CAST: Effective and Efficient User Interaction for Context-Aware Selection in 3D Particle Clouds

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

Abstract—We present a family of three interactive Context-Aware Selection Techniques (CAST) for the analysis of large 3D particle datasets. For these datasets, spatial selection is an essential prerequisite to many other analysis tasks. Traditionally, such interactive target selection has been particularly challenging when the data subsets of interest were implicitly defined in the form of complicated structures of thousands of particles. Our new techniques SpaceCast, TraceCast, and PointCast improve usability and speed of spatial selection in point clouds through novel context-aware algorithms. They are able to infer a user’s subtle selection intention from gestural input, can deal with complex situations such as partially occluded point clusters or multiple cluster layers, and can all be fine-tuned after the selection interaction has been completed. Together, they provide an effective and efficient tool set for the fast exploratory analysis of large datasets. In addition to presenting Cast, we report on a formal user study that compares our new techniques not only to each other but also to existing state-of-the-art selection methods. Our results show that Cast family members are virtually always faster than existing methods without tradeoffs in accuracy. In addition, qualitative feedback shows that PointCast and TraceCast were strongly favored by our participants for intuitiveness and efficiency.

Index Terms—Selection, spatial selection, structure-aware selection, context-aware selection, exploratory data visualization and analysis, 3D interaction, user interaction

1 INTRODUCTION

Exploratory data visualization and analysis [48] is a fundamental pillar of many visualization systems. It provides domain experts with tools to study unknown datasets, an activity during which the data is examined more closely to discover interesting or unexpected patterns. One essential tool for exploratory visualization is the ability to select different subsets of the dataset [53], based on the current state of the exploration and/or one’s own intuitions. Selection is relatively easy for 2D datasets by means of picking, lasso-selection, or brushing. Three-dimensional datasets such as particle simulations, however, pose the much harder challenge of needing to specify one or more 3D volumes that enclose the intended subset of the data. Most existing selection mechanisms for 3D data [3] provide means to pick or ray-point at objects, but these fail if no explicit 3D shapes to pick exist—as it is the case for particle data.

Recently, a number of structure-aware selection techniques [39, 43, 56] were proposed that promise to deduce a person’s selection intention based on simple input, the characteristics of the dataset, and the current viewing conditions. All these techniques use a 2D lasso drawn on the projection of the 3D dataset and then derive a selection volume, typically by analyzing the particle density [43, 56] or the scalar properties of volume data [39]. While these approaches present a considerable improvement to a cylinder-based lasso selection in 3D space, all of them require the users of a visualization system to first find a good view for the selection interaction—a task that is often difficult or impossible in realistic scenarios with complex datasets. Moreover, the existing structure-aware approaches do not take the location or shape of the drawn lasso into account when determining which part of the enclosed data space may represent the intended selection region.

In order to address these issues, we present SpaceCast, TraceCast, and PointCast, a family of new interactive context-aware selection techniques (CAST) (see Fig. 1). SpaceCast (Fig. 1(a)) analyzes the parts of the clusters that project inside the 2D lasso, selecting only the part that has the best overlap with the input lasso. TraceCast (Fig. 1(b)) is inspired by stroke recognition techniques, such as those used for the input of Chinese characters. In contrast to SpaceCast, TraceCast does not use the lasso metaphor but interprets the path of the drawn selection stroke, selecting that candidate cluster whose 2D projection is best approximated by the drawn stroke. In particular, TraceCast does not restrict selection to within the drawn stroke. Both SpaceCast and TraceCast are useful to select clearly defined particle clusters, even if they are occluded (partially, or even completely) by others. PointCast (Fig. 1(c)), finally, uses the picking/ray pointing metaphor known from
we face another spatial dimension, but we also are typically limited to providing input in a 2D space (such as through mouse movements, pen input, or touch input). Unless one is only interested in a single location (e.g., voxel) [51] or surface patch [47] determined (i.e., picked) in a context-aware fashion, this task therefore requires the more difficult specification of a suitable 3D enclosing volume.

Oswada et al.'s [39] Volume Catcher, for example, solves this problem for unsegmented volume data by asking users to draw 2D strokes along the perceived borders of features for unsegmented volume data, and then deduces an appropriate segmentation of the dataset, i.e., a selection. In an approach designed for line data, Aker's [1] CINCH provides a marking interface for structure-aware 3D neurologic pathway selection. Yu et al.'s [56] TeddySelection and CloudLasso allow users to enclose spatial regions with a lasso drawn on the 2D projection and derive an appropriate selection subspace for particle data. They demonstrate, in particular, that their CloudLasso is able to handle complex scenarios in less time than a simple cylinder-based selection. Shan et al. [43] then discuss an extension to CloudLasso, in that they analyze the different clusters that are generated by the CloudLasso technique and only select the one with the largest 2D projection.

These context/structure-aware selections thus all specify what is interesting by deriving a sub-space, doing that in a manner that depends on the data, the view, and other parameters [38]. While Wills [53] derives his taxonomy of selection techniques for visual encodings of abstract data on the 2D plane, his distinctions of selections with respect to space differentiation and data dependency also apply to our three-dimensional datasets as he also discusses selection as a way to spatially specify what to investigate further. The Cast family of selection techniques that we discuss in this paper are, hence, data-dependent tools that provide intuitive means to specify selection through a lasso, through a drawn stroke, or through a specified point. We thus address the challenges of the existing structure-aware selection techniques [39, 43, 56] which are only able to make global decisions for the entire dataset (or the lasso-enclosed part), while SpaceCast, TraceCast, and PointCast derive criteria for more local, precise, and intuitive selections by the shape and location of the drawn primitive.

### 3 Description of the Selection Techniques

**To achieve these goals**, the Cast family members have to derive the selection intention of the user from the type and shape of the provided input—in a way that goes beyond a simple two-dimensional cut-off mask as done by previous structure-aware approaches [43, 56]. We thus employ three dedicated heuristics that use different ways for interactively constraining the selection volume. These heuristics do not directly use the particle positions but rather a continuous density field \( \rho(\mathbf{r}) \) that represents the particle density at a point \( \mathbf{r} \) in space. 1 We first compute the density field at the nodes \( n(\mathbf{r}) \) of a regular 3D-grid that covers the dataset. We then derive the value \( \rho(\mathbf{r}) \) of the density field at any other point \( \mathbf{r} \) in space through linear interpolation from the values of \( \rho \) at the grid nodes closest to \( \mathbf{r} \).

*1This aspect also makes it possible to apply the techniques to volumetric data which samples a chosen scalar field. This scalar field does not have to be a density, but any visually salient aspect of the data when visualized.*
where the sum runs over the set of grid-nodes \( r^{(k)} \) inside \( F \) and \( N_F \) is the number of such nodes. We then choose the initial density threshold as \( \rho_0 = 0.2 \rho_L \). Using the Marching Cubes algorithm \([36, 55]\), we identify the volume \( V \) (inside the frustum \( F \)) where the density \( \rho \) is above the threshold \( \rho_0 \). More precisely, we define the scalar quantity

\[
f(r) = \min \left\{ \frac{\rho(r) - \rho_0}{\sigma \rho}, \frac{\delta(r)}{D_L} \right\},
\]

where \( \delta(r) \) is the signed distance from the lasso to the projection of \( r \) on the screen, that is, if we denote by \( d(r) \geq 0 \) the distance of \( r \) from \( L \) then \( \delta(r) = -d(r) \) otherwise. To ensure that the physically disparate compared quantities have the same order of magnitude we scale distances by the lasso diameter \( D_L \) and densities by the density standard deviation \( \sigma \rho \). Volume \( V \) then results as the set for which \( f(r) \geq 0 \).

In the previous step we automatically set the density threshold to \( \rho_0 \). To facilitate finer control of the selection volume we 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 recompute the scalar \( f(r) \) for all grid nodes with \( \rho_0 \) replaced by \( \rho_s \) and obtain the iso-surface \( f(r) = 0 \) using Marching Cubes. It is important to note that we only need to recompute \( f(r) \) in this case because \( \rho \) and \( \delta(r) \) remain constant, therefore adjusting the threshold is computationally much less expensive than the previous step and can be done interactively.

If \( V \) consists of a single connected component then this is all we need to do for selecting the volume and all particles inside \( V \) are marked as selected—SpaceCast would then work similar to CloudLasso. In general, however, \( V \) consists of more than one disjoint volumes \( V_1, \ldots, V_K \). We identify these volumes using the algorithm described in Sect. 3.5.

In contrast to CloudLasso and WYSIWYG selection, however, we now use the shape of the drawn lasso to select one of the disjoint volumes. In particular, we project the volume \( V_k \) for each \( k = 1, \ldots, K \), to an area \( S_k \) on the screen. We then compare each area \( S_k \) to the area \( S_0 \), enclosed by the lasso \( L \) and identify the area \( S_{k_0} \), with \( k_0 \in \{1, \ldots, K\} \), that has the maximal overlap with \( S_0 \). We then determine the selection volume to be \( V_{k_0} \), except in the following case. We choose the selection volume to be \( V_{k_0} \) if there is a volume \( V_{k_0} \) such that \( S_{k_0} > 0.8 \delta_{S_0} \) and \( S_{k_0} \) is closer to the eye compared to \( V_{k_0} \), for the following reason. Let us suppose that the user tries to select a cluster \( V_k \) in front of a more extended cluster \( V_B \) as shown in Fig. 2(a). If the user then draws a lasso \( L \) encircling \( S_k \) we generally get \( S_k < S_0 \) (because \( L \) will not be perfectly wrapped around \( S_k \)) but \( S_B = S_0 \). Choosing the area with the largest overlap would then give \( V_B \) although the intention was \( V_k \). If more than one volume \( V_k \) exists that satisfies the condition \( S_{k_0} > 0.8 \delta_{S_0} \), we choose the one closest to the eye.

We note here that if the structure of the dataset and the density threshold are such that \( V \) consists of a single connected component then SpaceCast and CloudLasso give the same result. Furthermore, if the density threshold is set to a very low value then SpaceCast gives the same result as a simple cylinder-based selection.

### 3.2 TraceCast

A potential problem arises for SpaceCast in certain cases when a selection in complex dataset is required: SpaceCast requires the users to always outline the desired clusters fairly accurately and to draw closed (or almost closed) lasso loops. To ease such interaction we thus present TraceCast which, despite being only subtly different in its definition, can thus lead to significantly different interaction schemes as well as different selected sub-spaces.

With TraceCast the user draws a lasso-like stroke \( L \) around the particles to be selected as before. If the stroke is not closed, we close it as described in Sect. 3.1. However, unlike SpaceCast, we do not only consider the resulting frustum \( F \) but also take regions outside \( F \) into account for all subsequent operations. We first compute an approximate average density \( \rho_F \) inside \( F \), as for SpaceCast. We then choose the initial density threshold as \( \rho_0 = 0.2 \rho_F \). Using the Marching Cubes algorithm, we identify the volume \( V \) in the whole space where the density \( \rho \) is above the threshold \( \rho_0 \). In particular, the volume \( V \) is where \( f(r) \geq 0 \) but for this case we define the function \( f \) as

\[
f(r) = \rho(r) - \rho_0.
\]

whose threshold \( \rho_0 \) can interactively be modified as in SpaceCast.

We then consider the disjoint volumes \( V_{k_1}, \ldots, V_{k_T} \) that form \( V \) and use the lasso shape to select one of these disjoint volumes. As in SpaceCast, we project the volume \( V_k \) for each \( k = 1, \ldots, K \) onto an area \( S_k \) on the screen. Note that in TraceCast the volume \( V_k \) can project (completely or partly) outside the area \( S_k \) that is enclosed by the lasso-like stroke \( L \). We discard \( V_k \) if it projects completely outside \( S_k \) (that is, \( S_k \cap S_k = \emptyset \)). Otherwise, we compare the area \( S_k \) to the area \( S_k \) enclosed by the lasso \( L \) and assign it an overlap measure \( m_k \), defined as

\[
m_k = \text{area}(S_k \cap S_k) - \text{area}(S_k \triangle S_k) = 2 \text{area}(S_k \cap S_k) - \text{area}(S_k \cup S_k),
\]

where \( \triangle \) represents the symmetric difference of two sets as shown in Fig. 2(b). We thus identify the area \( S_{k_0} \), with \( k_0 \in \{1, \ldots, K\} \), that has the maximal overlap measure \( m_{k_0} \) and choose the selection volume to be the corresponding volume \( V_{k_0} \). Our approach for TraceCast thus results in selecting that subset of space generally in the area of the lasso-like stroke that best approximates its shape.

Finally, we note that although TraceCast depends on the user-drawn lasso, the latter can be much less precise than in SpaceCast. This is clearly shown in Fig. 1(b) where a galaxy is selected using a “swirling” stroke. Using SpaceCast in the same dataset, we would need to draw a lasso that surrounds the target galaxy since SpaceCast does not consider any particle for the selection that falls outside the lasso.

### 3.3 PointCast

While SpaceCast and TraceCast are well-suited for larger clusters with often complicated structure, in practical applications we also frequently need to be able to quickly select many smaller structures—in a way like using ray-pointing to select dedicated objects in a 3D scene. We thus also introduce PointCast, a technique based on the concept of treating whole clusters as individual objects and using the simplest interaction method—a single click or touch action—to mark such a cluster.

With PointCast we thus start by projecting a ray into the dataset, guided by the marked point that is located on the interior of the target’s projection to the screen. To be able to infer the intended selection volume, we start by identifying a density threshold as follows. We first sample the particle density \( \rho \) at equally spaced points along the ray, from the closest point \( r_0 \) to the farthest point \( r_1 \) that still lies inside the bounds of the dataset. Let

\[
\rho(s) = \rho(r_0 + s(r_1 - r_0)), \quad 0 \leq s \leq 1,
\]

represent the particle density along the ray. By sampling \( \rho(s) \) we then obtain the sequence \( \rho_i = \rho(s_i) \) with \( s_i = i/N_s \), for \( i = 0, 1, \ldots, N_s \) where \( N_s + 1 \) is the number of sample points. We identify clusters along this ray by locating the intervals along the ray where \( \rho(r) \geq 0.1 \max \{\rho_i\} \). This produces a sequence of intervals \( I_1 = [s_{i_1}, s_{i_2}] \), \( I_2 = [s_{i_2}, s_{i_3}] \), etc., each of these intervals corresponding to a cluster.
A common aspect of all methods in the Cast family is that they replace the particle position information by a continuous density function. We use the same continuous density, based on a standard Kernel Density Estimation (KDE) technique, as done by Yu et al. [56]. To aid comprehension, we briefly summarize its construction next. KDE methods obtain a smooth density function by ‘smearing’ each particle over a larger volume. We consider a rectangular box $B$ that covers the dataset (or the region of interest if the latter is restricted) and a uniform grid with nodes at positions $r^{(n)}$ where the superscript $n$ identifies the node. For the estimation of the scalar density field we use the modified Breiman kernel density estimation method (MBE) with a finite-support adaptive Epanechnikov kernel [20, 52]. This works as follows. First, for each direction $k = x, y, z$ we define the smoothing length

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

where $N$ is the particle count in $B$ and $P_k^{(q)}$ is coordinate $k$’s q-th percentile value. Then, we derive the pilot density $\rho_{\text{pilot}}(r^{(n)})$ at node $n$ as

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

where $r^{(i,n)}$ is the vector with $k$-th component

$$r_k^{(i,n)} = (r_k^{(i)} - r_k^{(n)}) / \ell_k,$$

$||r^{(i,n)}||$ is its length, and $E(x) = 1 - x^2$ for $|x| \leq 1$ while $E(x) = 0$ for $|x| \geq 1$. The specific form of the Epanechnikov kernel, $E(x)$, implies that only particles for which the node $n$ is inside an ellipsoid with semi-axes $\ell_x, \ell_y, \ell_z$ centered at the particle contribute to $\rho_{\text{pilot}}(r^{(n)})$.

The pilot density $\rho_{\text{pilot}}(r^{(i)})$ at the position of the $i$-th particle is then given by multi-linear interpolation with respect to the nearby nodes. For each particle $i$ and with $k = x, y, z$ we then define the smoothing lengths $\ell_k$ which are unique to each particle as

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

where $m$ is the arithmetic mean of $\rho_{\text{pilot}}(r^{(i)})$ over all particles in $B$, and $s_k$ is the distance between adjacent grid points in the $k$-th direction. Now, we can re-define the vectors $r^{(i,n)}$ as

$$r_k^{(i,n)} = (r_k^{(i)} - r_k^{(n)}) / \ell_k^{(i)},$$

The density $\rho(r^{(n)})$ at the node $n$ is then

$$\rho(r^{(n)}) = \frac{15}{8\pi N} \frac{1}{\ell_x \ell_y \ell_z} \sum_i E(||r^{(i,n)}||).$$

Finally, we compute the density $\rho(r)$ at an arbitrary position using multi-linear interpolation with respect to the nearby nodes.

### 3.5 Identification of Disjoint Volumes

A common step in the Cast methods is to identify the disjoint volumes that constitute the volume as computed by the Marching Cubes isosurface extraction. We do this using an algorithm similar to that of Shan et al.’s [43] WYSIYG selection. In contrast to Shan et al.’s approach, however, we identify disjoint volumes by first identifying the connected components of the Marching Cubes isosurface. This leads to a more robust detection that does not have any problems with nearby but disjoint selection volumes that could be wrongly identified as part of the same component. Specifically, we first use a disjoint-set data structure to compute equivalence classes for the faces that make up the Marching Cube isosurface, where we consider two faces to be equivalent if they share an edge. Each such equivalence class represents a connected component of the isosurface. This task is significantly simplified by the fact that we keep track of the relations between vertexes, edges, and faces of the isosurface. We first assign an index $k \in \{1, \ldots, K\}$ to each equivalence class, and thus to each connected isosurface component. We then consider the grid-nodes $r^{(n)}$. To each of these grid-nodes that lies inside the isosurface (i.e., for which $f(r^{(n)}) \geq 0$) we assign the index of the connected isosurface component using a flood-filling algorithm. We start with a grid-cell that contains a face with a given index $k$. Then we find a node in this cell that is contained within the isosurface component with index $k$. We assign the index $k$ to the found node and then we proceed to neighboring nodes assigning the same index $k$ if they can be reached without crossing the isosurface. We thus identify all grid-nodes with index $l$. By repeating this process for all indices, we finally mark each node with an index $k \in \{1, \ldots, K\}$ if it lies inside the Marching Cubes isosurface. For completeness, we assign the index $k = 0$ to nodes outside the isosurface.
16 participants reported prior experience with 3D computer games and noise particles. These datasets were designed to explain the differences between the selection techniques. The datasets contained target particles, interfering particles, and noise. Participants were always asked to select all or different numbers of particles with equal uniform densities inside of a low density noise environment. Each ring had the same uniform density with target and interfering particles partially surrounded by a semispherical shell of interfering particles. Both structures had the same uniform density.

Clusters: The clusters dataset in Fig. 4(a) contained 15 compact clusters of particles with equal uniform densities inside a low density noise environment. The target cluster was located in the center so that no viewing direction allowed a clear view to the whole target. Shell: The shell datasets in Fig. 4(b) included a half-ball of target particles partially surrounded by a semispherical shell of interfering particles. Both structures had the same uniform density. Rings: The rings dataset (Fig. 4(c)) contained a circular 'ring' intertwined with a 'figure-8' (a 'double ring'), inside a low-density noise environment. Each ring had the same uniform density and participants were asked to select the circular one. In this dataset the structure of the target intertwined with other structures, making selection challenging.

Simulation: Fig. 4(d) shows a cosmological N-Body simulation dataset with a high density core in the center and many small and high density clusters all around the space. Participants were required to select several small clusters. This real-world dataset was chosen to represent a realistic selection scenario in astronomy. Considering that the high density core is relatively easy to select with any technique, we chose small clusters as the target to see how our selection strategies worked when multiple selections had to be performed.

Task and Procedure Participants were always asked to select all orange target particles and avoid selecting interfering or noise particles. Before the actual experiment, participants practiced with three additional training datasets to get accustomed to the selection techniques. The practice datasets were chosen to explain the differences between techniques and the interaction with the experimental software. These practice datasets included: 1) a simulation of two colliding galaxies (for practicing trackball rotation), 2) five randomly placed compact clusters of particles with equal uniform densities inside of a low-density noise environment, with one being set as the target cluster (for explaining how the method works and the differences compared with other methods), and 3) a higher-density volume of orange target particles with two simple geometric shapes: a pyramid and a torus (for explaining the method for different selection target shapes). In the training trials, we gave the participants as much time as they wanted, before proceeding to the actual experiment. Training took on average 14 mins for CylinderSelection, 8 mins for CloudLasso, 7 mins for SpaceCast, and 6 mins each for TraceCast and PointCast.

We asked participants to perform their selections as quickly and accurately as possible but we did not tell them when they had accomplished the selection goal. For each trial-dataset combination we chose a unique dataset starting orientation, that changed between trials but was kept the same for different participants. We also counter-balanced the order of selection methods presented to the participants such that each possible starting-ending pair of methods was used exactly once and that no participant saw the same progression order. To avoid measuring unnecessary time spent on navigation, we only provided trackball rotation. A selection was activated after participants pressed a 'selection button' to change to selection mode, otherwise the 2DOF input on the data was used for rotation. We provided three possible selection modes, corresponding to three Boolean operations: union (+), intersection (∩), and subtraction (−). We also provided a slider (Fig. 14) through which participants could adjust the automatically determined density threshold even after rotating the view. As Yu et al. [56] had previously discussed, subtraction and intersection are best done using CylinderSelection and not using structure-aware techniques. We thus implemented subtraction and set intersection as CylinderSelection in all trials.

We allowed participants to undo/redo the five most recent operations. Once they 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.

Design We used a repeated-measures design with the within-subject independent variables selection technique (CylinderSelection, CloudLasso, SpaceCast, TraceCast, and PointCast) and dataset (Clusters, Shell, Rings, Simulation). Each technique was used for each dataset in 3 repetitions. In summary, the design consisted of 20 participants × 5 methods × 4 datasets × 3 repetitions = 1200 trials in total.

Measures and Analysis After an inspection of timing results, we removed the first repetition of each dataset × technique combination in order to reduce the influence of learning effects. This left us with 80 trials per participant (800 trials in total). Traditionally, such study data would have been analyzed using null-hypothesis significance testing (NHST). The use of NHST, however, has been increasingly criticized as a tool to analyze user study experiments [12, 17, 18], also discussed in the context of visualization [16, 27]. Consistent with recent APA recommendations [50], we thus report results using estimation techniques with effect sizes and confidence intervals rather than p-value statistics.

To compare the five techniques, we measured task completion time and computed two different accuracy scores, the F1 and MCC scores, similar to Yu et al.’s work [56]. Both accuracy scores are based on the identification of true positives (TP, number of correctly selected particles), false positives (FP, number of incorrectly selected particles), and false negatives (FN, number of target particles that were not selected). The F1 score is a weighted average of precision $P = TP / (TP + FP)$ and recall $R = TP / (TP + FN)$ and is often used in information retrieval to assess query classification. F1 is calculated as: $F1 = 2 \cdot (P \cdot R) / (P + R)$. The second accuracy measure, the Matthews correlation coefficient (MCC), additionally uses the number of true negative particles (TN, non-target particles that were not selected) to calculate an accuracy score. It is commonly used in machine learning for the assessment of binary classifiers. MCC is calculated as:

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

Both scores normalize accuracy: for F1, 1 indicates a perfect performance and 0 represents the worst possible result, while the corresponding values for MCC are 1 and -1. For accuracy scores we computed means and 95% bootstrap confidence intervals (all $n = 20$) [32].

We analyzed completion time data using exact confidence intervals on log-transformed data (all $n = 20$) making outlier removal unneces-
Table 1. Mean task completion times, accuracy scores, and their 95% confidence intervals per technique.

<table>
<thead>
<tr>
<th>Technique</th>
<th>Time (s)</th>
<th>CI 95%</th>
<th>F1</th>
<th>CI 95%</th>
<th>MCC</th>
<th>CI 95%</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cylinder</td>
<td>88</td>
<td>[73,106]</td>
<td>.96</td>
<td>[95,.97]</td>
<td>.96</td>
<td>[95,.96]</td>
</tr>
<tr>
<td>CloudLasso</td>
<td>43</td>
<td>[38,48]</td>
<td>.96</td>
<td>[96,.97]</td>
<td>.96</td>
<td>[95,.97]</td>
</tr>
<tr>
<td>SpaceCast</td>
<td>23</td>
<td>[20,27]</td>
<td>.97</td>
<td>[94,.97]</td>
<td>.96</td>
<td>[93,.97]</td>
</tr>
<tr>
<td>PointCast</td>
<td>16</td>
<td>[13,19]</td>
<td>.97</td>
<td>[93,.98]</td>
<td>.97</td>
<td>[93,.98]</td>
</tr>
</tbody>
</table>

Fig. 5. Mean completion time (in seconds) across all participants for each selection technique. Error bars show 95% confidence intervals. Means are geometric means.

<table>
<thead>
<tr>
<th>Ratio between completion times</th>
<th>0</th>
<th>1</th>
<th>1.5</th>
<th>2</th>
<th>2.5</th>
</tr>
</thead>
<tbody>
<tr>
<td>TraceCast/PointCast</td>
<td>0</td>
<td>1</td>
<td>1.5</td>
<td>2</td>
<td>2.5</td>
</tr>
<tr>
<td>SpaceCast/TraceCast</td>
<td>no effect</td>
<td>0</td>
<td>1</td>
<td>1.5</td>
<td>2</td>
</tr>
<tr>
<td>SpaceCast/PointCast</td>
<td>0</td>
<td>1</td>
<td>1.5</td>
<td>2</td>
<td>2.5</td>
</tr>
</tbody>
</table>

Fig. 6. Ratios between mean completion times for the Cast selection techniques. Error bars show 95% confidence intervals.

4.2 Results

Here we report results of both the quantitative analysis as described above, the results of the post-session questionnaire, as well as a few observations from watching participants complete the study.

4.2.1 Overall Results: Time, Accuracy, and Preference

Fig. 5 shows the mean completion times and 95% confidence intervals for each technique. We can see that CylinderSelection and CloudLasso were much slower than the three Cast techniques. CylinderSelection was also much slower than CloudLasso, which is consistent with the results of Yu et al.’s [56] experiment. Exact average task completion times as well as confidence interval ranges are listed in Table 1. Given the high completion time difference for CylinderSelection, Fig. 5 is less clear as to the difference amongst the Cast techniques. Therefore, we provide pairwise ratios of completion times of the Cast techniques in Fig. 6. Here we can see that SpaceCast took about 1.3–2.2 times longer than PointCast on average and 1.1–1.8 times longer than TraceCast. Participants performed only about 1–1.4 times slower with TraceCast than PointCast. Thus, overall we have good evidence that both PointCast and TraceCast outperform SpaceCast, and some indication that PointCast may outperform TraceCast. At any rate, the differences among Cast methods are marginal compared to the differences between each Cast method and CloudLasso or CylinderSelection.

At the beginning of the study (before the trials began), we had asked participants how they would intuitively approach a selection of 3D particle data. More than 80% of participants had then reported that drawing a lasso would feel most natural. In the post-session questionnaire, however, 13 participants reported to favor the point-based method PointCast. Out of the remaining participants, six preferred the lasso-based method TraceCast and one participant preferred both PointCast and TraceCast over the other techniques. This is interesting also in the context of the actual results. The analysis of accuracy scores (refer to Fig. 5 for exact numbers) for the five techniques shows that participants were very accurate with all techniques. All F1 and MCC scores are ≥ .96 and, therefore, close to a perfect result (1).

4.2.2 Analysis Per Dataset

Overall, participants completed all trials with similar completion times in just under 30s on average for the three synthetic selection problems Rings, Shell, and Cluster (see Fig. 7). Trials of the simulation dataset, on the other hand, were completed in over one minute on average, so the selection problem was more complex. Next we discuss the results for the three synthetic datasets and the real-world dataset separately.

Synthetic Datasets As can be seen in Fig. 9(a-c), the ranking of techniques according to task completion times for each of the datasets is similar to the overall ranking result. The three Cast techniques outperform CloudLasso and Cylinder for each of these three datasets. Amongst the Cast techniques, SpaceCast was generally the slowest with TraceCast and PointCast showing similar averages. Table 2 shows that the accuracy scores were also similarly high for each of the selection problems, when examined by technique.

In the post-session questionnaire we had asked the participants to choose the method they preferred for each dataset. PointCast was the preferred technique for all datasets, with exact numbers reported in Fig. 8. Participants preferred PointCast for the clusters datasets because it was fast (6x), accurate (9x), and because it was easier to select occluded particles (10x). Similar reasons were given also for the other datasets, with slightly different frequencies. TraceCast was the second most preferred technique across all datasets. Reasons for choosing TraceCast were that it offered a feeling of control and more accuracy, in particular for the Shell dataset where it was possible for PointCast to select unwanted clusters given certain viewing directions.

N-body mass simulation In this dataset we tested a multi-step selection process by asking participants to select several small clusters that were located in different places in space. Participants thus needed to rotate the whole space in order to find the targets. This explains the increased task completion time for this dataset (see Fig. 7) when compared to the three synthetic datasets. Interestingly the ranking of techniques according to task completion time changed for this dataset. Cylinder was still by far the worst technique for this task. The performance of CloudLasso, however, was much closer to the Cast techniques than for the other datasets (see Fig. 9(d)). In Fig. 10 we can see evidence that participants performed almost equally well with CloudLasso and SpaceCast as well as with PointCast. The highest ratio difference between techniques was found between TraceCast and SpaceCast, with TraceCast performing 1.2x slower than SpaceCast. In summary, this comparison provides us with some evidence that SpaceCast performed...
were almost equally fast with CloudLasso. Similar to the other datasets, however, accuracy was high for all techniques as we show in Table 2.

Table 2. Mean task completion times, accuracy scores, and their 95% confidence intervals per technique and dataset.

<table>
<thead>
<tr>
<th>Technique</th>
<th>Time CI</th>
<th>F1 CI</th>
<th>MCC CI</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Clusters Dataset</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Cylinder</td>
<td>65s [53,80]</td>
<td>.99 [.98,.99]</td>
<td>.99 [.98,.99]</td>
</tr>
<tr>
<td>CloudLasso</td>
<td>41s [36,48]</td>
<td>.99 [.98,.99]</td>
<td>.99 [.98,.99]</td>
</tr>
<tr>
<td>SpaceCast</td>
<td>18s [14,24]</td>
<td>.99 [.98,.99]</td>
<td>.99 [.98,.99]</td>
</tr>
<tr>
<td>PointCast</td>
<td>10s [8,8,12]</td>
<td>.99 [.95,.99]</td>
<td>.99 [.96,.99]</td>
</tr>
<tr>
<td>TraceCast</td>
<td>12s [11,14]</td>
<td>.99 [.98,1]</td>
<td>.99 [.98,1]</td>
</tr>
<tr>
<td><strong>Shell Dataset</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Cylinder</td>
<td>86s [71,105]</td>
<td>.97 [.97,.97]</td>
<td>.96 [.96,.97]</td>
</tr>
<tr>
<td>CloudLasso</td>
<td>46s [39,54]</td>
<td>.95 [.92,.96]</td>
<td>.94 [.91,.96]</td>
</tr>
<tr>
<td>SpaceCast</td>
<td>21s [17,25]</td>
<td>.95 [.86,.98]</td>
<td>.94 [.82,.97]</td>
</tr>
<tr>
<td>PointCast</td>
<td>10s [8,2,13]</td>
<td>.97 [.93,.98]</td>
<td>.97 [.93,.98]</td>
</tr>
<tr>
<td>TraceCast</td>
<td>11s [9,6,14]</td>
<td>.98 [.97,.98]</td>
<td>.97 [.97,.98]</td>
</tr>
<tr>
<td><strong>Rings Dataset</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Cylinder</td>
<td>66s [55,87]</td>
<td>.96 [.96,.97]</td>
<td>.96 [.94,.96]</td>
</tr>
<tr>
<td>CloudLasso</td>
<td>35s [27,45]</td>
<td>.98 [.97,.98]</td>
<td>.97 [.96,.98]</td>
</tr>
<tr>
<td>SpaceCast</td>
<td>16s [13,19]</td>
<td>.98 [.97,.98]</td>
<td>.97 [.96,.98]</td>
</tr>
<tr>
<td>PointCast</td>
<td>10s [7,9,13]</td>
<td>.97 [.91,.99]</td>
<td>.97 [.91,.98]</td>
</tr>
<tr>
<td>TraceCast</td>
<td>11s [9,7,13]</td>
<td>.98 [.97,.99]</td>
<td>.98 [.96,.98]</td>
</tr>
<tr>
<td><strong>Simulation Dataset</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Cylinder</td>
<td>161s [134,192]</td>
<td>.92 [.91,.93]</td>
<td>.92 [.91,.93]</td>
</tr>
<tr>
<td>CloudLasso</td>
<td>50s [43,59]</td>
<td>.94 [.93,.95]</td>
<td>.94 [.93,.95]</td>
</tr>
<tr>
<td>SpaceCast</td>
<td>40s [41,57]</td>
<td>.94 [.93,.95]</td>
<td>.94 [.93,.95]</td>
</tr>
<tr>
<td>PointCast</td>
<td>58s [51,65]</td>
<td>.94 [.89,.96]</td>
<td>.95 [.90,.96]</td>
</tr>
<tr>
<td>TraceCast</td>
<td>60s [51,70]</td>
<td>.95 [.93,.96]</td>
<td>.95 [.93,.96]</td>
</tr>
</tbody>
</table>

5 Discussion

Based on these results and selection strategies, we now compare the evaluated selection techniques. We not only discuss the relative strengths and weaknesses of each method, but also contemplate the users’ selection strategies in different situations.

5.1 Conceptual Foundation

Our new Cast techniques are all spatial, structure-aware selection techniques, a property they share with CloudLasso. However, we significantly improve upon CloudLasso as demonstrated by the user study. The fundamentally new idea put forward by the Cast methods is that the specific user interactions used to make the selection already contain rich information about a user’s intention, and consequently we take advantage of this information for the Cast techniques. Even without a precisely draw lasso we thus enable users to obtain accurate and fast results, based on our heuristics that guess a user’s intention. Ultimately, it is therefore the interacting user who decides which method to employ, based on the given dataset and exploration situation.

SpaceCast is conceptually closest to CloudLasso. Nevertheless, it improves upon it by only selecting a single component of the set of subspaces indicated by the lasso, facilitating different selection strategies: While CloudLasso can be used for multiple connected or non-connected targets in one step, SpaceCast allows users to specify a single intended target cluster though the shape of the drawn lasso.

TraceCast and PointCast deviate conceptually from SpaceCast and CloudLasso in that they no longer restrict the domain where a selection is made within some stroke or other input, thus no longer requiring an actual lasso to be drawn. Moreover, and more importantly, the conceptual basis of these two methods is that they treat clusters as distinct objects and no longer just as an arbitrary collection of particles. This new concept facilitates more freedom and flexibility in designing selection heuristics, either as simple click or touch operations with PointCast or using a shape recognition approach as in TraceCast.

5.2 Selection Scenarios

To better illustrate the different characteristics of the new selection methods we now discuss them with respect to changing selection scenarios: the selection of multiple clusters, the selection of a part of a cluster, and the selection of partially or completely occluded structures.

Multiple clusters

SpaceCast and TraceCast are based on a user-drawn 2D stroke, while PointCast requires pointing at the target. Point-based methods such as PointCast are usually faster for selecting single targets because they do not require any drawing. The trade-off lies in the resulting accuracy and needed fine-tuning since the 2D stroke used in

This issue is due to the interaction design with the mouse/pen-based input—for the future we envision to either provide a dedicated selection mode for such input modalities or to use bi-manual control for touch-based interaction.
SpaceCast and TraceCast provides more information about the likely user intention. Theoretically, all selection methods are able to select multiple clusters if there is no time limitation or restrictions on the number of Boolean operations. CloudLasso, in fact, supports selection of several clusters in one step, although users typically need to adjust the density threshold to arrive at the desired results (Fig. 11(a)). In the user study, however, we observed that participants tended to select clusters one-by-one and use Boolean operations, where necessary, to adjust their selection. Our new methods SpaceCast, TraceCast, and PointCast thus only select connected volumes and, therefore, require (at least) $N$ operations for the selection of $N$ clusters. The PointCast method is consequently particularly efficient for such selections of a few small clusters (Fig. 11(b)), while SpaceCast and TraceCast facilitate the selection of few more complex clusters.

**Partial selection** Both TraceCast and PointCast are based on the concept of treating clusters as individual objects, and they do not facilitate the direct selection of a subset of a cluster (although the latter is always possible through Boolean operations). SpaceCast (just as CloudLasso), in contrast, does facilitate partial cluster selection (Fig. 12) since the selection is always restricted within the drawn lasso.

**Occlusion** All Cast methods support the dedicated selection of partially occluded targets (Fig. 13)—in contrast to previous structure-aware selection methods. The impact of this property is reflected in the results of the post-session questionnaire, in which our participants rated all studied techniques on selecting in a noisy environment with occluded targets on a 7-point Likert scale. Cast methods were rated highly with an average of 6 (agree), CloudLasso was rated at an average of 4.5 (between no opinion and somewhat agree) and CylinderSelection received an average rating of 2.7 (between disagree and somewhat disagree). To achieve the selection of partially occluded clusters, SpaceCast requires users to trace the border of the target. PointCast, in contrast, only needs a small part of the target to not be occluded in order to facilitate picking it out from other clusters. TraceCast, finally, was regarded as the most effective method with respect to this point by the study participants. The reason was that, with TraceCast, the right cluster would be selected as long as its shape was closer to the input lasso than compared to that of other clusters—even if the target cluster was completely occluded.

### 5.3 Uncovering Global Structures

The selection volume created by PointCast selection depends to a large degree on the chosen density threshold, and at the same time is in no way restricted by a lasso. This property allows us to use PointCast to analyze how different parts of the dataset are connected with each other. In Fig. 14(a) we show an initial selection produced by PointCast using the automatically deduced density threshold, using a sample of the Millennium-II dataset [45] which represents galaxy halos. For the depicted selection case, the dataset contains more clusters that lie above the density threshold but which initially are not selected since they are not connected to the cluster under the clicked point. In Fig. 14(b) we show the situation when the density threshold has been lowered. Now, elongated structures have appeared that connect the previously disjoint clusters and thus produce a single connected volume that is now selected. This property of PointCast assists users in finding density concentrations in the dataset and helps them to understand how these are connected to each other on a more global level. The stroke based methods (SpaceCast, TraceCast, and the older CloudLasso) can produce similar structures, albeit with certain restrictions. Especially SpaceCast and CloudLasso can only reveal such connecting structures within the drawn lasso. TraceCast does not share this restriction, thus it can produce results similar to PointCast. Nevertheless, changing the density threshold can lead to the shape of the selection volume significantly diverging from the shape of the drawn lasso or stroke which may initially be surprising for some users.

### 5.4 Other Applications

The Cast methods were designed for selection in 3D particle datasets in general. While most examples we show in this paper come from astronomy, our methods can also be applied for selection in 3D particle datasets of different origin. Such datasets do not even need to have a spatial character but they could be based on other properties that can be mapped to 3D space. Such a representation can be generated from a multi-dimensional dataset by either choosing exactly three of these dimensions or by first performing a Principal Component Analysis and selecting the first three components. To show a practical example we present a subset of the Hurricane Isabel dataset in Fig. 15(a). We used the properties pressure, water vapor, and wind component and show selections with different Cast methods in Fig. 15(b)–(d).
The main limitation of the Cast selection methods is their dependence of an appropriate automatically determined density threshold. In datasets with steep density gradients it is a difficult problem to determine an appropriate initial value of the density threshold, and a manual adjustment of the threshold can improve the selection. Nevertheless, such manual adjustment is also a complication of the otherwise straight-forward interaction used by the Cast methods. Furthermore, density gradients are typically less steep at the edge of a cluster. This means that if a selection volume already roughly matches the cluster boundary, and the user tries to fine-tune the selection, then a small decrease of the density threshold produces a large increase of the selection volume. In the user study, however, we have not found this problem to affect the results.

Nevertheless, the occasional need to manually adjust the density threshold led some user study participants to remark that the TraceCast and PointCast methods are not as accurate as SpaceCast—in the latter the selection is always bound by the lasso. One way of improving this manual threshold setting would be by using a mapping from the slider position to the threshold value so that small movements of the slider produce correspondingly small changes of the selection volume. A very different approach would be to not use the density threshold for determining the cluster boundary but to use other edge-detection algorithms originating in the image recognition domain or using topological analysis to determine an appropriate density isosurface [21].

We note here that the Cast methods have been designed for selection in particle datasets with distinct clusters. Particle datasets with different structure characteristics may demand different heuristics for determining an appropriate selection volume.

Finally, in the future we want to combine different selection techniques in an integrated visualization and data analysis environment and perform user studies with real-world scenarios to validate the results of the present study. Since different methods are more appropriate for different types of selection, keeping only one of them in our toolkit would not be optimal. One possible combination would be that of the lasso-based CloudLasso which allows multiple selection with the point-based PointCast which allows efficient selection of individual clusters. Nevertheless, the user study showed that SpaceCast and TraceCast are also very strong contenders and a more thorough and extended study in real-world environments would be necessary to help decide between these very strong selection methods: CloudLasso, SpaceCast, TraceCast, and PointCast. Potentially, all of them could be included, together with Boolean operations that facilitate further fine-tuning.

5.5 Limitations and Further Improvements

The main limitation of the Cast selection methods is their dependence of an appropriate automatically determined density threshold. In datasets with steep density gradients it is a difficult problem to determine an appropriate initial value of the density threshold, and a manual adjustment of the threshold can improve the selection. Nevertheless, such manual adjustment is also a complication of the otherwise straight-forward interaction used by the Cast methods. Furthermore, density gradients are typically less steep at the edge of a cluster. This means that if a selection volume already roughly matches the cluster boundary, and the user tries to fine-tune the selection, then a small decrease of the density threshold produces a large increase of the selection volume. In the user study, however, we have not found this problem to affect the results.

Nevertheless, the occasional need to manually adjust the density threshold led some user study participants to remark that the TraceCast and PointCast methods are not as accurate as SpaceCast—in the latter the selection is always bound by the lasso. One way of improving this manual threshold setting would be by using a mapping from the slider position to the threshold value so that small movements of the slider produce correspondingly small changes of the selection volume. A very different approach would be to not use the density threshold for determining the cluster boundary but to use other edge-detection algorithms originating in the image recognition domain or using topological analysis to determine an appropriate density isosurface [21].

We note here that the Cast methods have been designed for selection in particle datasets with distinct clusters. Particle datasets with different structure characteristics may demand different heuristics for determining an appropriate selection volume.

Finally, in the future we want to combine different selection techniques in an integrated visualization and data analysis environment and perform user studies with real-world scenarios to validate the results of the present study. Since different methods are more appropriate for different types of selection, keeping only one of them in our toolkit would not be optimal. One possible combination would be that of the lasso-based CloudLasso which allows multiple selection with the point-based PointCast which allows efficient selection of individual clusters. Nevertheless, the user study showed that SpaceCast and TraceCast are also very strong contenders and a more thorough and extended study in real-world environments would be necessary to help decide between these very strong selection methods: CloudLasso, SpaceCast, TraceCast, and PointCast. Potentially, all of them could be included, together with Boolean operations that facilitate further fine-tuning.

6 Conclusion

With SpaceCast, TraceCast, and PointCast we introduced a new family of context-aware interactive selection techniques. Through a set of well-designed heuristics, these techniques derive a user’s selection intention from the input he or she has provided. The techniques are effective because together they cover a wide range of possible selection goals, based on each technique’s individual selection algorithm and user input gestures. Furthermore, our study showed that our new techniques are efficient as they allow users to arrive at selections faster than with other structure-aware techniques, without jeopardizing their accuracy.

Ultimately, our paper contributes to the development of flexible interaction environments for the exploration of complex data visualizations [29]. As selection is needed in virtually any exploratory data analysis, our new Cast family of techniques—together with the older CloudLasso and Boolean operations—provides a powerful toolkit that can support many different selection tasks and datasets in visualization. While we did not specifically design the techniques for any particular type of input device, SpaceCast, TraceCast, and PointCast are suitable not only for mouse and pen-based input but also for newer input and interaction paradigms [30] such as those based on touch (e.g., [11, 25, 26, 57]).