currently in the order of a few seconds, it can easily be parallelized using multiple parallel threads and cores (we discuss further possible improvements in Section 6.1). Moreover, changing the density threshold and recomputing the selection as in the selection of Fig. 5(e) takes only little time since the density is not recomputed. Thus, the technique is well suited for interactive applications.

5 User Study

To understand the user performance of and satisfaction with our selection technique we conducted a comparative quantitative evaluation (Fig. 6). As the baseline we selected Lucas and Bowman's Tablet Freehand Lasso (CylinderSelection) which, at this point, can be thought of as the standard selection method for point-based datasets. Due to the constraints of a controlled experiment we had to restrict ourselves to compare it to only one of our own techniques: CloudLasso or TeddySelection. We decided to use CloudLasso in the study because it is more flexible than TeddySelection due to the included threshold adjustment and the flexible construction of selection shapes in depth. Our comparison was based on both speed and accuracy as well as participants' qualitative feedback for four tasks. Because CloudLasso is capable of creating selection shapes based on the intended selection's spatial structure we hypothesized that CloudLasso would outperform CylinderSelection for all tasks based on speed. Due to the flexible threshold-based adjustment of the selection possible with CloudLasso we also hypothesized that it would be at least as accurate as CylinderSelection. We further hypothesized that the CloudLasso would score higher on questions related to how efficient participants perceived each method to be and would be generally preferred.

5.1 Study Description

Participants. Twelve people (8 male, 4 female) participated in the study. Eight participants were students from different disciplines and four non-students. All of them had at least a Bachelor's degree. Ten participants reported prior experience with 3D computer games with playing games up to three times per week, with four participants reporting at least weekly experience. Ages ranged from 24 to 33 years ($M = 28.75, SD = 3.3$). Ten participants reported to be right-handed, while the remaining 2 people reported to be ambidextrous.

Apparatus. The experiment was performed on a 52″ LCD screen with full HD resolution (1920×1080 pixels, $115.4\,\text{cm} \times 64.5\,\text{cm}$). The display was equipped with a DViT overlay [28] from Smart Technologies, capable of recognizing two independent inputs. The display was positioned so that its center was at a height of 1.47 m above the ground.

Tasks. Our study comprised four tasks. The dataset for each task (Fig. 7) contained target particles (orange), interfering particles (blue), and noise particles (light blue). Participants were always asked to select the orange target particles. To avoid measuring unnecessary time spent on navigation we only provided trackball rotation. A selection was activated through spring-loaded modes [2, 27] that allowed the participants to adjust selections while keeping a 'button' pressed (Fig. 6); otherwise the 2DOF input on the data was used for rotation. There were three possible selection modes corresponding to three Boolean operations (Fig. 6): union ($+$), intersection ($\cap$), and subtraction ($-$).

Before the actual experiment, four additional practice tasks were provided for participants to get accustomed to the selection techniques. The dataset for each of these tasks consisted of a low density, cubic volume of noise particles depicted in light blue and a higher density volume of orange target particles with a simple geometric shape: a sphere, a pyramid, a cylinder, and a torus.

The datasets used in the actual experiment are illustrated in Fig. 7. These tasks were designed to have different features and were ordered by difficulty. Fig. 7(a) shows five randomly placed compact clusters of particles with equal uniform densities inside a low density noise environment, with one being set as the target cluster. Fig. 7(b) is a simulation of two colliding galaxies which do not have uniform density. The participants were required to select one of the two galaxies. Fig. 7(c) shows a spherical, high-density core of target particles surrounded by a medium-density, cubic shell of interfering particles. Both structures are inside a low-density noise environment. Fig. 7(d) contains three intertwined particle 'strings,' each one shaped as a figure-eight knot, inside a low-density noise environment. Each 'string' had the same uniform density and participants were asked to select one of them.

At the start of each trial, the data space was oriented in a defined way (different orientations per trial). Participants were asked to finish the selection goal, i. e., to select the orange particles as quickly and accurately as possible. Participants thus needed to try their best to select, if possible, all target particles but to avoid selecting interfering or noise particles. They were asked in advance to find a balance between accuracy and speed and it was pointed out that a perfect selection was difficult or even impossible. We allowed participants to undo/redo the 5 most recent operations. Once participants felt that they accomplished the selection goal or that they were not able to improve the result, they could press a finish button to advance to the next trial. An additional density threshold slider was provided for CloudLasso trials (Fig. 6).

Design. We used a repeated-measures design with the within-subject independent variable *selection method* (CylinderSelection, CloudLasso). Per method, each participant performed 4 tasks and per task 4 trials. For each trial we chose a unique dataset starting orientation. Tasks were always performed in the same order and the presentation order of the two methods was counterbalanced among participants.

In summary, the design consisted of 12 participants × 2 methods × 4 tasks × 4 trials = 384 interactions in total. Participants moved from training to experiment after they reported to be able to perform a selection with the presented technique. After the experiment, participants were given a written questionnaire. They were asked to rate the usability of the techniques on a seven-step Likert scale with respect to ease of remembering, ease of use, efficiency, ease of drawing the lasso, and whether it was working as expected. Also, participants were asked to compare both techniques, comment on which technique they preferred and why, and whether the techniques allowed them to select the data as they desired. Finally, they filled in their demographic background information and were asked to provide additional verbal feedback on their experience of which the experimenters took notes.

5.2 Results

Completion times, errors, and selection volumes were recorded for the analysis. The density field for the CloudLasso method was precomputed to compare only the real interaction times. With this optimization, the processing times for computing the selected particles for CylinderSelection and CloudLasso were both approximately the same (in the order of 0.5s). Time and error in our experiment did not follow a normal distribution. We thus used the non-parametric Wilcoxon Signed Ranks test to analyze the data. The first block of trials was removed from the analysis due to a strong learning effect being present between blocks 1 and 2, across participants. Thus, we analyzed three trial blocks per task and method for each participant. In addition, trials were marked as outliers when each metric (time, error) was beyond two standard deviations from the mean for a given task and method per participant. Outliers were replaced with the closest value two standard deviations from the mean for each participant according to standard procedure. The datasets used in the different tasks differed in their characteristics, so we analyzed the results of each task independently.

We used two different metrics from information retrieval to calculate the error of our results [22]. To compute these scores we used the response number of true positives (TP, correctly selected particles), false positives (FP, incorrectly selected particles), false negatives (FN, missing particles that had to be selected), and true negatives (TN, correctly unselected particles). From these, the precision (P)—the fraction of target particles of all the retrieved particles—is calculated as: $P = TP/(TP+FP)$ and the recall (R)—the fraction of the target particles that were selected—is calculated as $R = TP/(TP+FN)$.

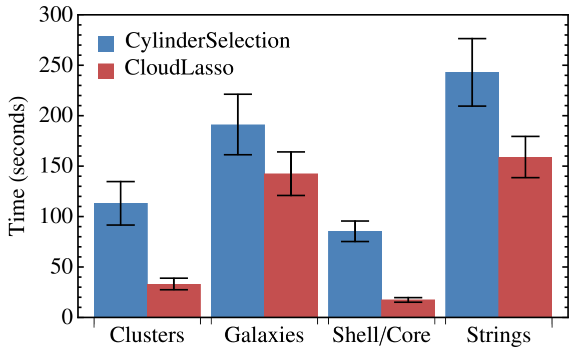


Fig. 8. Mean completion times for the user study tasks. Error bars represent 95% confidence intervals.

Table 2. Mean completion time (in seconds), mean F_1, mean MCC, and mean volume ratios for CloudLasso and CylinderSelection for the four user study tasks together with the corresponding significance scores.

		Clusters	Galaxies	Shell/Core	Strings
Mean Time (s)	CloudLasso	35.96	141.14	17.86	169.01
	CylinderSelection	118.94	189.62	88.01	248.68
	Z	3.06	1.41	3.06	2.28
	p	**<.01**	.16	**<.01**	**.023**
Mean F_1	CloudLasso	.9789	.9866	.9980	.7494
	CylinderSelection	.9759	.9855	.9960	.7303
	Z	2.67	0.71	3.06	2.04
	p	**<.01**	.48	**<.01**	**.041**
Mean MCC	CloudLasso	.9765	.9733	.9974	.6519
	CylinderSelection	.9731	.9712	.9948	.6305
	Z	2.75	0.78	3.06	1.89
	p	**<.01**	.43	**<.01**	.06
Mean V_S/V_R	CloudLasso	1.244	4.055	1.327	1.852
	CylinderSelection	1.303	5.855	1.360	2.691
	Z	0.94	1.49	1.57	2.98
	p	.347	.14	.875	**<.01**

The first metric, 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 $F_1 = P \cdot R/(P+R)$. A value of 1 for the F_1 score signifies a perfect result, while 0 is the worst possible result. Since the F_1 metric does not take the TN rate into account, we also used the Matthews correlation coefficient (MCC) as our second error metric which is often used in machine learning to assess the performance of a binary classifier. The MCC is calculated as:

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

Finally, we use a third error metric: the ratio of the selection volume and the target's real volume V_S/V_R, with values closer to 1 being better.

The task completion time analysis showed a significant effect with CloudLasso being significantly faster than CylinderSelection in all tasks except the task of the galaxies (Fig. 8). Moreover, with the same exception of the galaxies task and the string task for MCC, the F_1 score and the MCC score also showed statistically significant differences between the two methods, with CloudLasso being more accurate than CylinderSelection. Table 2 shows the details of the statistical analysis.

Five clusters. Tests showed that CloudLasso was significantly faster than CylinderSelection (36 s vs. 119 s). Moreover, the F_1 and MCC scores showed statistically significant differences between the two methods, with CloudLasso being more accurate. While CloudLasso created smaller volumes, the V_S/V_R differences were not significant.

In the post-session questionnaire the participants were asked to choose the method they preferred for each task. All participants chose CloudLasso over CylinderSelection on this dataset. They reported that CloudLasso was easy and convenient to use and that they could always get the right result (8×), was faster (8×), and used less steps (3×). In fact, many participants finished the task with only one CloudLasso selection step while, at the same time, getting more accurate results than with CylinderSelection. Furthermore, CloudLasso did not require participants to precisely draw a close lasso around the target particles.

Two colliding galaxies. While CloudLasso, on average, was faster than CylinderSelection (141 s compared to 190 s) and more accurate according to all error metrics, we did not observe a significant effect in any of these measurements. In the post-session questionnaire, six participants preferred CloudLasso, four favored CylinderSelection, while the remaining two felt both methods were more or less the same.

A reason for these results may lay in the fact that this selection task required participants to select particles with varying average density, the density of the galaxy's core being high while the arms have much lower density. Because participants were asked to select the whole galaxy, multi-step selections were necessary for both CloudLasso and CylinderSelection. We had intentionally designed this task with these characteristics because we hypothesized that it would be difficult for CloudLasso to select the whole galaxy in one step because it sets a density threshold based on the average density inside the lasso frustum. This threshold can only be varied inside a finite range because otherwise the interaction would become too imprecise. Since the average density in the galaxy is dominated by the very high density core, setting the threshold to its minimum possible value is still not enough to include the very low density arm edges. More than one selection step is thus necessary with CloudLasso to select all parts of the galaxy. CylinderSelection, in contrast, does not depend on the density distribution and always selects or deselects all particles in the lasso frustum. Therefore, CylinderSelection might have been more straightforward for some of the participants who all saw this dataset for the first time.

Cubic shell and core. CloudLasso was both faster than CylinderSelection (18 s vs. 88 s) and more accurate with respect to F_1 and MCC, with statistical significance. It also produced smaller volumes (smaller V_S/V_R) but this difference was not statistically significant. This clear advantage for CloudLasso is due to the fact that the noise and interfering particles with a low density did not create any problem for selecting the high-density core in the center. With CloudLasso, participants could finish the selection task in just one step, while CylinderSelection required several Boolean operations for satisfying results. We consequently also received positive feedback about using CloudLasso for this dataset in the post-session questionnaire: all participants preferred CloudLasso over CylinderSelection. The main reason for this choice as reported by participants was that CloudLasso was faster (7×), easier to use (4×), more precise (2×), and needed less steps (2×).

Three figure-eight knot shaped particle strings. In this last task, CloudLasso was also significantly faster than CylinderSelection (169 s as opposed to 249 s) and significantly more accurate with respect to F_1. It also showed a trend to be significantly more accurate on MCC and produced better (but not statistically significant) volume ratios.

We used this task at the end of the study because it is, by its very design, impossible to complete perfectly. Nevertheless, after the first CloudLasso selection step the selection already contained mostly target and interfering particles and not many noise particles. Thus, in subsequent steps, participants only needed to deselect the interfering particles. With CylinderSelection, in contrast, participants had to spend much time on 'carving' the particle strings. Many complained about fatigue using CylinderSelection for this task. In the questionnaire, eleven participants reported to prefer CloudLasso and one participant had no preference. All felt that both techniques were hard to use with this dataset, but also that the task was almost impossible with CylinderSelection while they could get better results with CloudLasso.

Overall Preferences. Participants were asked to compare both techniques in general. All of them named CloudLasso as their preferred technique. As their main reasons they reported CloudLasso to be easier to use (5×), faster (4×), more efficient (3×), and more precise (2×).

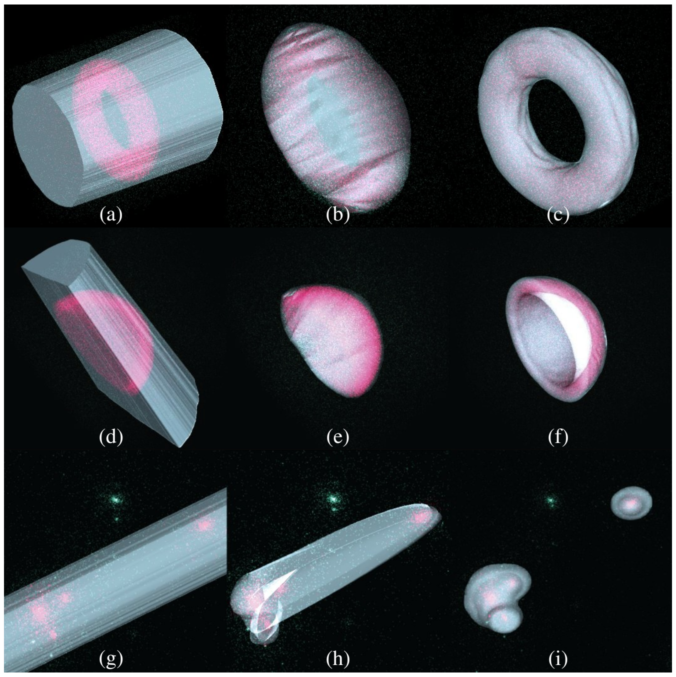


Fig. 9. Comparison of CylinderSelection (left column), TeddySelection (middle column), and CloudLasso (right column) in three different examples. From top to bottom: selection of a torus-shaped structure, a concave hemispherical shell (i. e., an empty, thickly-walled bowl), and two clusters from an astronomical N-body simulation. The three selections in each example are made from the same viewpoint with respect to the particles and the result is viewed from roughly the same direction.

6 Discussion

Based on the results from our study we now provide a comparing discussion between the three selection techniques mentioned in this paper, mention application domains other than the ones used so far, and discuss a further processing of the selections made with our techniques.

6.1 Comparison of the Selection Techniques

The TeddySelection and CloudLasso techniques introduced in this paper are two new *spatial* and *structure-aware* selection techniques. While we had previously already briefly touched upon their merits and limitations, we now provide an extended comparison between them and also include the CylinderSelection technique in this discussion.

Both CloudLasso and TeddySelection are based on the same principle: a user-specified 2D lasso defines a region in space and particles inside this region are selected based on the particle cloud structure. Nevertheless, this structure is taken into account in different ways by both methods so their results typically differ. In contrast to our structure-aware techniques, CylinderSelection starts with the same 2D lasso but selects everything in the lasso frustum. Therefore, CylinderSelection usually selects much more space than the desired target and several Boolean operations are needed to achieve the intended selection, something that we saw confirmed in our user study. This property of CylinderSelection is clearly visible in the examples shown in Fig. 9(a,d,g) where, in all cases, the selection geometry is very large after one step of the CylinderSelection method.

For roughly spherical shapes (not shown), both TeddySelection and CloudLasso give similar results and both methods are equally suitable because they deliver the intended selection in a single step. Nevertheless, even in such simple cases CloudLasso's selection geometry is often smoother than that of TeddySelection due to CloudLasso's density approximation. Furthermore, CloudLasso is generally more flexible than TeddySelection as the comparisons in Fig. 9 demonstrate.

In the selection of the torus and the hemispherical shell, CloudLasso perfectly fits the intended target (Fig. 9(c,f)). On the other hand, TeddySelection provides a closed geometry that approximates the intended

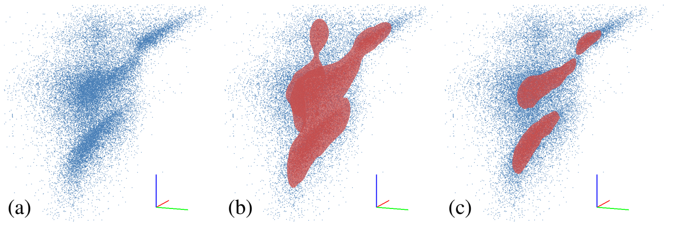


Fig. 10. CloudLasso being applied to a 3D scatter plot; data from a subset of the milliMillennium dataset [5] (showing the dimensions virial radius, R-I color, and low X-ray luminosity); (a) original 3D scatter plot, (b) initial CloudLasso selection, (c) after interactive threshold adjustment.

selections better than CylinderSelection but also selects the hole in the center of the torus (Fig. 9(b)) and the bowl's cavity (Fig. 9(e)). Finally, Fig. 9(h) shows that TeddySelection selects not only the two clusters that lie behind each other but also the space in-between them. CloudLasso addresses this issue and correctly selects only the clusters, resulting in several disconnected parts (Fig. 9(i)).

6.2 Limitations

CloudLasso's most important limitation is its performance. In our performance analysis reported in Section 4.5 we found that a selection typically required a few seconds. Results could vary, ranging from 2 s to more than 6 s in a way that can be unpredictable for the user. The main bottleneck is density estimation which can be addressed by parallelizing the respective computation as mentioned before. However, in the future we also want to investigate a GPU implementation of the kernel density estimation [3, 30] as well as of Marching Cubes. This would also address the second performance bottleneck of CloudLasso and take advantage of the fact that the data from the density estimation would already be available in GPU memory.

Both TeddySelection and CloudLasso are based on a number of parameter choices. For instance, TeddySelection uses two binning stages to stabilize the structure detection or clusters along the camera's z-direction. While CloudLasso also needs several parameters, most (e. g., the number of initial binning levels) do typically not have to be adjusted for different datasets. In fact, CloudLasso automatically adjusts to the density of an intended selection and provides a single parameter to adjust how closely a selection wraps around a cluster.

6.3 Other Applications and Possible Modifications

Our motivation for this work and the examples we have shown in the paper concern the selection of clusters in 3D astronomical particle datasets. Nevertheless, our methods can easily be applied for efficient selection in particle datasets that arise in other visualization domains as well as for any grid-based numerical data. For example, we recently applied CloudLasso as the selection technique in an application for the analysis of abstract, high-dimensional data. In the application the possible n-dimensional subspaces of a high-dimensional dataset are automatically ranked in terms of predefined quality measures, based on the number and prominence of clusters, to focus subsequent analysis on the highest-ranked subspaces. Three-dimensional subspaces are visualized directly and for larger-dimensional subspaces their respective three principal components are visualized. Based on such data or any other three-dimensional scatter plot, CloudLasso can be used for spatially selecting dense clusters as shown in Fig. 10. The selections can then be further analyzed by means of brushing and linking.

So far we have only considered the selection of unstructured 3D particle data. In many applications such as simulations of 3D flows and medical imaging, however, it is common to have scalar data defined on a 3D grid. We can easily modify CloudLasso for such datasets. CloudLasso would need to select those grid points that project inside the drawn lasso and where the value of the scalar quantity is above an automatically or user-defined threshold. In this case we can avoid, in fact, the density estimation step of CloudLasso since we already know the value of the scalar data on a 3D grid. On the other hand, in the visualization for such applications the region of interest may be enclosed by a semi-transparent surface that the user wants to ignore but that may otherwise interfere with the selection. In such cases CloudLasso should be combined by automated methods that can detect and resolve such ambiguities (e. g., [23, 33, 34]).

Finally, CloudLasso is suitable for particle selection based on scalar properties other than density. If a desired property is continuously defined in space we can evaluate it on a regular 3D grid. We can thus use Marching Cubes to select the spatial region where the target value is above a given threshold as described in the previous paragraph.

6.4 Further Selection Processing

While our selection methods allow users to intuitively perform spatial selections based on particle density, users sometimes need to modify their current selections if they did not achieve their desired precision in one step. This issue can easily be addressed by allowing Boolean operations with consecutively specified selection volumes. In this Boolean processing we can also combine selection methods, so that some of them are done with CloudLasso or TeddySelection, while others are done with CylinderSelection. In fact, adding new selections works best for CloudLasso or TeddySelection, while subtractions or set intersections work most intuitively with CylinderSelection.

Moreover, particle datasets including the cosmological N-body simulation data we used as our examples often consist of tens of terabytes of particle data, all of which rarely fits into the computer's main memory. To enable selections for these situations we can perform our selection specification based on a well-defined sample of the whole data and then store the selection shapes along with the employed Boolean operations in a selection pipeline. We can then apply this pipeline to all particles of the large dataset in an off-line process.

7 Conclusion

We presented CloudLasso and TeddySelection—two spatial, structure-aware selection techniques that can be applied to 3D particle cloud datasets in a visualization context. These techniques only require that users draw a lasso around the 2D projection of what they consider to be important by means of 2DOF input such as a mouse/pen or a finger on a direct-touch display (Fig. 6). With this input, our approaches allow users to select whole regions of space in a single step without having to rely on the selection of individual objects or having to refine a single selection repeatedly. In TeddySelection we married a technique from sketch-based modeling with generalized cone-based lasso selection and extended both to enable data selections that take 3D structural information into account. In CloudLasso we combined density estimation methods and a constrained version of Marching Cubes to uncover and select the most prominent 3D structures along the lasso direction.

Our techniques allow people to perform complex spatial selections in a wide range of applications and datasets, and the resulting selection shapes can be considered to be intuitive and efficient according to the feedback from the user study and from our collaborating experts. Both methods solve issues with current selection techniques employed in data analysis tools, allow people to explore sub-regions of particle datasets without requiring prior knowledge about the structures within this data, and thus open up new ways for data exploration.

CloudLasso and TeddySelection are the first steps in the development of structure-aware selection techniques for 3D particle datasets. We are certain that further work can improve these methods and generate interesting new ideas. In this paper we have established the relevance of such techniques for 3D particle selection and showed that they are more efficient than the traditional selection techniques. We, thus, contribute to the integration of interaction and visualization research [14] with the goal of improving scientific data analysis workflows.

Acknowledgments

From the University of Groningen, we would like to thank Amina Helmi for the cosmological simulation dataset from the Aquarius Project [29], Frans van Hoesel for the galaxy collision simulation (originally from http://www.galaxydynamics.org/), and Michael Wilkinson for his advice and valuable comments on the paper and for his density estimation code from DATAPLOT [35].

References

[1] F. Argelaguet and C. Andujar. Efficient 3D Pointing Selection in Cluttered Virtual Environments. *IEEE Computer Graphics and Applications*, 29:34–43, Nov./Dec. 2009. doi> 10.1109/MCG.2009.117

[2] W. Buxton. Chunking and Phrasing and the Design of Human-Computer Dialogues. In *Proc. IFIP World Computer Congress*, pp. 475–480, 1986.

[3] O. Daae Lampe and H. Hauser. Interactive Visualization of Streaming Data with Kernel Density Estimation. In *Proc. IEEE Pacific Vis*, pp. 171–178. IEEE Computer Society, Los Alamitos, 2011. doi> 10.1109/PACIFICVIS.2011.5742387

[4] G. de Haan, M. Koutek, and F. H. Post. IntenSelect: Using Dynamic Object Rating for Assisting 3D Object Selection. In *Proc. EGVE*, pp. 201–209. Eurographics Association, Goslar, Germany, 2005. doi> 10.2312/EGVE/IPT_EGVE2005/201-209

[5] G. De Lucia and J. Blaizot. The Hierarchical Formation of the Brightest Cluster Galaxies. *Monthly Notices of the Royal Astronomical Society*, 375(1):2–14, Feb. 2007. doi> 10.1111/j.1365-2966.2006.11287.x

[6] H. Dehmeshki and W. Stuerzlinger. Intelligent Mouse-Based Object Group Selection. In *Proc. Smart Graphics*, volume 5166 of *Lecture Notes in Computer Science*, pp. 33–44. Springer-Verlag, Berlin / Heidelberg, 2008. doi> 10.1007/978-3-540-85412-8_4

[7] H. Dehmeshki and W. Stuerzlinger. GPSel: A Gestural Perceptual-Based Path Selection Technique. In *Proc. Smart Graphics*, volume 5531 of *Lecture Notes in Computer Science*, pp. 243–252. Springer-Verlag, Berlin / Heidelberg, 2009. doi> 10.1007/978-3-642-02115-2_21

[8] N. Elmqvist, P. Dragicevic, and J.-D. Fekete. Rolling the Dice: Multidimensional Visual Exploration using Scatterplot Matrix Navigation. *IEEE Transactions on Visualization and Computer Graphics*, 14(6):1141–1148, 2008. doi> 10.1109/TVCG.2008.153

[9] B. J. Ferdosi, H. Buddelmeijer, S. C. Trager, M. H. F. Wilkinson, and J. B. T. M. Roerdink. Comparison of Density Estimation Methods for Astronomical Datasets. *Astronomy & Astrophysics*, 531:A114/1–16, July 2011. doi> 10.1051/0004-6361/201116878

[10] T. Grossman and R. Balakrishnan. The Design and Evaluation of Selection Techniques for 3D Volumetric Displays. In *Proc. UIST*, pp. 3–12. ACM, New York, 2006. doi> 10.1145/1166253.1166257

[11] T. Igarashi, S. Matsuoka, and H. Tanaka. Teddy: A Sketching Interface for 3D Freeform Design. In *Proc. SIGGRAPH*, pp. 409–416. ACM, New York, 1999. doi> 10.1145/311535.311602

[12] T. Isenberg and M. Hancock. *Gestures* vs. Postures: 'Gestural' Touch Interaction in 3D Environments. In *Proc. 3DCHI (ACM CHI 2012 Workshop on "The 3rd Dimension of CHI: Touching and Designing 3D User Interfaces")*, pp. 53–61, 2012.

[13] B. Jackson, D. Coffey, and D. F. Keefe. Force Brushes: Progressive Data-Driven Haptic Selection and Filtering for Multi-Variate Flow Visualizations. In *Short Paper Proc. EuroVis*, pp. 7–11. Eurographics Association, Goslar, Germany, 2012. doi> 10.2312/PE/EuroVisShort/EuroVisShort2012/007-011

[14] D. F. Keefe. Integrating Visualization and Interaction Research to Improve Scientific Workflows. *IEEE Computer Graphics and Applications*, 30(2):8–13, Mar./Apr. 2010. doi> 10.1109/MCG.2010.30

[15] D. F. Keefe, R. C. Zeleznik, and D. H. Laidlaw. Tech-note: Dynamic Dragging for Input of 3D Trajectories. In *Proc. 3DUI*, pp. 51–54. IEEE Computer Society, Los Alamitos, 2008. doi> 10.1109/3DUI.2008.4476591

[16] K. Kin, M. Agrawala, and T. DeRose. Determining the Benefits of Direct-Touch, Bimanual, and Multifinger Input on a Multitouch Workstation. In *Proc. Graphics Interface*, pp. 119–124. CIPS, Toronto, 2009.

[17] R. Kopper, F. Bacim, and D. A. Bowman. Rapid and Accurate 3D Selection by Progressive Refinement. In *Proc. 3DUI*, pp. 67–74. IEEE Computer Society, Los Alamitos, 2011. doi> 10.1109/3DUI.2011.5759219

[18] S. Lee, J. Seo, G. J. Kim, and C.-M. Park. Evaluation of Pointing Techniques for Ray Casting Selection in Virtual Environments. In *Proc. 3rd Int. SPIE Conf. on Virtual Reality and Its Application in Industry*, pp. 38–44. SPIE, Bellingham, WA, USA, 2003. doi> 10.1117/12.497665

[19] J. Liang and M. Green. JDCAD: A Highly Interactive 3D Modeling System. *Computers & Graphics*, 18(4):499–506, July/Aug. 1994. doi> 10.1016/0097-8493(94)90062-0

[20] W. E. Lorensen and H. E. Cline. Marching Cubes: A High Resolution 3D Surface Construction Algorithm. *ACM SIGGRAPH Computer Graphics*, 21(4):163–169, July 1987. doi> 10.1145/37402.37422

[21] J. F. Lucas and D. A. Bowman. Design and Evaluation of 3D Multiple Object Selection Techniques. Report, Virginia Polytechnic Institute and State University, USA, 2005.

[22] C. D. Manning, P. Raghavan, and H. Schütze. *Introduction to Information Retrieval*. Cambridge University Press, Cambridge, 2008.

[23] K. Mühler, C. Tietjen, F. Ritter, and B. Preim. The Medical Exploration Toolkit: An Efficient Support for Visual Computing in Surgical Planning and Training. *IEEE Transactions on Visualization and Computer Graphics*, 16(1):133–146, Jan./Feb. 2010. doi> 10.1109/TVCG.2009.58

[24] S. Owada, F. Nielsen, and T. Igarashi. Volume Catcher. In *Proc. I3D*, pp. 111–116. ACM, New York, 2005. doi> 10.1145/1053427.1053445

[25] J. S. Pierce, A. S. Forsberg, M. J. Conway, S. Hong, R. C. Zeleznik, and M. R. Mine. Image Plane Interaction Techniques in 3D Immersive Environments. In *Proc. I3D*, pp. 39–44. ACM, New York, 1997. doi> 10.1145/253284.253303

[26] I. Poupyrev, M. Billinghurst, S. Weghorst, and T. Ichikawa. The Go-Go Interaction Technique: Non-linear Mapping for Direct Manipulation in VR. In *Proc. UIST*, pp. 79–80. ACM, New York, 1996. doi> 10.1145/237091.237102

[27] A. J. Sellen, G. P. Kurtenbach, and W. A. S. Buxton. The Prevention of Mode Errors Through Sensory Feedback. *Human Computer Interaction*, 7:141–164, June 1992. doi> 10.1207/s15327051hci0702_1

[28] Smart Technologies Inc. Digital Vision Touch Technology. White paper, Feb. 2003.

[29] V. Springel, J. Wang, M. Vogelsberger, A. Ludlow, A. Jenkins, A. Helmi, J. F. Navarro, C. S. Frenk, and S. D. M. White. The Aquarius Project: The Subhalos of Galactic Halos. *Monthly Notices of the Royal Astronomical Society*, 391(4):1685–1711, Dec. 2008. doi> 10.1111/j.1365-2966.2008.14066.x

[30] B. V. Srinivasan, Q. Hu, and R. Duraiswami. GPUML: Graphical Processors for Speeding up Kernel Machines. SIAM Data Mining 2010 Workshop on High Performance Analytics – Algorithms, Implementations, and Applications, 2010.

[31] A. Steed and C. Parker. 3D Selection Strategies for Head Tracked and Non-Head Tracked Operation of Spatially Immersive Displays. In *Proc. 8th International Immersive Projection Technology Workshop*, pp. 163–170, 2004.

[32] A. Ulinski, C. Zanbaka, Z. Wartell, P. Goolkasian, and L. Hodges. Two Handed Selection Techniques for Volumetric Data. In *Proc. 3DUI*, pp. 107–114. IEEE Computer Society, Los Alamitos, 2007. doi> 10.1109/3DUI.2007.340782

[33] A. Wiebel, F. M. Vos, D. Foerster, and H.-C. Hege. WYSIWYP: What You See Is What You Pick. *IEEE Transactions on Visualization and Computer Graphics*, 18(12), Dec. 2012. In this issue.

[34] A. Wiebel, F. M. Vos, and H.-C. Hege. Perception-Oriented Picking of Structures in Direct Volumetric Renderings. Technical Report 11–45, ZIB, Berlin, Germany, 2011.

[35] M. H. F. Wilkinson and B. C. Meijer. DATAPLOT: A Graphical Display Package for Bacterial Morphometry and Fluorimetry Data. *Computer Methods and Programs Biomedicine*, 47(1):35–49, June 1995. doi> 10.1016/0169-2607(95)01628-7

[36] C. A. Wingrave, R. Tintner, B. N. Walker, D. A. Bowman, and L. F. Hodges. Exploring Individual Differences in Raybased Selection: Strategies and Traits. In *Proc. IEEE VR*, pp. 163–170. IEEE Computer Society, Los Alamitos, 2005. doi> 10.1109/VR.2005.1492770

[37] G. Wyvill, C. McPheeters, and B. Wyvill. Data Structure for *Soft* Objects. *The Visual Computer*, 2(4):227–234, Aug. 1986. doi> 10.1007/BF01900346

[38] L. Yu, P. Svetachov, P. Isenberg, M. H. Everts, and T. Isenberg. FI3D: Direct-Touch Interaction for the Exploration of 3D Scientific Visualization Spaces. *IEEE Transactions on Visualization and Computer Graphics*, 16(6):1613–1622, Nov./Dec. 2010. doi> 10.1109/TVCG.2010.157

[39] W. Zhou, S. Correia, and D. H. Laidlaw. Haptics-Assisted 3D Lasso Drawing for Tracts-of-interest Selection in DTI Visualization. IEEE Visualization 2008 Poster Compendium (Best Poster Nominee), 2008.